Intro to COSICC

Magdalena Strauss

2024-11-15

Welcome to COSICC! COSICC R an R package for **CO**mparative analysis of **S**ingle-**C**ell-RNA-seq data for **C**omplex cell populations. This vignette gives an introduction and overview to COSICC.

Overview and use cases

For exact reproducibility, we set a fixed seed.

```
set.seed(112244)
suppressPackageStartupMessages(library(COSICC))
suppressPackageStartupMessages(library(SingleCellExperiment))
```

COSICC_DA_group

COSICC_DA_group tests for differential abundance of groups of cells (.e.g. cell types) after perturbations, compared to a control experiment without perturbations. This type of set-up occurs for many experiments. One example is patient data. Assume we have an scRNA-seq data set of a cancer patient at the day of diagnosis, and another one from after treatment. Both samples contain normal and malignant cells, and different cell types.

Simulating data

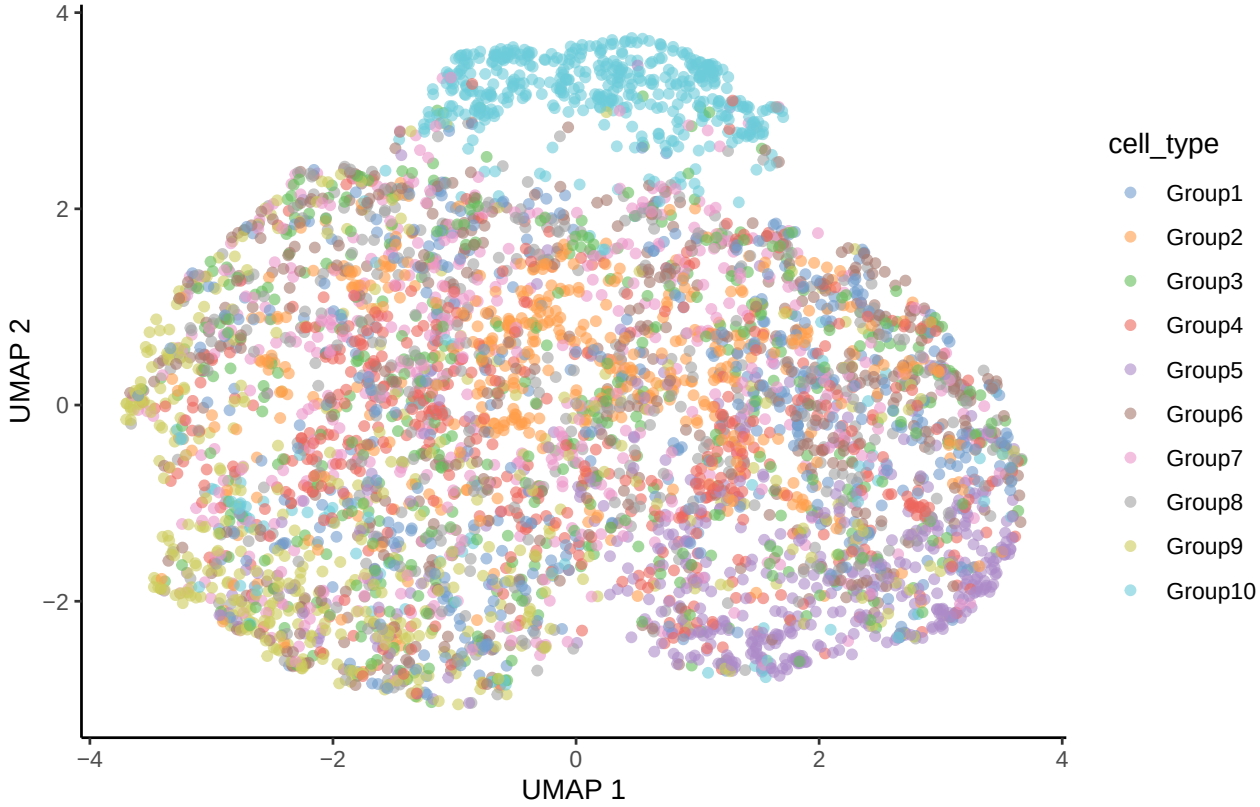
Simulating the control experiment

The control experiment is an experiment without the perturbation studied. For instance, it could be a sample from a cancer patient at the day of diagnosis if the perturbation we want to look at is the treatment following the diagnosis. Or it could be a control chimera experiment as in Strauss et al. (2023), where unperturbed stem cells were injected into a mouse embryo at an early developmental stage, which served as a control experiment for injected cells with a gene knockout.

We simulate data using the Splatter Bioconductor package (Zappia, Phipson, and Oshlack (2017)). We simulate 10 cell types with an average of 500 cells each, and visualise the data on a UMAP.

```
sce_sim_cell_type_control <- splatSimulate(
    nGenes = 10,
    batchCells = 5000,
    group.prob = rep(0.1,10),
    method = "groups",
    verbose = FALSE
)

sce_sim_cell_type_control <- logNormCounts(sce_sim_cell_type_control)
sce_sim_cell_type_control <- runPCA(sce_sim_cell_type_control,ncomponents=3)
sce_sim_cell_type_control <- runUMAP(sce_sim_cell_type_control)
```

```
names(colData(sce_sim_cell_type_control))[3] <- "cell_type"

plotReducedDim(sce_sim_cell_type_control, dimred="UMAP",colour_by = "cell_type") +
    theme_classic()
```

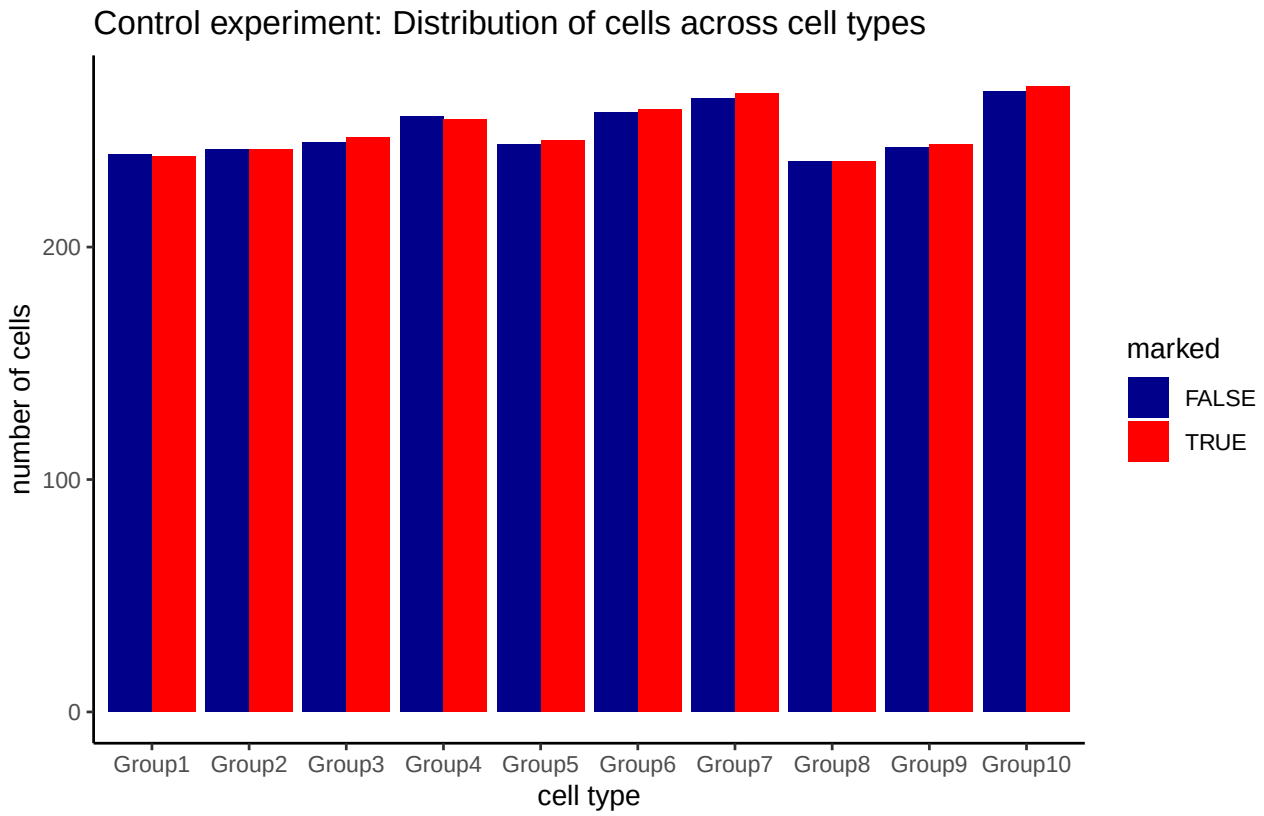


For each of the cell types, we label 50% of the cells as normal and 50% as marked. In practice, this may be a mock perturbation (e.g. injected cells for chimeras marked with fluorescent markers), malignant cells in a cancer at time of diganosis etc.

```
sce_sim_cell_type_control$marked <- FALSE
for (j in 1:10){
    xx <- which(sce_sim_cell_type_control$cell_type == paste0("Group",j))
    sce_sim_cell_type_control$marked[sample(xx,floor(length(xx) * 0.5+
                                    (runif(1)<0.5)))]  <- TRUE
}
```

We plot the distribution of cell types.

