User manual for analaysis

Louisa Hadj Abed

4/06/2022

Overview

In this vignette, we provide a walkthrough analysis of a lentiviral barcoding study of dendritic cells as described in: HadjAbed et al. 2022. Specifically, we wish to assess if a single MPP4 can produce both the cDC1 and the cDC2 subsets in the lung or whether a single MPP4 is fate-restricted to only produce cDC1 or cDCs.

To address this research question, we used a lentiviral barcoding approach focusing on the differentiation of (Lin-, Sca-1+, cKit+, Flt3+) MPP4s towards lung-resident cDCs. Murine MPP4s were purified from the bone marrow of donor mice by fluorescence activated cell sorting and infected with the LG2.2 lentiviral barcoding library. Labelled cells where then injected I.V into 3 irradiated recipient mice. 14 days later, lungs were isolated from the mice, and barcoded cDC1s, and cDC2s were purified by FACS. Samples were then processed for barcode detection in genomic DNA by deep sequencing.

Now, we've seen that the QC is ok, we visualise in this script key analysis steps of the data comparing cDC1 and cDC2 dendritic cells subtypes in three mice.

Load libraries

```
library(ggplot2)
library(devtools)
#load packages
devtools::install_github("TeamPerie/CellDestiny", quiet = TRUE)
library(CellDestiny)
```

Load data and give individuals variable name

Like for the application format, the first step of the analysis part of the package is to load count and metadata matrcies and give the name of the variable describing your *individuals*. It corresponds to one of your metadata column name.

```
# set working directory
setwd(getwd())
# import files
count_matrix <- read.csv("../../testData/LentiviralBarcodingData/Analysis_data/Analysis_matrix_Mouse_Lu
metadata <- read.csv("../../testData/LentiviralBarcodingData/Analysis_data/Analysis_matrix_Mouse_Lung_c
metadata
```

```
##   type mouse
## 1 cDC1     2
## 2 cDC2     4
## 3          5
```

Here, it is "mouse".

```
# Common parameters
indiv_var="mouse"
indiv_val=c("2", "4", "5") # you can change it according if you
                           # want to plot all individuals or not
```

All along the script, there are two *types of functions*:

1) the ones that create matrices for which the name begin with ***"Make"***
2) the ones that plot graphs for which the name begin with ***"Plot"***

All plotting functions need as input matrix their corresponding "Make" function.

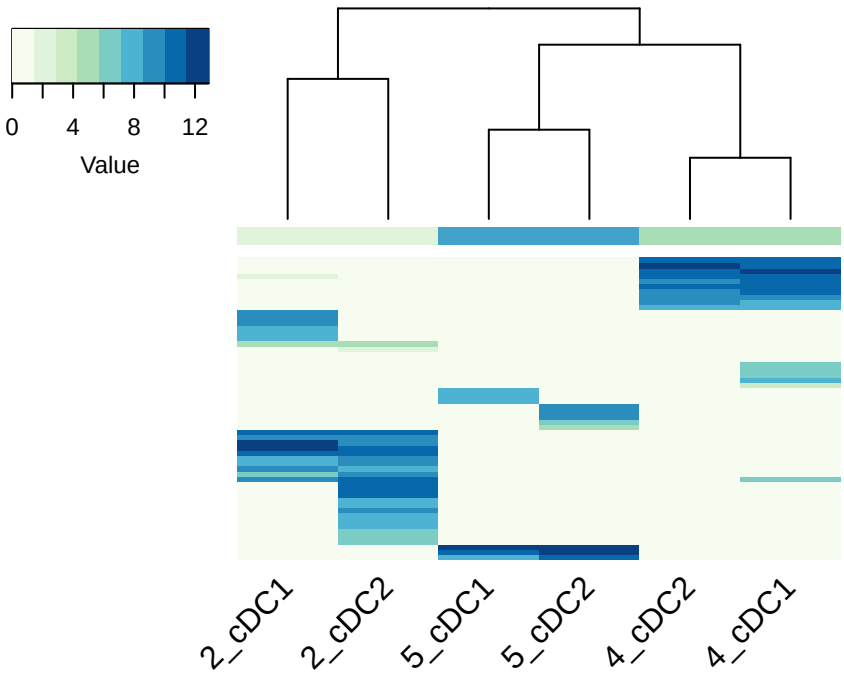
Sample similarities

Similarities between samples can be visualized using a heatmap together with hierarchical clustering. In the package, the function MakeHeatmapMatrix() prepare the data to plot heatmaps using the PlotHeatmap() function. Several options are available for the distance and algorithm used for clustering. Additionally, a correlogram can be plotted (MakeHeatmapMatrix() and PlotCorrelogram() in the package) that illustrates the correlation of barcode abundances between all pairs of variables.

Heatmap

```
## heatmap functions parameters
list_var = c("type")
list_val = metadata$type
distance<-"euclidean"
clusteringMeth<-"complete"
dendro<-"yes"
barcodes<-"no"
nclust<-3
pool=FALSE # do you want to pool individuals ?
```