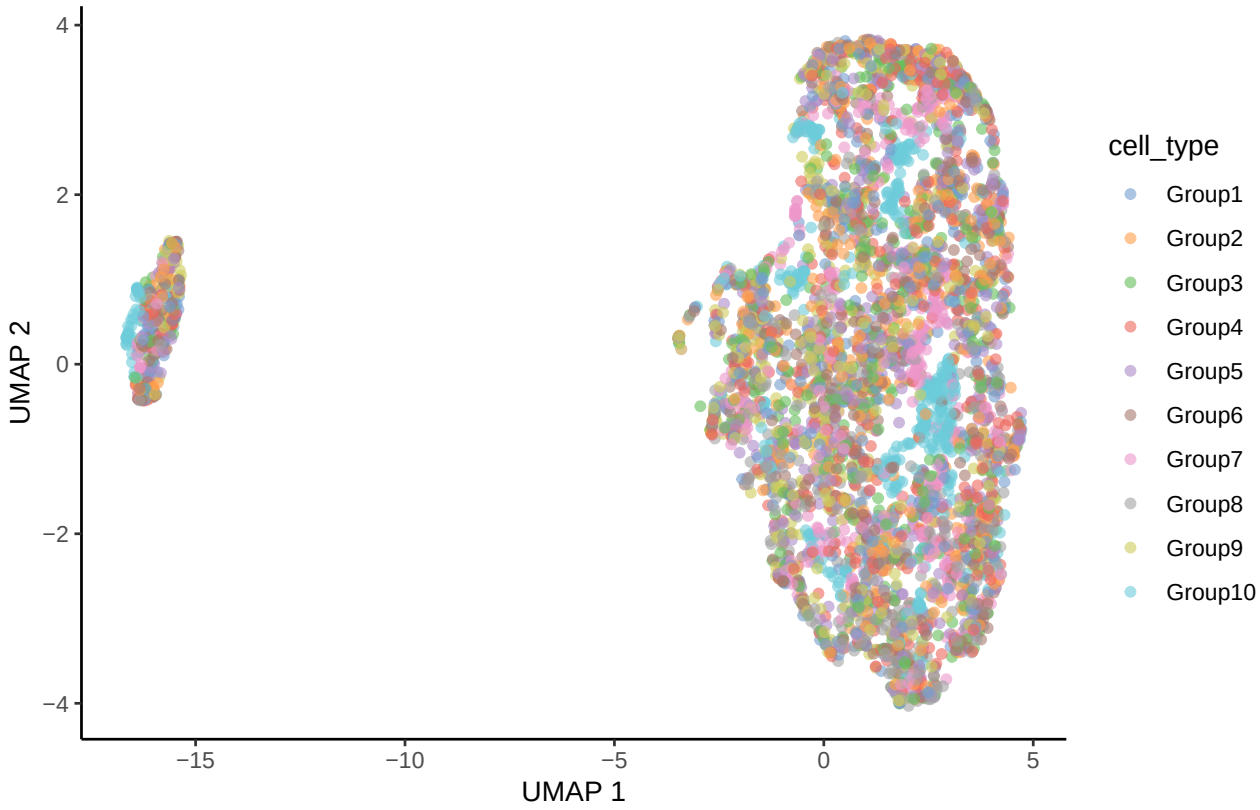
```
table_cell_types <-
    as.data.frame(colData(sce_sim_cell_type_control)[,c("cell_type","marked")]) %>%
    dplyr::group_by_all() %>% dplyr::count()

ggplot(table_cell_types,aes(x=cell_type,y=n,fill=marked)) +
    geom_bar(stat="identity",position="dodge")+theme_classic()+
    xlab("cell type") + ylab("number of cells") +
    ggtitle("Control experiment: Distribution of cells across cell types")+
    scale_fill_manual(values=c("FALSE"="darkblue","TRUE"="red"))
```



Simulating the experiment with perturbation

For the experiment with perturbation, we first perform the same simulation as above. Note that while the parameters are identical, this is a separate independent simulation.

```
sce_sim_cell_type_case <- splatSimulate(
    nGenes = 10,
    batchCells = 5000,
    group.prob = rep(0.1,10),
    method = "groups",
    verbose = FALSE
)

sce_sim_cell_type_case <- logNormCounts(sce_sim_cell_type_case)
sce_sim_cell_type_case <- runPCA(sce_sim_cell_type_case,ncomponents=3)
sce_sim_cell_type_case <- runUMAP(sce_sim_cell_type_case)
names(colData(sce_sim_cell_type_case))[3] <- "cell_type"

plotReducedDim(sce_sim_cell_type_case, dimred="UMAP",colour_by = "cell_type") +
    theme_classic()
```




```
sce_sim_cell_type_case$marked <- FALSE
for (j in 1:10){
    xx <- which(sce_sim_cell_type_case$cell_type == paste0("Group",j))
    sce_sim_cell_type_case$marked[sample(xx,floor(length(xx) * 0.5 +
                                          (runif(1) <= 0.5)))]  <- TRUE
}
```

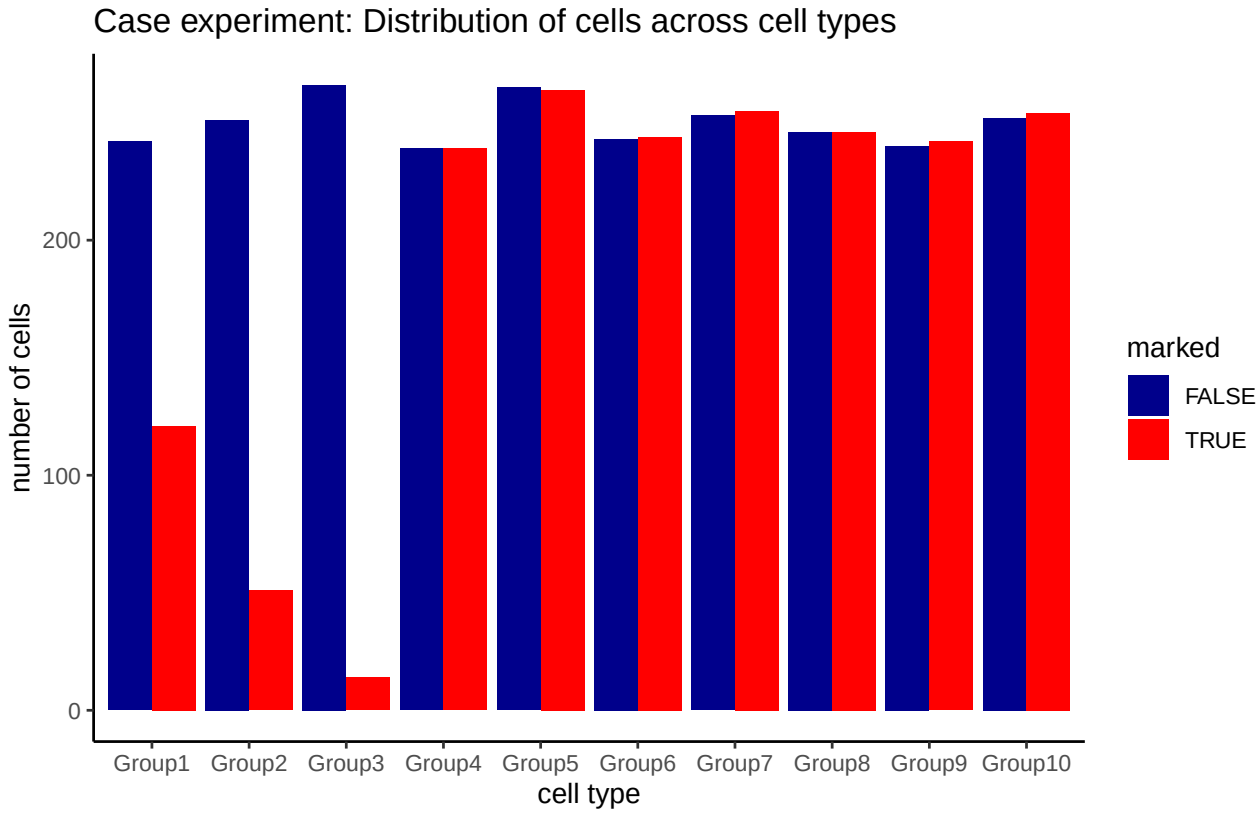
Now we assume that Group1 and Group2 and Group3 have been affected by a perturbation and therefore the marked cells have been depleted to 50%, 20% and 5% of their original number of cells, respectively.

```
marked_cells_group1 <- which(sce_sim_cell_type_case$marked &
                             sce_sim_cell_type_case$cell_type=="Group1")
marked_cells_group2 <- which(sce_sim_cell_type_case$marked &
                             sce_sim_cell_type_case$cell_type=="Group2")
marked_cells_group3 <- which(sce_sim_cell_type_case$marked &
                             sce_sim_cell_type_case$cell_type=="Group3")
depleted_cells <- c(sample(marked_cells_group1,floor(0.5*length(marked_cells_group1) +
                                                     (runif(1)<0.5))),
                    sample(marked_cells_group2,floor(0.8*length(marked_cells_group2) +
                                                     (runif(1)<0.5))),
                    sample(marked_cells_group3,floor(0.95*length(marked_cells_group3) +
                                                     (runif(1)<0.5))))
sce_sim_cell_type_case <- sce_sim_cell_type_case[,-depleted_cells]
```