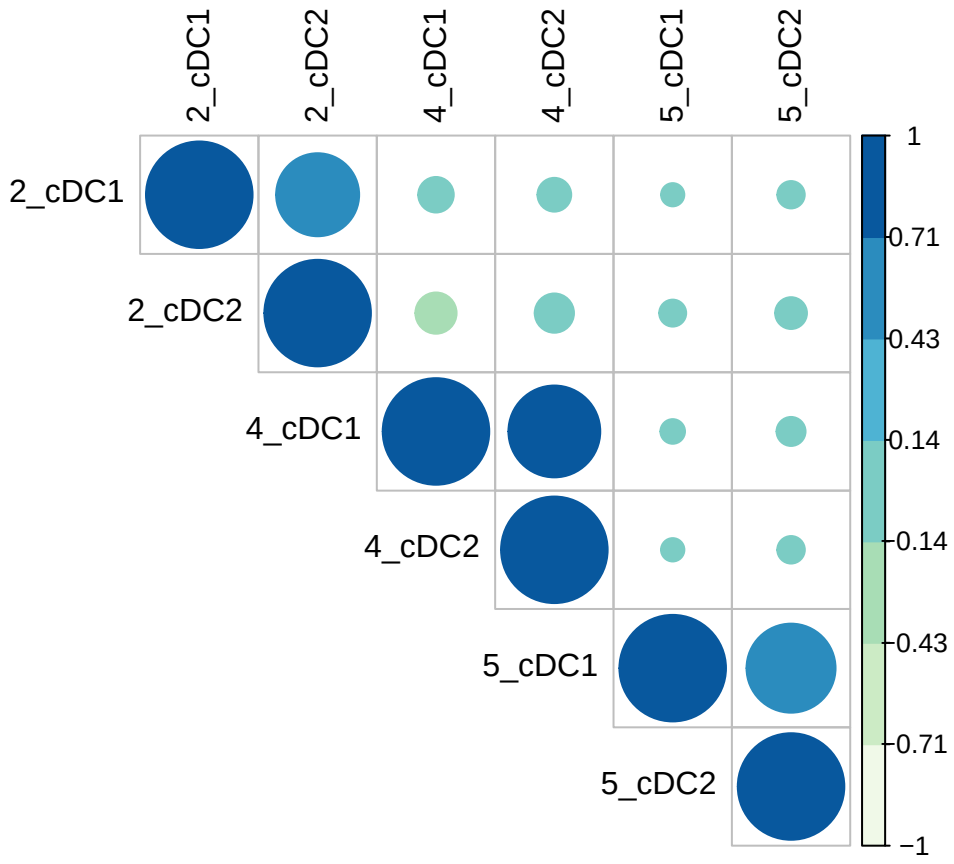
```
heat_mat<-MakeHeatmapMatrix(matrix = count_matrix, metadata = metadata,
                            indivVar = indiv_var, indivVal = indiv_val,
                            listVar = list_var,  listVal = list_val,
                            poolIndiv = pool)
p<-PlotHeatmap(heat_mat, distance = distance, clustering = clusteringMeth,
               showDendro = dendro, showBarcodes = barcodes,
               nClusters = nclust, columnTextSize = 1.5)
```



Correlogram

Spearman correleations are outputed.

```
p<-PlotCorrelogram(heat_mat)
```



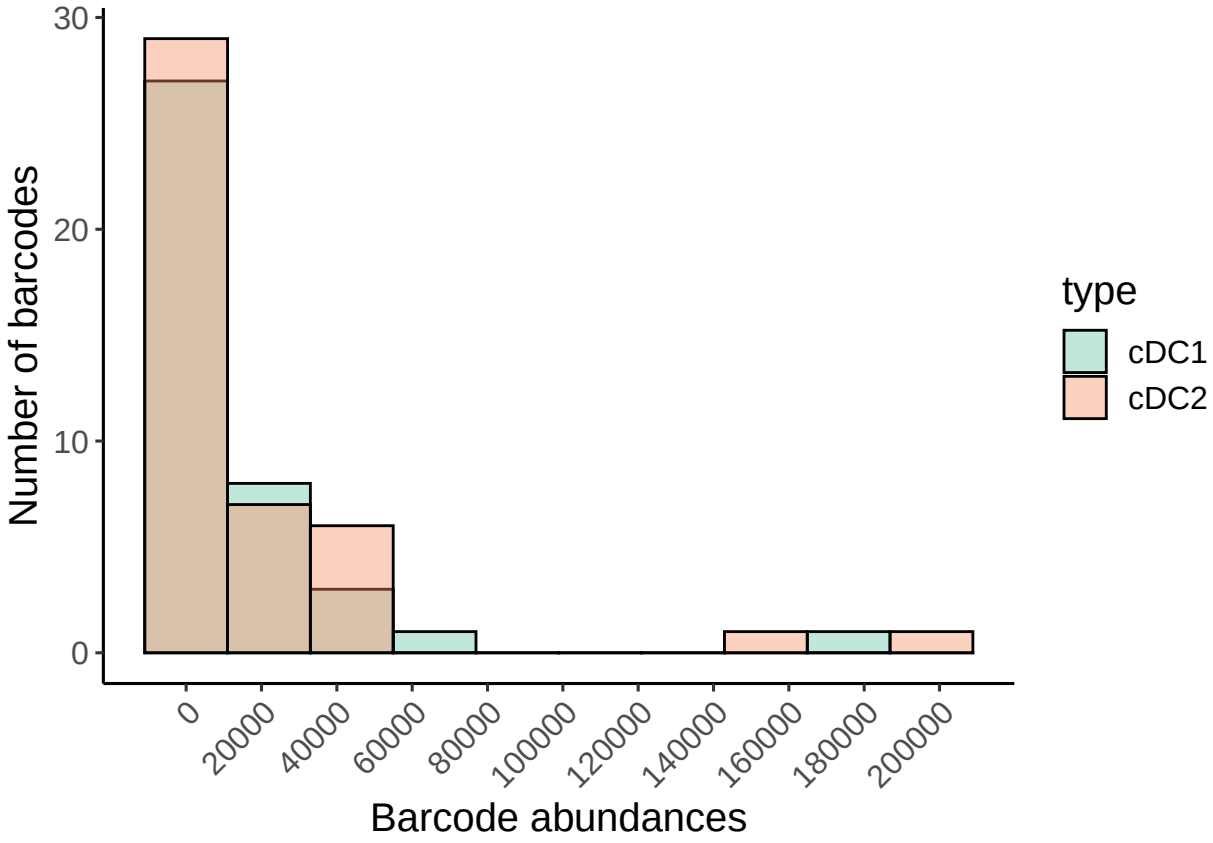
Clone sizes

CellDestiny offers two types of clone size visualizations. The first one is a cumulative diagram (use MakeCumulativeDiagramMatrix() and PlotCumulativeDiagram() in the package). If the cumulative graph has a concave shape, it means that a cell population is dominated by a small number of large clones. On the contrary, if the shape is linear, the sample is composed of a number of clones which contribute equally to the cellularity of the population. The second type of graph is a frequency distribution plot (MakeBarcodeFrequenciesMatrix() and PlotBarcodeFrequencies ()) in the package) where the user can choose between histogram or density curve -based representations of the data.