We now plot the distribution of cells across cell types.

```
table_cell_types <-
    as.data.frame(colData(sce_sim_cell_type_case)[,c("cell_type","marked")]) %>%
    dplyr::group_by_all() %>% dplyr::count()
```

```
ggplot(table_cell_types,aes(x=cell_type,y=n,fill=marked)) +
    geom_bar(stat="identity",position="dodge")+theme_classic()+
    xlab("cell type") + ylab("number of cells") +
    ggtitle("Case experiment: Distribution of cells across cell types")+
    scale_fill_manual(values=c("FALSE"="darkblue","TRUE"="red"))
```



For experiments with visually marked cells, e.g. using fluorescent markers as in Pijuan-Sala et al. (2019), Guibentif et al. (2021), or Strauss et al. (2023), the following sampling bias occurs: experimentalists aim for approximately 50% (or a different fixed proportion) of cells with the fluorescent markers, which means that the non-depleted cell types will appear enriched in marked cells.

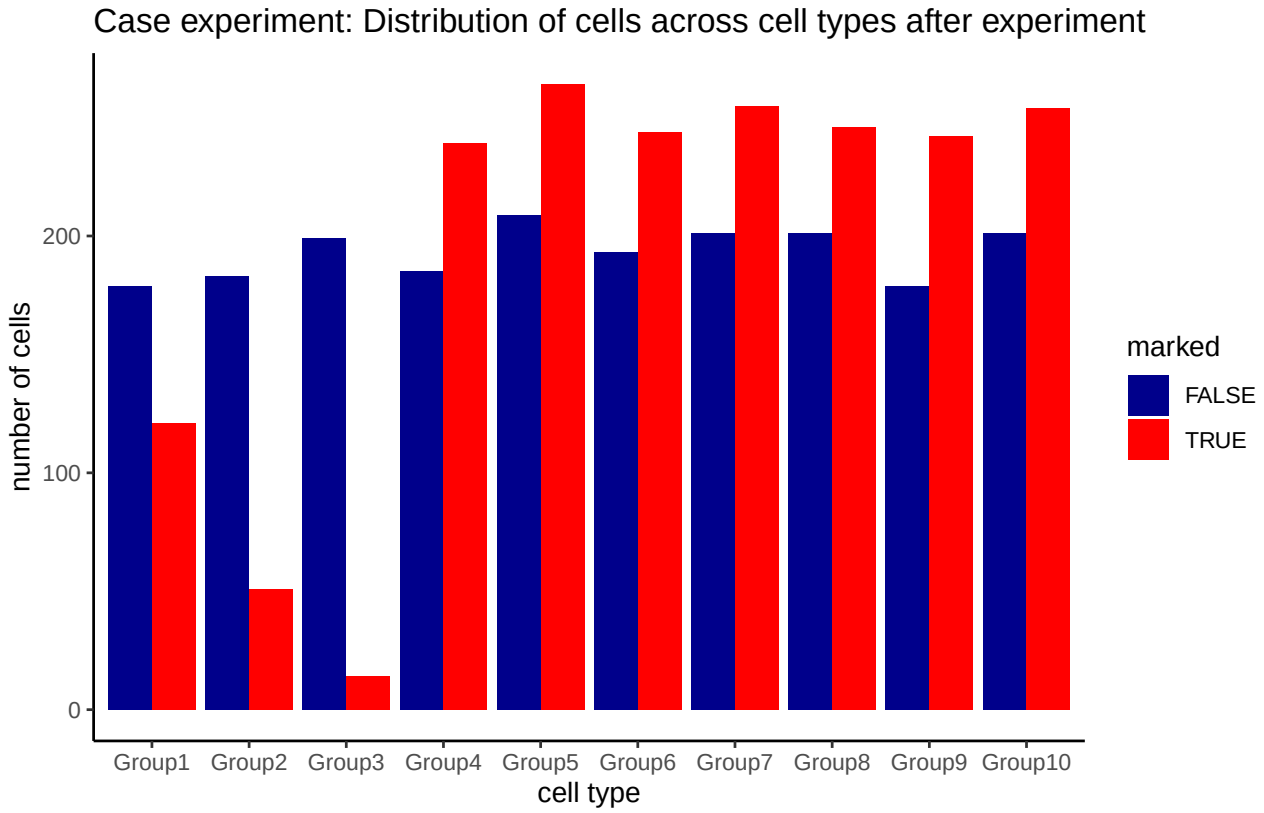
We simulate the experimental bias described above by sampling an equal number of marked and unmarked cells.

```
marked_cells <- which(sce_sim_cell_type_case$marked)
unmarked_cells <- which(!(sce_sim_cell_type_case$marked))
sce_sim_cell_type_case_sampled <-
    sce_sim_cell_type_case[,c(sample(marked_cells,length(marked_cells)),
                sample(unmarked_cells,length(marked_cells)))]
```

We plot the resulting distribution of cells across cell types. The unaffected cell types now appear to be enriched for marked cells. However, this is only because of the experimental bias.

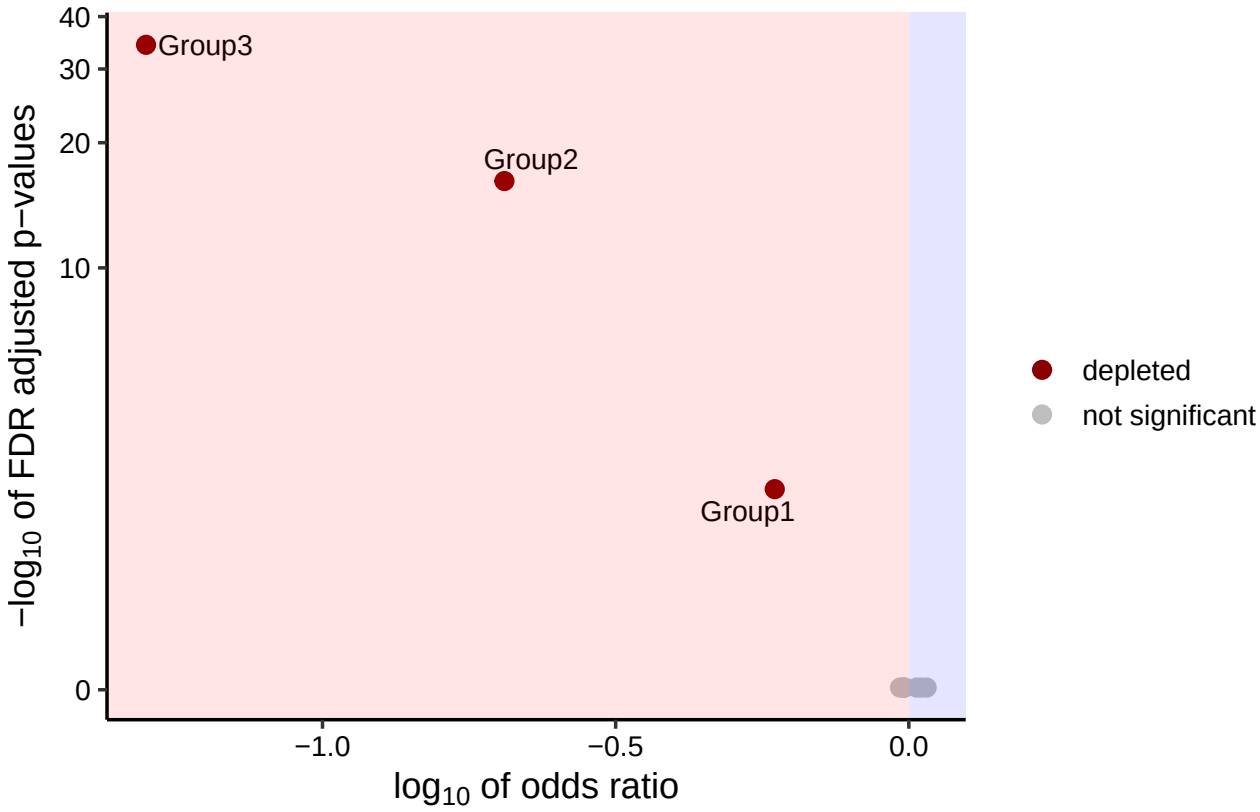
```
table_cell_types <-
    as.data.frame(colData(sce_sim_cell_type_case_sampled)[,c("cell_type","marked")]) %>%
    dplyr::group_by_all() %>% dplyr::count()

ggplot(table_cell_types,aes(x=cell_type,y=n,fill=marked)) +
    geom_bar(stat="identity",position="dodge")+theme_classic()+
    xlab("cell type") + ylab("number of cells") +
```

```
    ggtitle("Case experiment: Distribution of cells across cell types after experiment")+
    scale_fill_manual(values=c("FALSE"="darkblue","TRUE"="red"))
```



Below we show that, by running COSICC_DA_group, we are able to identify the different levels of depletion for the three cell type correctly, as COSICC_DA_group is able to correct for the spurious enrichment for marked cells resulting from the experimental bias explained above.

```
COSICC_DA_result <-
    COSICC_DA_group(sce_case=sce_sim_cell_type_case_sampled,
                        sce_control=sce_sim_cell_type_control)
```



Note about method and applicability

COSICC_DA_group assumes that the true median depletion or enrichment of marked cells across all cell types is 0. In our example, at most 4 groups could be depleted for the method to identify the true enrichment and depletion.

COSICC_kinetics

COSICC_kinetics identifies whether, as a result of a perturbation, there is significant delay or acceleration along a lineage trajectory.

Simulating data

Like for COSICC_DA_group above, we first simulate control data using the Splatter package.

Simulating the control experiment

```
sce_kinetics_control <-
splatSimulatePaths(batchCells=500,verbose=FALSE,nGenes=10,de.prob=0.5)
```

Splatter simulates developmental progression as "Step", we rename as pt (pseudotime), to apply COSICC_kinetics.

```
sce_kinetics_control$pt <- sce_kinetics_control$Step
sce_kinetics_control$Step <- NULL
```

We mark cells, with somewhat more cells marked from the earlier part of the lineage trajectory. This simulates an effect in the control experiment caused by experimental and not biological reasons.

```
sce_kinetics_control$marked <- (1:ncol(sce_kinetics_control)) %in% sample(1:ncol(sce_kinetics_control),
cells_higher_pt_marked <- colnames(sce_kinetics_control[,sce_kinetics_control$marked & sce_kinetics_con
cells_depleted_experimental <- sample(cells_higher_pt_marked,40)
sce_kinetics_control <- sce_kinetics_control[,!(sce_kinetics_control$Cell %in% cells_depleted_experimen
```

We compute normalised log-counts and PCA.

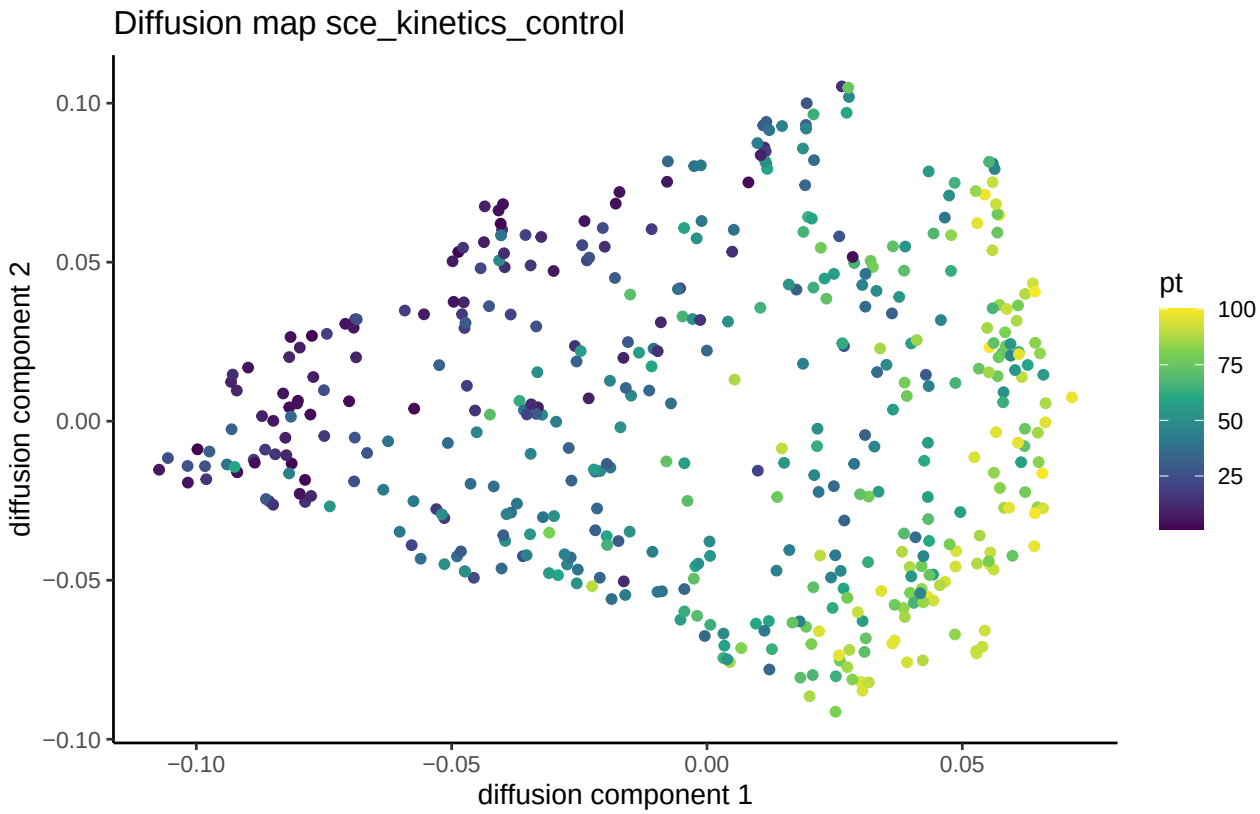
```
sce_kinetics_control <- logNormCounts(sce_kinetics_control)
sce_kinetics_control <- runPCA(sce_kinetics_control,ncomponents=3)
```

We create a plot of a diffusion map for the data set to illustrate pseudotime and progression along the lineage trajectory.

```
dm <- DiffusionMap(reducedDims(sce_kinetics_control)$PCA)
```