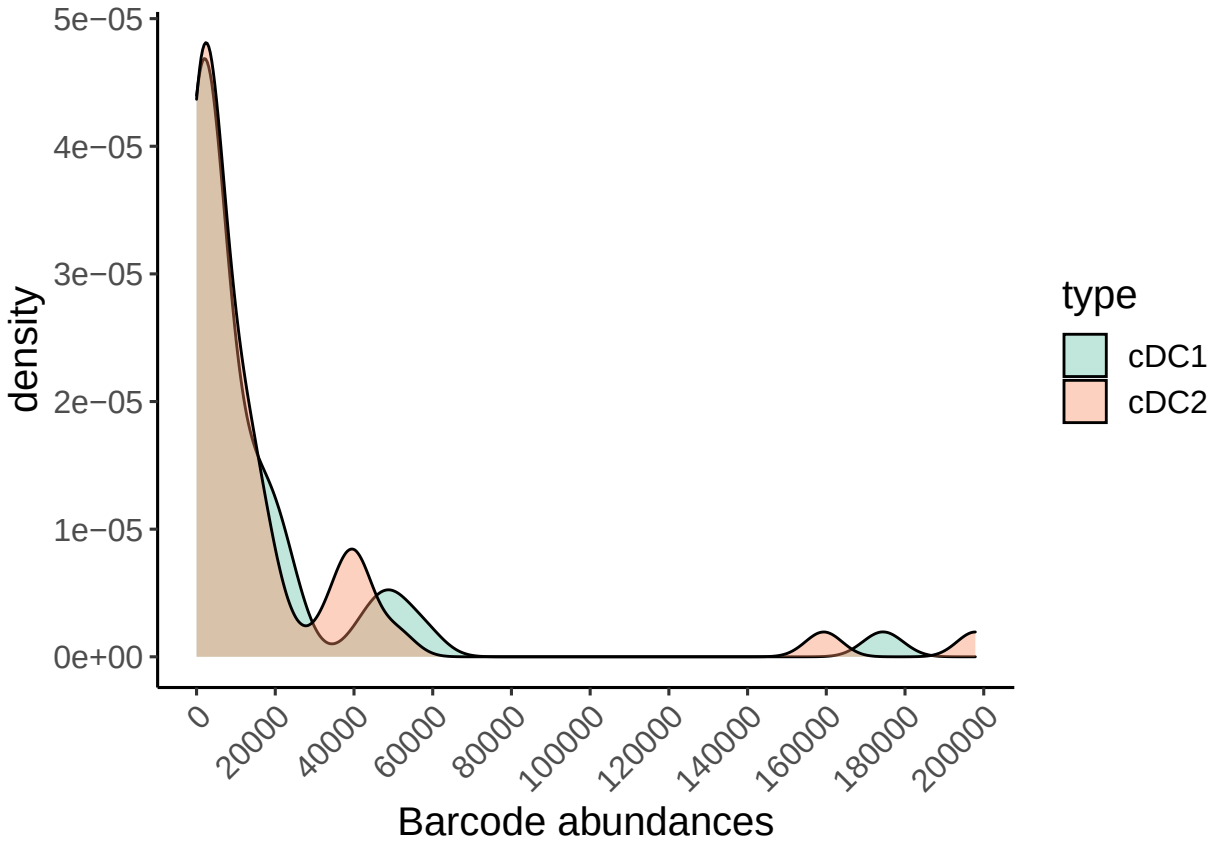
Plot clone size distributions using an histogram

Change y parameter depending which representation you prefer.

```
# fill wanted parameters
list_var = c("type")
list_val = metadata$type
indiv_val=c("2", "4", "5")
colorFreq="type"

freq_mat<-MakeBarcodeFrequenciesMatrix(count_matrix, metadata,
                                       indiv_var, indiv_val,
                                       list_var, list_val)
PlotBarcodeFrequencies(freq_mat, colorFreq, y = "histogram", nbins = 10)
```



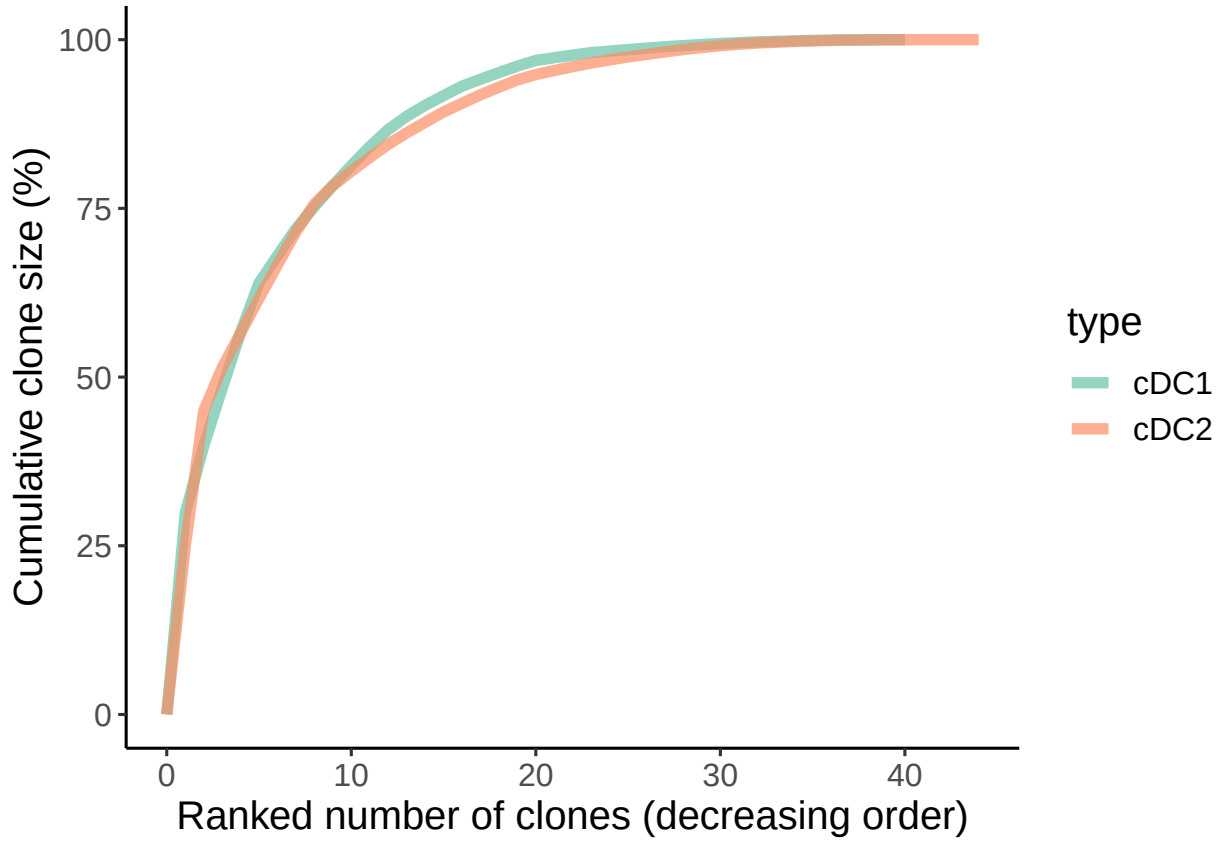

```
PlotBarcodeFrequencies(freq_mat, colorFreq, y="density", nbins = 10)
```