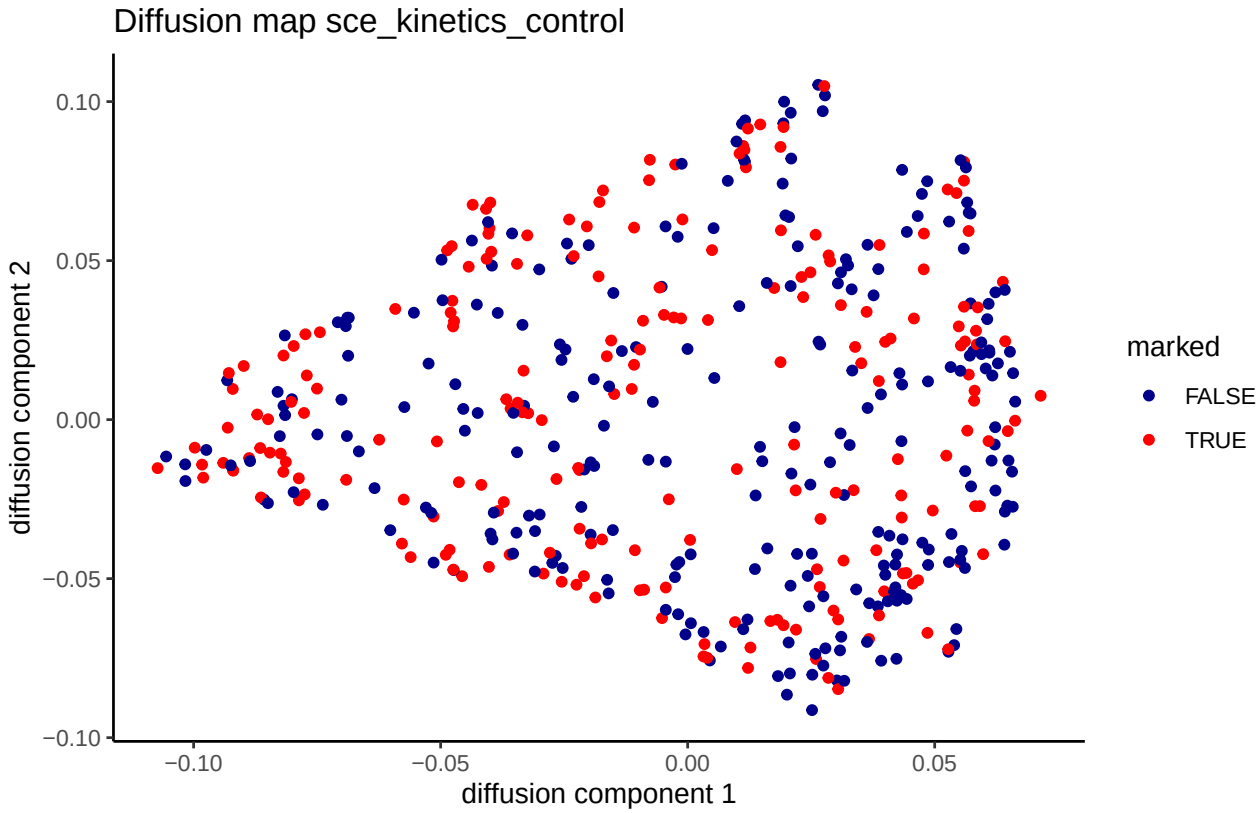
```
## 'as(<dsCMatrix>, "dgTMatrix")' is deprecated.
## Use 'as(as(., "generalMatrix"), "TsparseMatrix")' instead.
## See help("Deprecated") and help("Matrix-deprecated").
```

```
xx <- sample(1:length(dm$DC1))
tmp <- data.frame(DC1 = eigenvectors(dm)[xx, 1],
                  DC2 = eigenvectors(dm)[xx, 2],
                  pt = sce_kinetics_control$pt[xx],
                  marked = sce_kinetics_control$marked[xx])
ggplot(tmp,mapping=aes(x=DC1,y=DC2,color=pt)) + geom_point() + theme_classic() +
    xlab("diffusion component 1") + ylab("diffusion component 2") +
    scale_colour_viridis_c()+
    ggtitle("Diffusion map sce_kinetics_control")
```



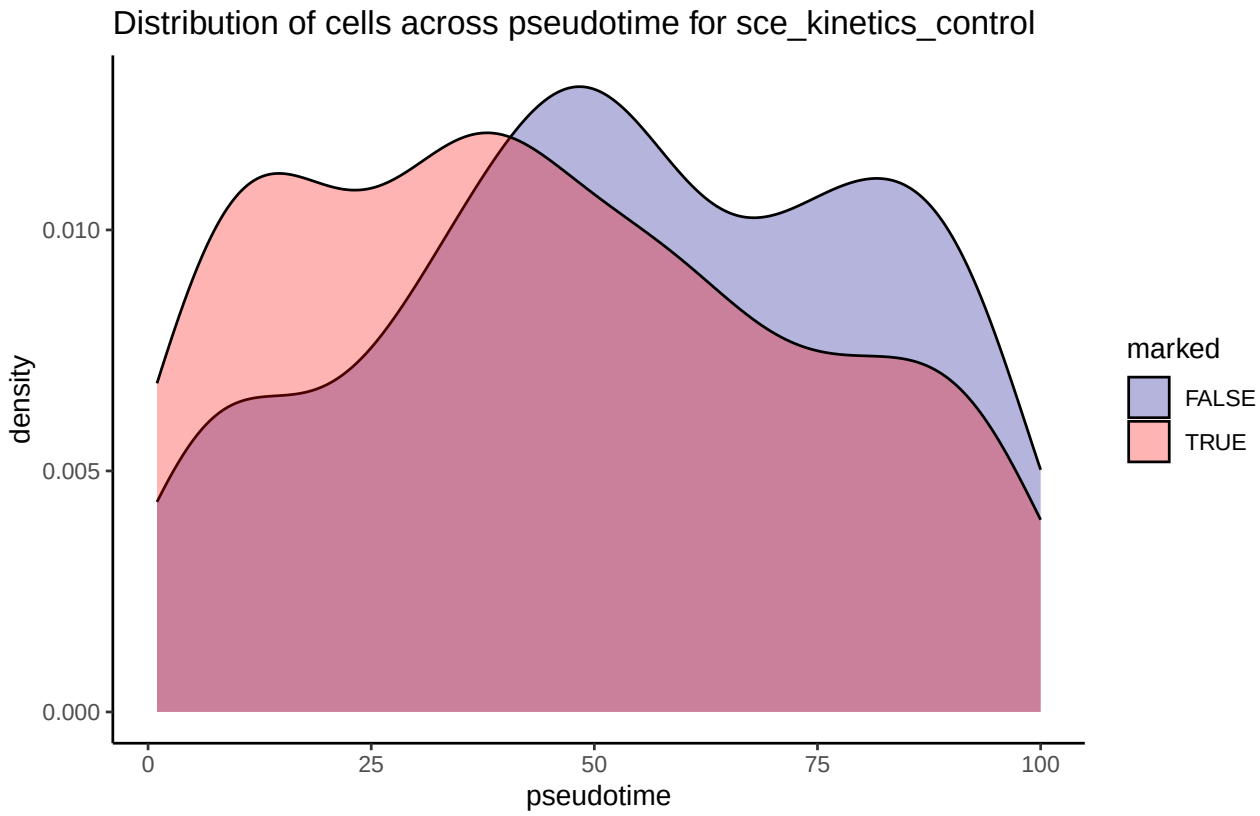
Now we repeat the plot, but coloured by whether the cells are marked.

```
ggplot(tmp,mapping=aes(x=DC1,y=DC2,color=marked)) + geom_point() + theme_classic() +
    xlab("diffusion component 1") + ylab("diffusion component 2") +
    scale_color_manual(values=c("FALSE"="darkblue","TRUE"="red"))+
    ggtitle("Diffusion map sce_kinetics_control")
```



We also plot the distributions of marked an unmarked cells across pseudotime. The figure below shows that there is some difference between pseudotimes for marked and unmarked cells for the control data set. This illustrates a possible experimental effect.

```
ggplot(colData(sce_kinetics_control),aes(fill=marked,x=pt)) + geom_density(alpha=0.3)+theme_classic() +
    ggtitle("Distribution of cells across pseudotime for sce_kinetics_control")+
    xlab("pseudotime")
```



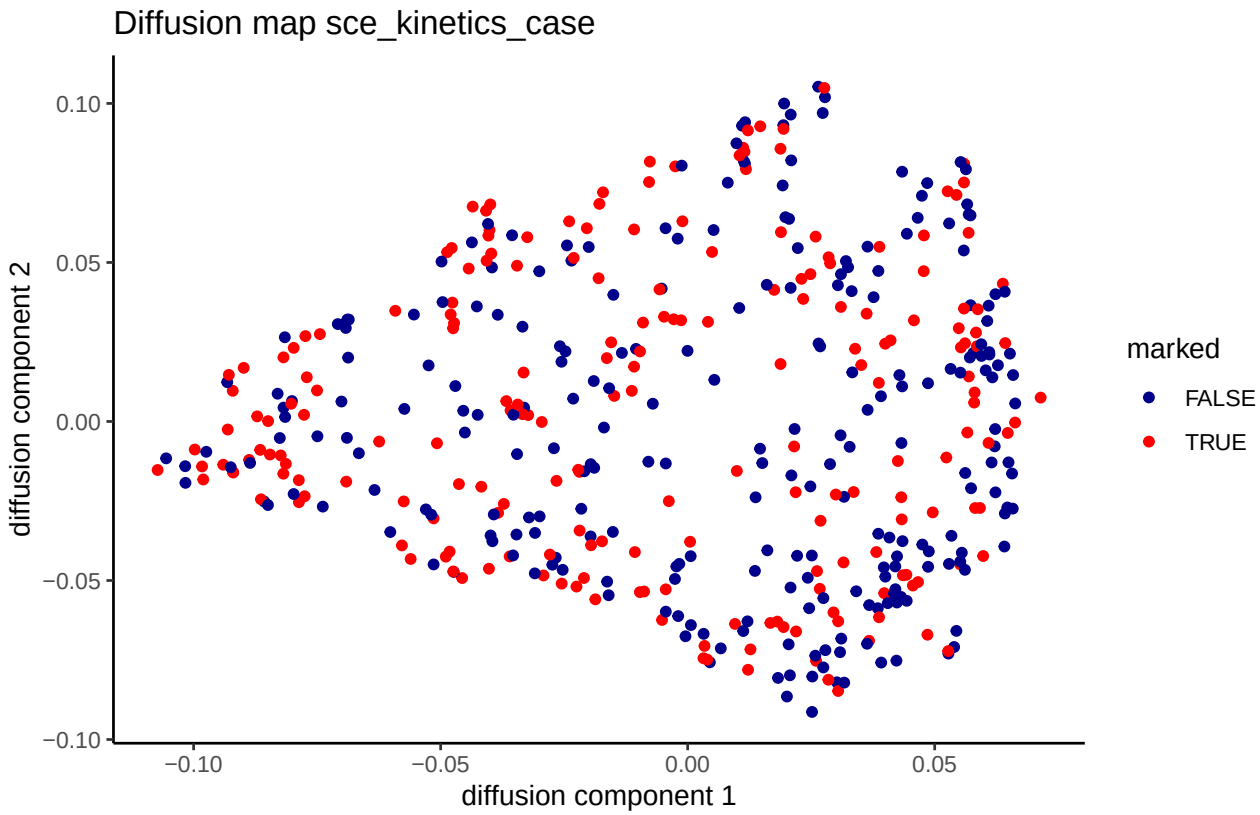
Simulating the experiment with perturbation