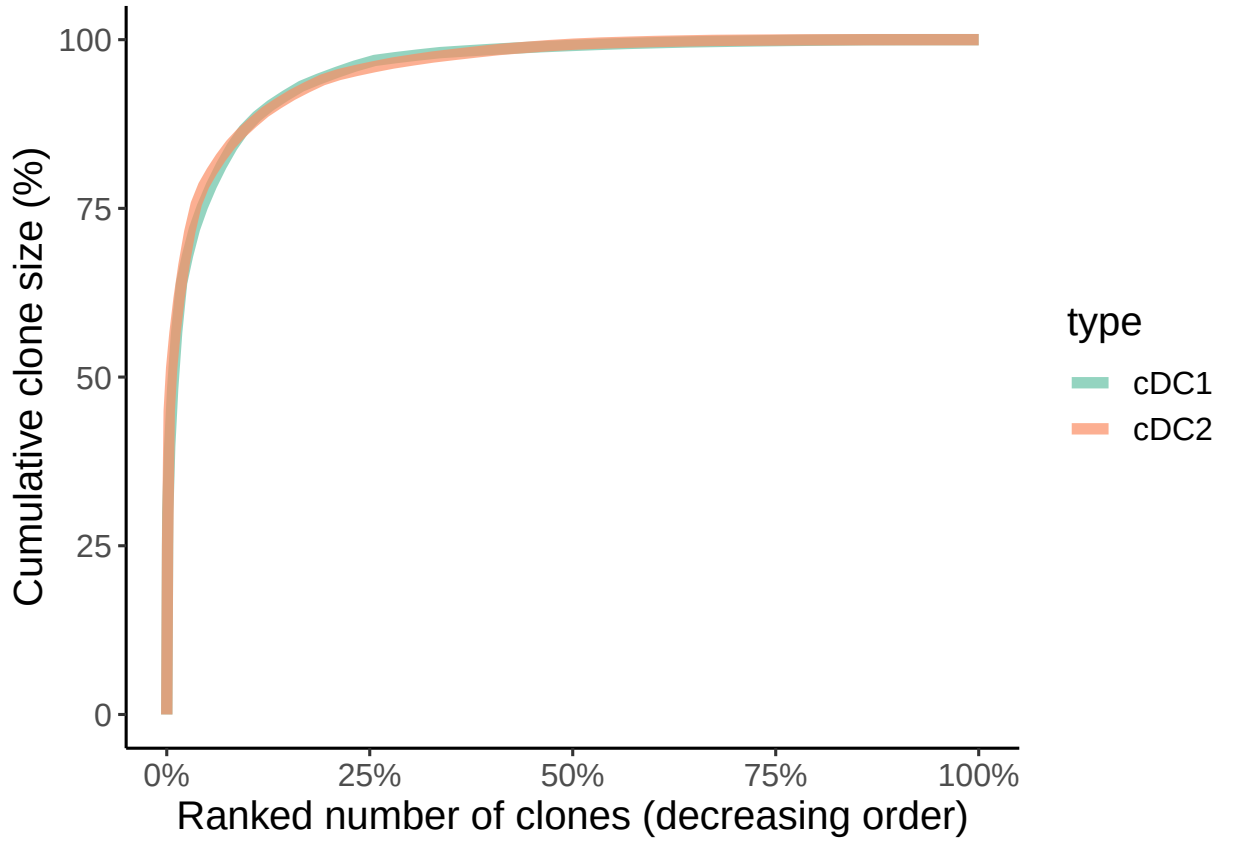
Plot clone-size distributions using a cumulative frequency diagram.

Change xProp parameter depending which x axis scale you prefer.

```
list_var = c("type")
list_val = metadata$type
colorCum="type"

cum_mat_notxProp<-MakeCumulativeDiagramMatrix(count_matrix, metadata,
                                              indiv_var, indiv_val,
                                              list_var, list_val, colorCum)
PlotCumulativeDiagram(cum_mat_notxProp,indiv_var, colorCum)
```




```
# add xProp = "yes"
cum_mat_xProp<-MakeCumulativeDiagramMatrix(count_matrix, metadata,
                                           indiv_var, indiv_val,
                                           list_var, list_val,
                                           colorCum, xProp = "yes")
PlotCumulativeDiagram(cum_mat_xProp,indiv_var, colorCum, xProp = "yes")
```

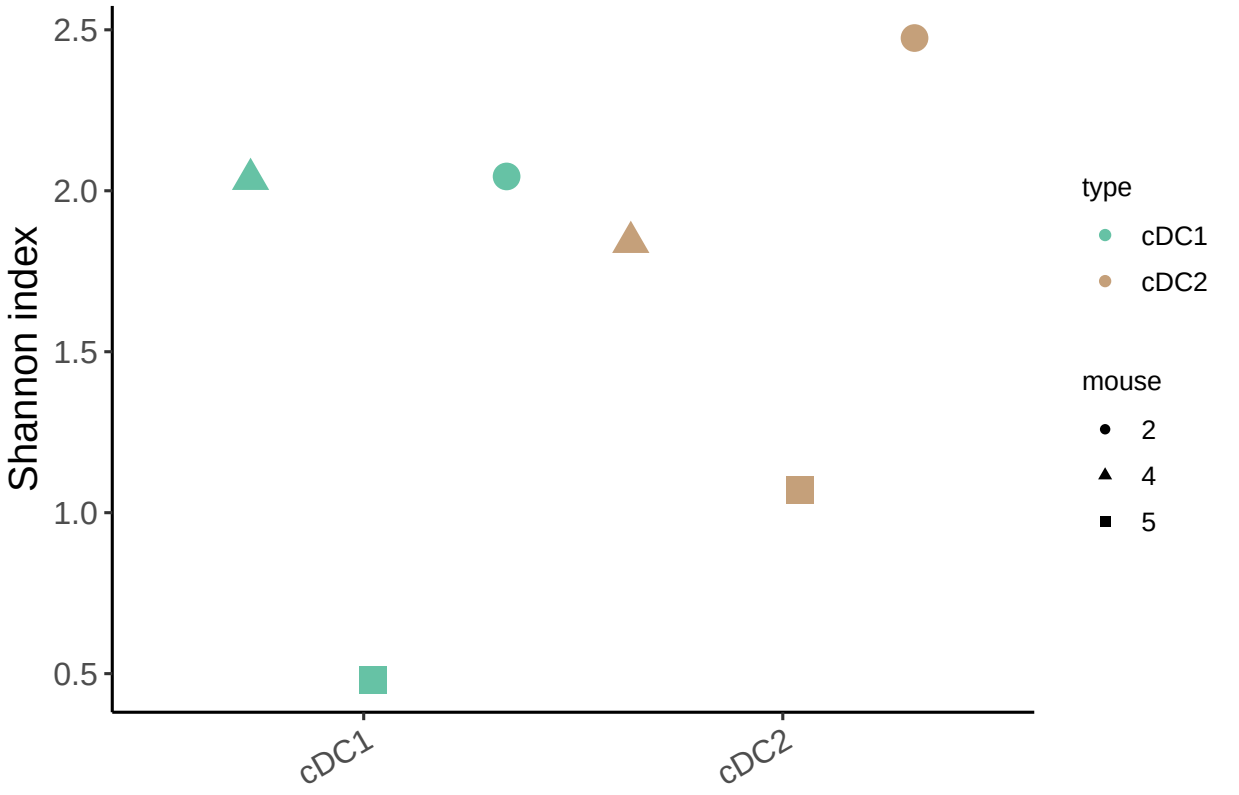


Diversity

Comparing sample diversities is a common step in lineage tracing analysis. Diversity is computed using the vegan R package. In the CellDestiny package, the function CalculDiversity() calculate diversity using the number of unique clones, the Shannon index or the Simpson index. For vizualizing clonal diversity, PlotDiversity() computes a boxplot of barcode diversities.

Quantify clonal diversity between cDC1 and cDC2 using the Shannon Index.

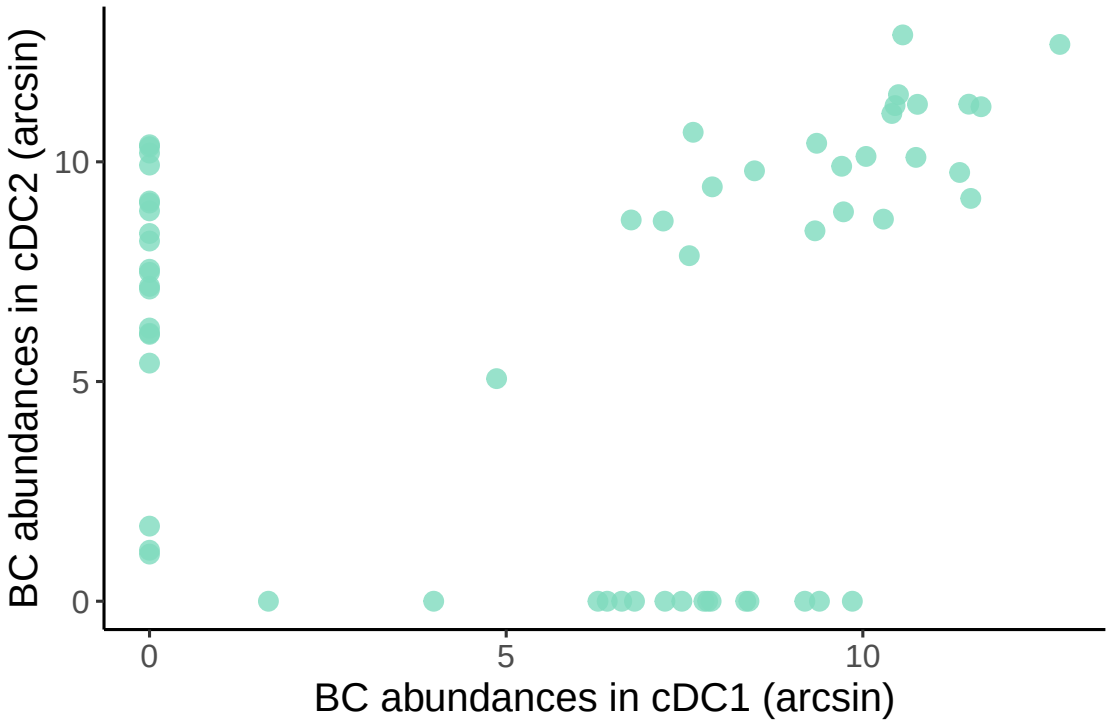
```
boxplotColor_var=""
diversityVar<-"Shannon index"

div_mat<-CalculDiversity(count_matrix, metadata,
                         indiv_var, indiv_val,
                         list_var, list_val ,
                         colorVar=boxplotColor_var, diversity=diversityVar)
PlotDiversity(div_mat,diversityVar, list_var, indiv_var,
              colorVar=boxplotColor_var, dots = "no")
```