To simulate a perturbation that delays progression along a lineage trajectory, we define probabilites for marking cells along the lineage trajectory that decrease with progression along the lineage trajectory.

```
probs <- sort(runif(ncol(sce_kinetics_control)),decreasing=TRUE)
cells_higher_pt_marked <- colnames(sce_kinetics_control[,sce_kinetics_control$marked & sce_kinetics_cont
kinetics_depleted_biological <- sample(cells_higher_pt_marked,80)
sce_kinetics_case <- sce_kinetics_control[,!(colnames(sce_kinetics_control)%in%kinetics_depleted_biologi
```

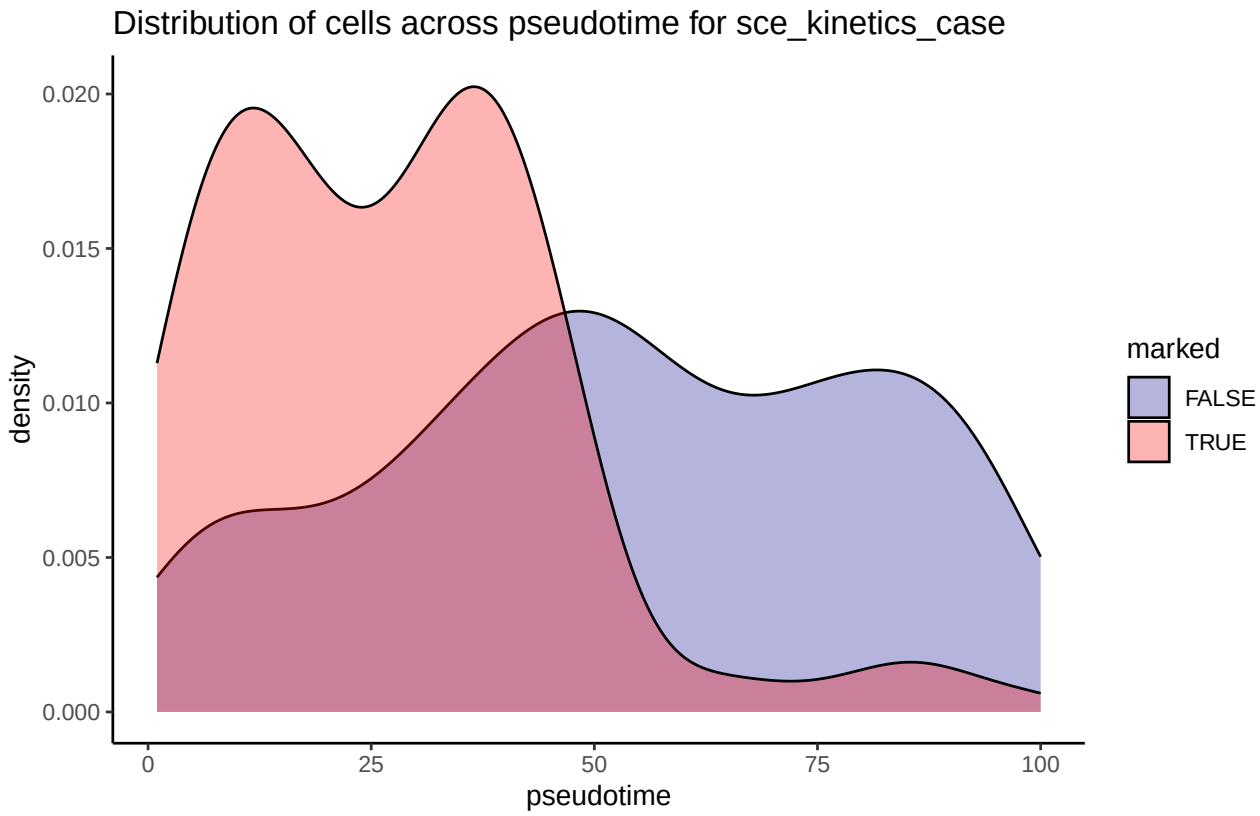
We plot the perturbed data set on a UMAP.

```
tmp <- data.frame(DC1 = eigenvectors(dm)[xx, 1],
                  DC2 = eigenvectors(dm)[xx, 2],
                  pt = sce_kinetics_case$pt[xx],
                  marked = sce_kinetics_control$marked[xx])

ggplot(tmp,mapping=aes(x=DC1,y=DC2,color=marked)) + geom_point() +
    theme_classic() +
    xlab("diffusion component 1") + ylab("diffusion component 2") +
    scale_color_manual(values=c("FALSE"="darkblue","TRUE"="red"))+
    ggtitle("Diffusion map sce_kinetics_case")
```



We also plot the distributions of marked an unmarked cells across pseudotime. The figure below shows that a difference between pseudotimes for marked and unmarked cells for the perturbation data set, which is larger than the effect seen for the control data. This illustrates a biological effect.

```
ggplot(colData(sce_kinetics_case),aes(fill=marked,x=pt)) +
    geom_density(alpha=0.3)+theme_classic() +
    scale_fill_manual(values=c("FALSE"="darkblue","TRUE"="red"))+
    ggtitle("Distribution of cells across pseudotime for sce_kinetics_case")+
    xlab("pseudotime")
```



Identification of developmental delay with COSICC_kinetics

We now run COSICC_kinetics to test whether the delay in development along the lineage trajectory seen in the figure above is significant. Indeed, COSICC_kinetics identifies a significant delay of marked versus unmarked cells for the perturbed case data. The fact that convidence_interval_overlapping_with_control = FALSE means that the 95% confidence intervals for the case and the control data do not overlap. Therefore, while the control data also shows a developmental delay, the one for the perturbed case data set may be seen as significantly more pronounced. COSICC_kinetics thereby allows the detection of significant developmental delay, while also comparing the this delay to the control data set in a principled manner.

```
COSICC_kinetics(sce_case=sce_kinetics_case,sce_control=sce_kinetics_control)
```

```
## $p_value
## [1] 1.212374e-18
##
## $convidence_interval_overlapping_with_control
## [1] FALSE
##
## $sig
## [1] TRUE
```

Illustration of COSICC_DA_lineage

COSICC_DA_lineage tests for differential abundance of lineages after perturbations, compared to a control experiment without perturbations. The input for lineage contribution of cells is a score for each cell that reflects the probability of the cell being part of the lineage, e.g. Waddington-OT scores ("Optimal-Transport Analysis of Single-Cell Gene Expression Identifies Developmental Trajectories in Reprogramming" (2019)).