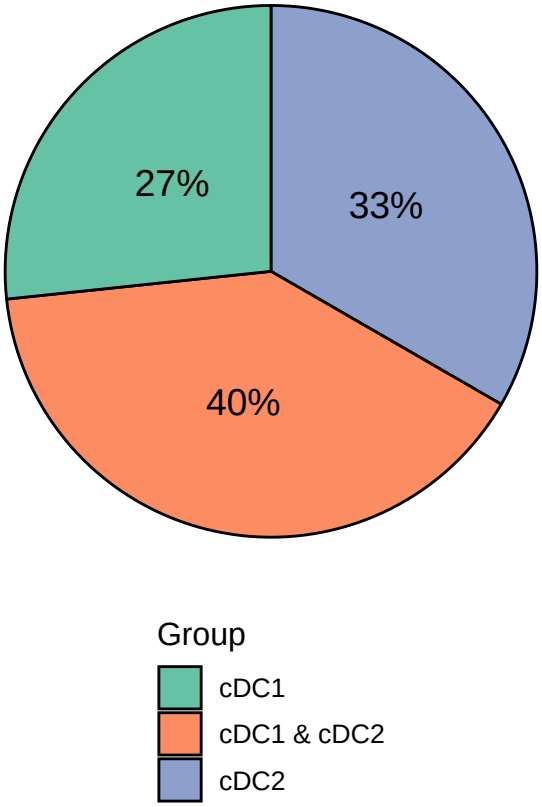
Now lets see if barcode abundances are similar between our cDC1 and cDC2 samples using scatter plot and pie chart visualisations.

```
## parameters for both dotplot and correlogram are similar
x_var<-c("type")
x_val<-c("cDC1")
y_var<-c("type")
y_val<-c("cDC2")
color="" # we dont want color
```

```
dot_mat<-MakeDotPlotMatrix(count_matrix, metadata,
                           indiv_var, indiv_val,
                           x_var, x_val,
                           y_var, y_val,
                           color)
PlotDotplot(dot_mat, indiv_var, color, textSize = 15)
```




```
pie_mat<-MakePieChartMatrix(dot_mat, indiv_var, color)
PlotPieChart(pie_mat, textSize = 5)
```



Categorisation

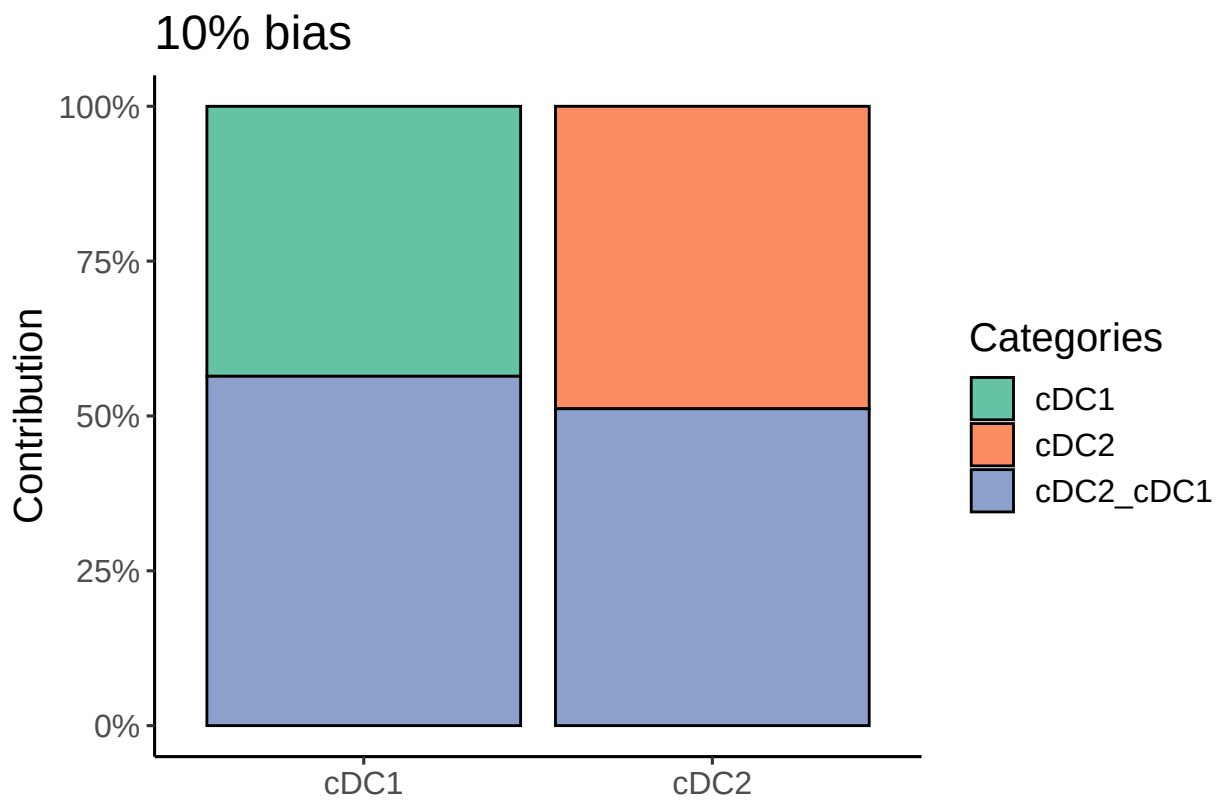
To classify barcodes by their lineage bias, CellDestiny uses a threshold based classifier lineage described (Naik et al. 2013b). In summary, an additional normalization step per barcode is applied in each individual, thereby enabling categorization of each barcode into classes of biased output towards the analyzed cell types. Barcodes are assigned a bias based on whether the % read abundance exceeds a threshold value. If one barcode contributes to a given lineage above the designated threshold then this barcode is assigned to be biased towards that lineage. Barcodes for which the % read abundance exceeds a threshold value across multiple lineages are classified as multi-outcome. In the CellDestiny app, the threshold used for categorization can be tuned manually. In the CellDestiny package, MakeCategoryMatrices() prepare the data to input the PlotCategories() and the complementary PlotCategoryCounts() functions that output the number of barcodes per category and the summed contribution of all the barcodes in this category. If several individuals are present, PlotCategoryCounts() averages the summed contribution over individuals.

```
# Get category variable and values
catVar<-"type"
catVal<-metadata$type
```

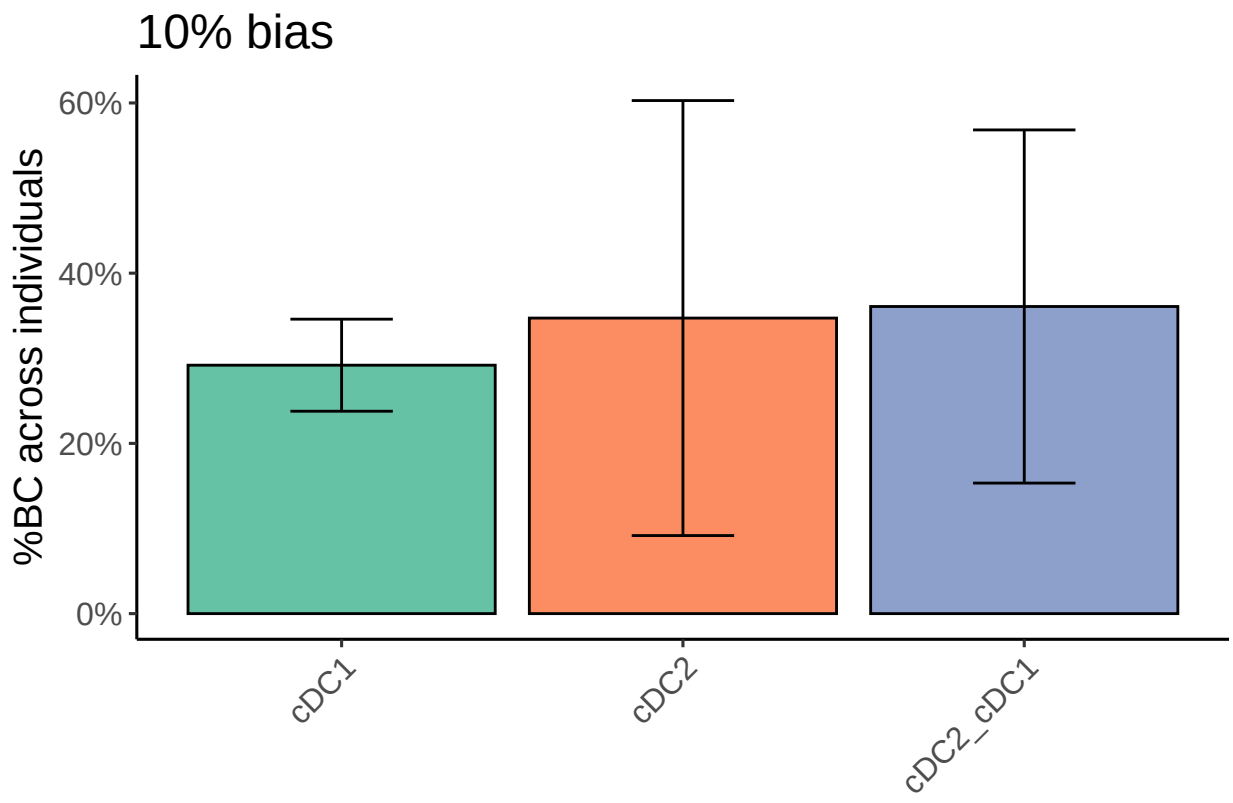
10% bias

```
slider<-10 # change the bias threshold thanks to this parameter

## the first matrix of the list outputed by MakeCategoryMatrices() is
## needed for PlotCategories() input matrix
```

```
count_per_type<-MakeCategoryMatrices(count_matrix, metadata,
                                     indiv_var, indiv_val,
                                     catVar, catVal, slider)[[1]]
PlotCategories(count_per_type, slider)
```