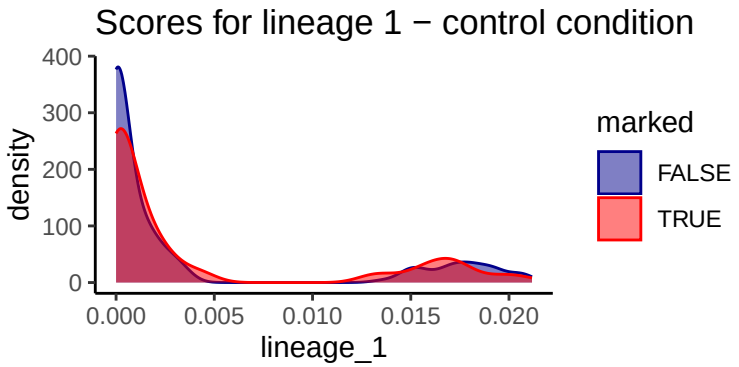
Simulation of lineage scores

We simulate scores for 5 lineages, with one lineage (lineage_1) reduced for marked cells in the perturbed condition. First, we simulate the scores for the control condition.

```
sim_lineage_scores_control <- matrix(rnorm(1500) * 0.1,nrow=300,ncol=5)
for (j in 1:5){
  sim_lineage_scores_control[((j-1)*60+1):(j*60),j] <- rnorm(60) *0.1+0.85
}
sim_lineage_scores_control <- apply(sim_lineage_scores_control,1,function(x) x/colSums(sim_lineage_scor
sim_lineage_scores_control <- mapply(function(x) min(x,1),sim_lineage_scores_control)
sim_lineage_scores_control <- mapply(function(x) max(x,0),sim_lineage_scores_control)
dim(sim_lineage_scores_control) <- c(300,5)
colnames(sim_lineage_scores_control) <- paste0("lineage_",1:5)
rownames(sim_lineage_scores_control) <- paste0("cell_",1:300,"_control")
sim_lineage_scores_control <- as.data.frame(sim_lineage_scores_control)
colData_DA_lineage_control <- data.frame(cell=rownames(sim_lineage_scores_control),
                              marked = sample(c(TRUE,FALSE),300,replace=TRUE))
sim_lineage_scores_control$id <- colData_DA_lineage_control$cell
colData_DA_lineage_control$cell <- sim_lineage_scores_control$id
sce_DA_lineage_control <- SingleCellExperiment(colData=colData_DA_lineage_control)
```

We plot the scores for lineage 1 for the control condition.

```
df <- as.data.frame(sim_lineage_scores_control)
df$marked <- colData_DA_lineage_control$marked
ggplot(df,aes(x=lineage_1,color=marked,fill=marked)) +
  geom_density(alpha=0.5) +
  scale_color_manual(values=c("FALSE"="darkblue","TRUE"="red"))+
  scale_fill_manual(values=c("FALSE"="darkblue","TRUE"="red"))+
  theme_classic()+
  labs(title="Scores for lineage 1 - control condition")
```



Now we simulate the unmarked cells for the perturbed condition.

```
sim_lineage_scores_case_unmarked <- matrix(rnorm(750) * 0.1,nrow=150,ncol=5)
for (j in 1:5){
  sim_lineage_scores_case_unmarked[((j-1)*30+1):(j*30),j] <- rnorm(30) *0.1+0.85
}
sim_lineage_scores_case_unmarked <- mapply(function(x) min(x,1),sim_lineage_scores_case_unmarked)
sim_lineage_scores_case_unmarked <- mapply(function(x) max(x,0),sim_lineage_scores_case_unmarked)
dim(sim_lineage_scores_case_unmarked) <- c(150,5)
colnames(sim_lineage_scores_case_unmarked) <- paste0("lineage_",1:5)
rownames(sim_lineage_scores_case_unmarked) <- paste0("cell_",1:150,"_case")
```

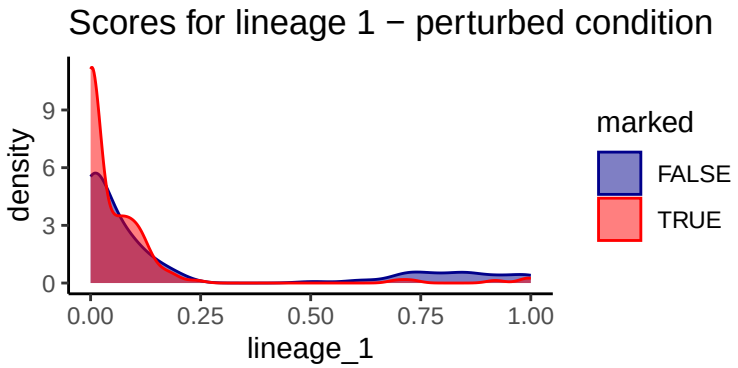
```
sim_lineage_scores_case_unmarked <- as.data.frame(sim_lineage_scores_case_unmarked)
colData_DA_lineage_case_unmarked <- data.frame(cell=rownames(sim_lineage_scores_case_unmarked),
                              marked = rep(FALSE,150))
sim_lineage_scores_case_unmarked$id <- colData_DA_lineage_case_unmarked$cell
colData_DA_lineage_case_unmarked$cell <- sim_lineage_scores_case_unmarked$id
sce_DA_lineage_case_unmarked <- SingleCellExperiment(colData=colData_DA_lineage_case_unmarked)
```

We simulate the marked cells for the perturbed condition, with fewer cells assigned to lineage 1.

```
sim_lineage_scores_case_marked <- matrix(rnorm(750) * 0.1,nrow=150,ncol=5)
sim_lineage_scores_case_marked[sample(1:30,5),1] <- rnorm(5) *0.1+0.85 # a
# smaller number of cells is assigned to lineage 1 for the perturbed condition
for (j in 2:5){
  sim_lineage_scores_case_marked[((j-1)*30+1):(j*30),j] <- runif(30) *0.1+0.85
}
sim_lineage_scores_case_marked <- mapply(function(x) min(x,1),sim_lineage_scores_case_marked)
sim_lineage_scores_case_marked <- mapply(function(x) max(x,0),sim_lineage_scores_case_marked)
dim(sim_lineage_scores_case_marked) <- c(150,5)
colnames(sim_lineage_scores_case_marked) <- paste0("lineage_",1:5)
rownames(sim_lineage_scores_case_marked) <- paste0("cell_",151:300,"_case")
sim_lineage_scores_case_marked <- as.data.frame(sim_lineage_scores_case_marked)
colData_DA_lineage_case_marked <- data.frame(cell=rownames(sim_lineage_scores_case_marked),
                              marked = rep(TRUE,150))
sim_lineage_scores_case_marked$id <- colData_DA_lineage_case_marked$cell
colData_DA_lineage_case_marked$cell <- sim_lineage_scores_case_marked$id
sce_DA_lineage_case <-
  SingleCellExperiment(colData=rbind(colData_DA_lineage_case_marked,
                       colData_DA_lineage_case_unmarked),
                       )
```

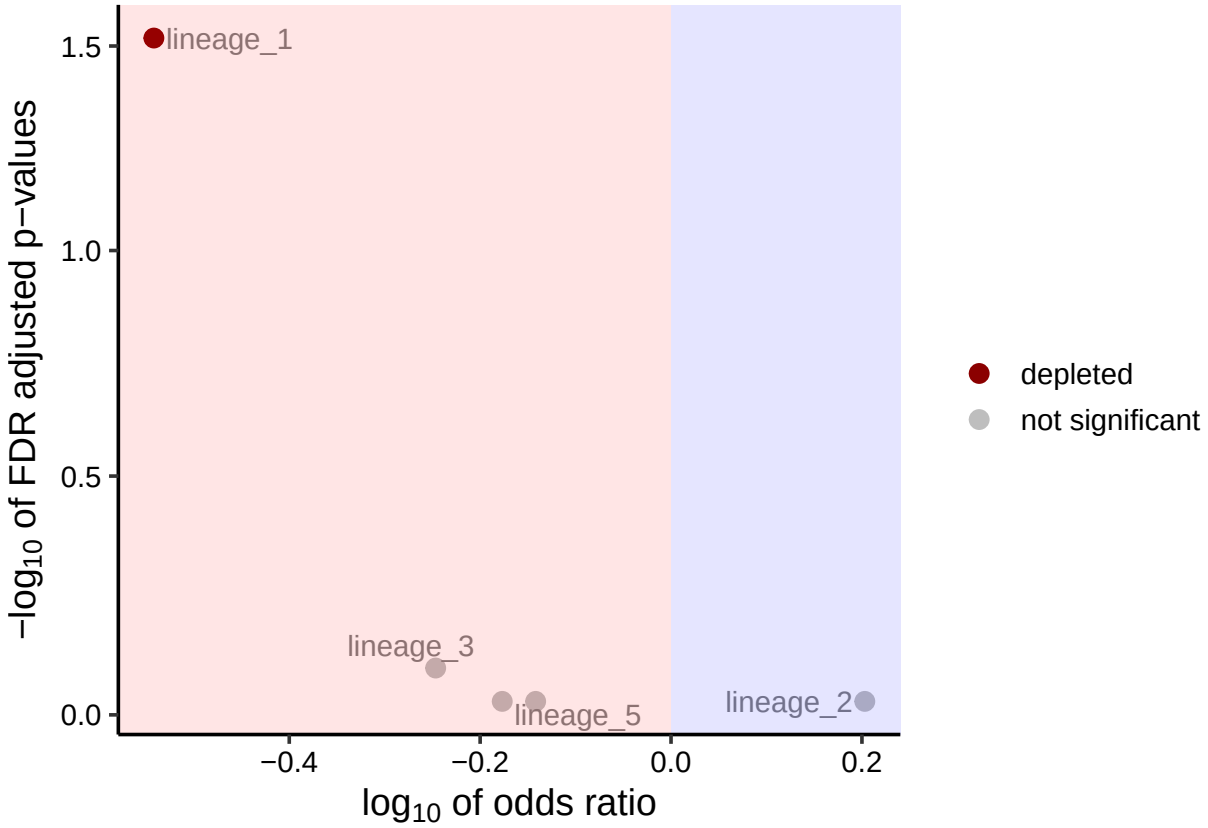
We plot the scores for lineage 1 for the perturbed condition.

```
df <- rbind(sim_lineage_scores_case_marked,sim_lineage_scores_case_unmarked)
df$marked <- sce_DA_lineage_case$marked
ggplot(df,aes(x=lineage_1,color=marked,fill=marked)) +
  geom_density(alpha=0.5) +
  scale_color_manual(values=c("FALSE"="darkblue","TRUE"="red"))+
  scale_fill_manual(values=c("FALSE"="darkblue","TRUE"="red"))+
  theme_classic()+
  labs(title="Scores for lineage 1 - perturbed condition")
```



We apply COSICC_DA_lineage and identify the depletion of lineage 1, illustrated in the plot below.

```
lineage_scores <- rbind(sim_lineage_scores_case_marked,
                        sim_lineage_scores_case_unmarked,sim_lineage_scores_control)
COSICC_DA_lineage(sce_DA_lineage_case,sce_DA_lineage_control,lineage_scores,alpha=0.1)
```




```
##     lineage   p_values odds_ratio             sig
## 1 lineage_1 0.03015308  0.2872039        depleted
## 3 lineage_3 0.79963049  0.5667589 not significant
## 2 lineage_2 0.93764654  1.5958865 not significant
## 4 lineage_4 0.93764654  0.7213453 not significant
## 5 lineage_5 0.93764654  0.6655057 not significant
```

Session Information

```
sessionInfo()
```

```
## R version 4.4.2 (2024-10-31)
## Platform: aarch64-apple-darwin20
## Running under: macOS Sonoma 14.7.1
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRlapack.dylib;  LAPACK v
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## time zone: Europe/London
```

```
## tzcode source: internal
##
## attached base packages:
## [1] stats4    stats     graphics  grDevices datasets  utils     methods
## [8] base
##
## other attached packages:
##  [1] COSICC_0.1.0                destiny_3.19.0
##  [3] dplyr_1.1.4                 scater_1.34.0
##  [5] ggplot2_3.5.1               scran_1.33.2
##  [7] scuttle_1.15.5              splatter_1.29.1
##  [9] SingleCellExperiment_1.28.0 SummarizedExperiment_1.36.0
## [11] Biobase_2.66.0              GenomicRanges_1.58.0
## [13] GenomeInfoDb_1.42.0         IRanges_2.40.0
## [15] S4Vectors_0.44.0            BiocGenerics_0.52.0
## [17] MatrixGenerics_1.18.0       matrixStats_1.4.1
##
## loaded via a namespace (and not attached):
##   [1] vcd_1.4-13              rstudioapi_0.17.1       jsonlite_1.8.9
##   [4] magrittr_2.0.3          ggbeeswarm_0.7.2        farver_2.1.2
##   [7] rmarkdown_2.29          zlibbioc_1.52.0         vctrs_0.6.5
##  [10] tinytex_0.54            htmltools_0.5.8.1       S4Arrays_1.6.0
##  [13] curl_6.0.0              BiocNeighbors_2.0.0     SparseArray_1.6.0
##  [16] Formula_1.2-5           TTR_0.24.4              zoo_1.8-12
##  [19] igraph_2.1.1            lifecycle_1.0.4         pkgconfig_2.0.3
##  [22] rsvd_1.0.5              Matrix_1.7-1            R6_2.5.1
##  [25] fastmap_1.2.0           GenomeInfoDbData_1.2.13 digest_0.6.37
##  [28] pcaMethods_1.94.0       colorspace_2.1-1        dqrng_0.4.1
##  [31] RSpectra_0.16-2         irlba_2.3.5.1           beachmat_2.22.0
##  [34] labeling_0.4.3          fansi_1.0.6             httr_1.4.7
##  [37] abind_1.4-8             compiler_4.4.2          proxy_0.4-27
##  [40] withr_3.0.2             backports_1.5.0         BiocParallel_1.40.0
##  [43] carData_3.0-5           viridis_0.6.5           hexbin_1.28.4
##  [46] knn.covertree_1.0       MASS_7.3-61             DelayedArray_0.32.0
##  [49] scatterplot3d_0.3-44    bluster_1.16.0          tools_4.4.2
##  [52] vipor_0.4.7             lmtest_0.9-40           ranger_0.16.0
##  [55] ggplot.multistats_1.0.1 beeswarm_0.4.0          nnet_7.3-19
##  [58] glue_1.8.0              grid_4.4.2              checkmate_2.3.2
##  [61] cluster_2.1.6           generics_0.1.3          gtable_0.3.6
##  [64] class_7.3-22            tidyr_1.3.1             data.table_1.16.2
##  [67] BiocSingular_1.22.0     ScaledMatrix_1.14.0     metapod_1.14.0
##  [70] sp_2.1-4                car_3.1-3               utf8_1.2.4
##  [73] XVector_0.46.0          RcppAnnoy_0.0.22        ggrepel_0.9.6
##  [76] pillar_1.9.0            stringr_1.5.1           RcppHNSW_0.6.0
##  [79] VIM_6.2.2               limma_3.62.1            robustbase_0.99-4-1
##  [82] lattice_0.22-6          renv_1.0.11             smoother_1.3
##  [85] tidyselect_1.2.1        locfit_1.5-9.10         knitr_1.49
##  [88] gridExtra_2.3           edgeR_4.3.21            xfun_0.49
##  [91] statmod_1.5.0           DEoptimR_1.1-3          stringi_1.8.4
##  [94] UCSC.utils_1.2.0        yaml_2.3.10             boot_1.3-31
##  [97] evaluate_1.0.1          codetools_0.2-20        laeken_0.5.3
## [100] RcppEigen_0.3.4.0.2     tibble_3.2.1            cli_3.6.3
## [103] uwot_0.2.2              munsell_0.5.1           Rcpp_1.0.13-1
## [106] parallel_4.4.2          viridisLite_0.4.2       ggthemes_5.1.0
```

```
## [109] scales_1.3.0           xts_0.14.1             e1071_1.7-16
## [112] purrr_1.0.2            crayon_1.5.3           rlang_1.1.4
## [115] cowplot_1.1.3
```

References

Guibentif, Carolina, Jonathan A. Griffiths, Ivan Imaz-Rosshandler, Shila Ghazanfar, Jennifer Nichols, Valerie Wilson, Berthold Göttgens, and John C. Marioni. 2021. "Diverse Routes Toward Early Somites in the Mouse Embryo." *Developmental Cell* 56 (1): 141–153.e6.

"Optimal-Transport Analysis of Single-Cell Gene Expression Identifies Developmental Trajectories in Reprogramming." 2019. *Cell* 176 (4): 928–943.e22.

Pijuan-Sala, Blanca, Jonathan A. Griffiths, Carolina Guibentif, Tom W. Hiscock, Wajid Jawaid, Fernando J. Calero-Nieto, Carla Mulas, et al. 2019. "A Single-Cell Molecular Map of Mouse Gastrulation and Early Organogenesis." *Nature* 566 (7745): 490–95.

Strauss, Magdalena E, Mai-Linh Nu Ton, Samantha Mason, Jaana Bagri, Luke TG Harland, Ivan Imaz-Rosshandler, Nicola K Wilson, et al. 2023. "Bespoke Single Cell Molecular and Tissue-Scale Analysis Reveals Mechanisms Underpinning Development and Disease in Complex Developing Cell Populations." *bioRxiv*.

Zappia, Luke, Belinda Phipson, and Alicia Oshlack. 2017. "Splatter: Simulation of Single-Cell RNA Sequencing Data." *Genome Biology*.