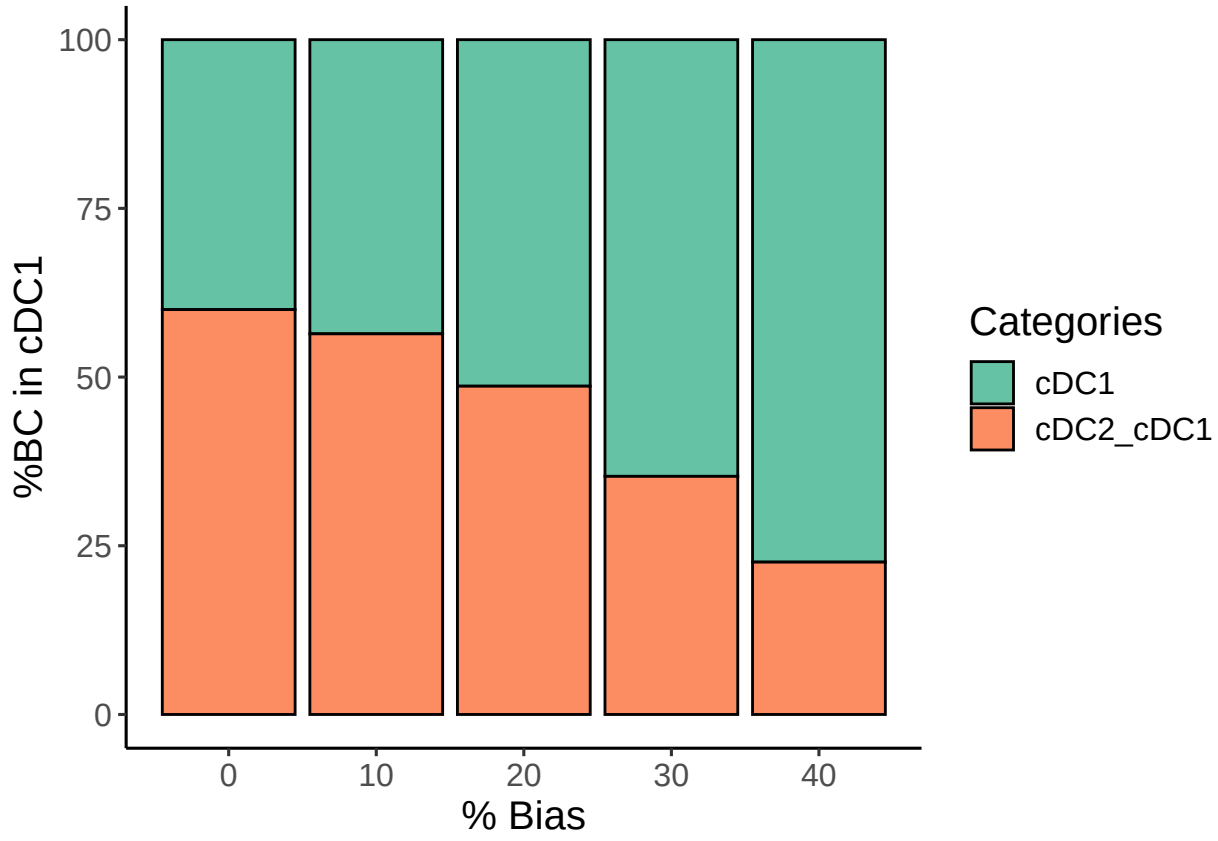

```
## the second matrix of the list outputed by MakeCategoryMatrices() is
## needed for PlotCategoryCounts() input matrix
count_per_cat<-MakeCategoryMatrices(count_matrix, metadata,
                                    indiv_var, indiv_val,
                                    catVar, catVal, slider)[[2]]
PlotCategoryCounts(count_per_cat, slider)
```



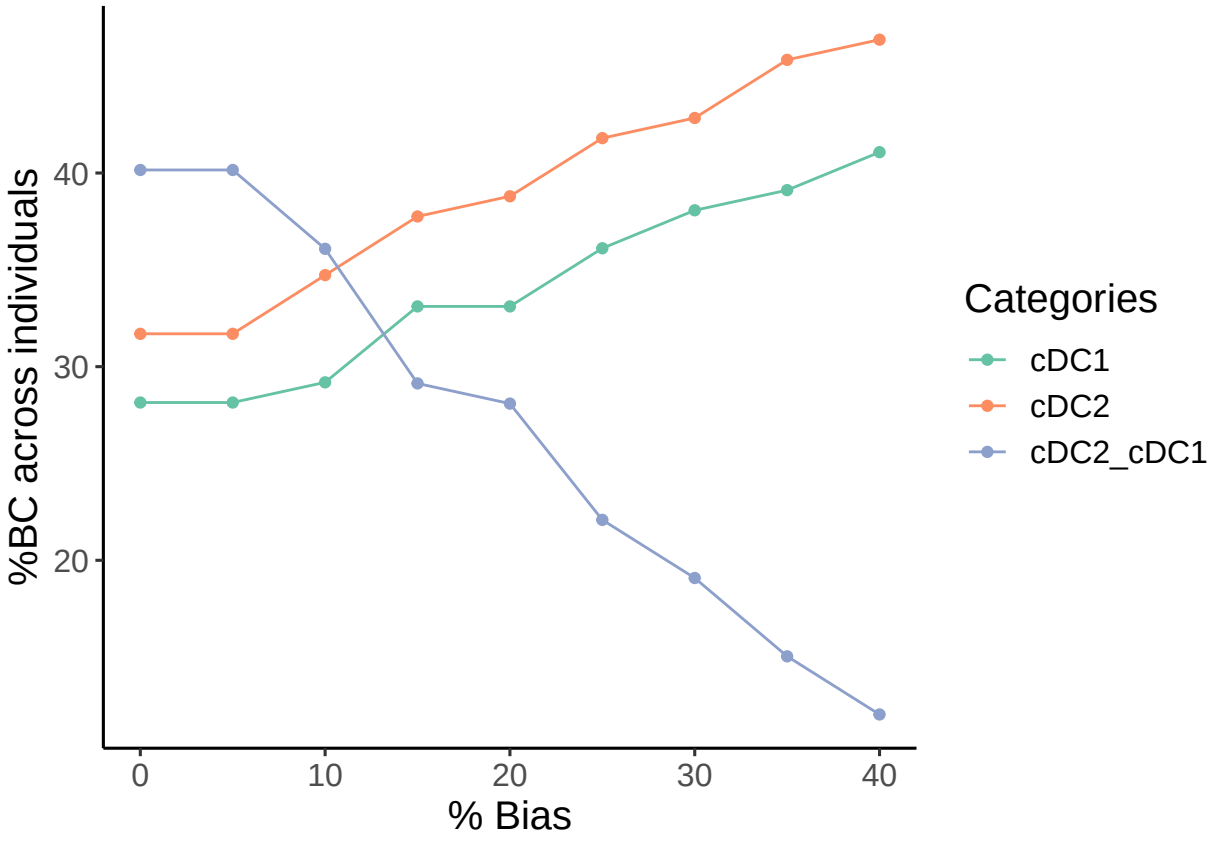
Bias threshold analysis

CellDestiny offers also a way to get an overview of the categorisation accross bias threshold values.

```
biasType<-MakeBiasPerTypeMatrix(count_matrix, metadata,
                                indiv_var, indiv_val,
                                catVar, catVal)
PlotBiasPerType(biasType, y = "cDC1")
```




```
bias<-MakeBiasPerCatMatrix(count_matrix, metadata,
                           indiv_var, indiv_val,
                           catVar, catVal)
PlotBiasPerCat(bias)
```


%BC across individuals
40
30
20
0
10
20
30
40
% Bias
Categories
cDC1
cDC2
cDC2_cDC1