Vignette for running the scDNAseq pipeline to measure rCNAs in two paired cell lines

Michael P Schneider

2024-02-16

Contents

1 Introduction

Here, I will run you through the analysis of a paired sample of shallow whole-genome sequencing single-cell DNA sequencing data (PEO1/PEO4). The two cell lines differ in their HRD activity. More data and in depth analysis can be found in the **scUnique** manuscript.

Please use the snakemake workflow provided in the GitHub repository for your analysis. This workflow automates the process and ensures reproducibility.

You can run the analysis with the snakemake workflow provided (recommended). Alternatively, you can try to use the public docker images to directly run parts of the analysis. We cannot support running the software outside the containers because of the range of software dependencies.

In order to run the pipeline, you will require a working snakemake (tested with version 8) and singularity installation. In order to access the docker images, you will need the aws client. Input bam files should be coordinate-sorted. In addition, the scAbsolute and scUnique git directories should be cloned to your home directory (by default "~/"). You can also modify the directory via the BASEDIR variable and via the singularity mount options when executing the containers. We also highly recommend using our naming scheme for the reads, described in the basic vignette. Every name consists of four parts, separated by an underscore.

```
[unique sample identifier]_[library identifier]_[running cell id]_[cell tag/all other information].bam
UID-10X-Fibroblast-cell_SLX-00000_000001_AAACCTGAGCAGCCTC-1.bam
```

2 Single-cell ploidy and replication analysis, and single-cell copy number calling

The data (subsampled to 100 cells each) are available to download at this link . Download the files to the folder data/aligned/PEO1 and PEO4, respectively. Normally, you would need to create a sample file, but these already exist in the config folder (see config/sample_PEO1.tsv and config/sample_PEO4.tsv). Running the first part of the pipeline produces a per sample output file, e.g. results/500/PEO1_500.rds for a bin size of 500kB. It also creates a folder results/500/PEO1/ with one rds file per cell. It is important to keep these per-cell files in places in order to run the second part of the pipeline on them.

3 Quality control

We highly recommend manually inspecting the results of the scAbsolute run. While it is possible to run the qc script automatically with the pre-defined cutoffs, it is necessary to check on a per-sample basis if the cutoffs make sense given your data might vary. It is also worth noting, that the QC step becomes more reliable with more cells per sample, as we use a density based approach.

Please have a look at the qc-script.R files under workflow/scripts. I will briefly go over the main checks in the script here to demonstrate what things to look for.

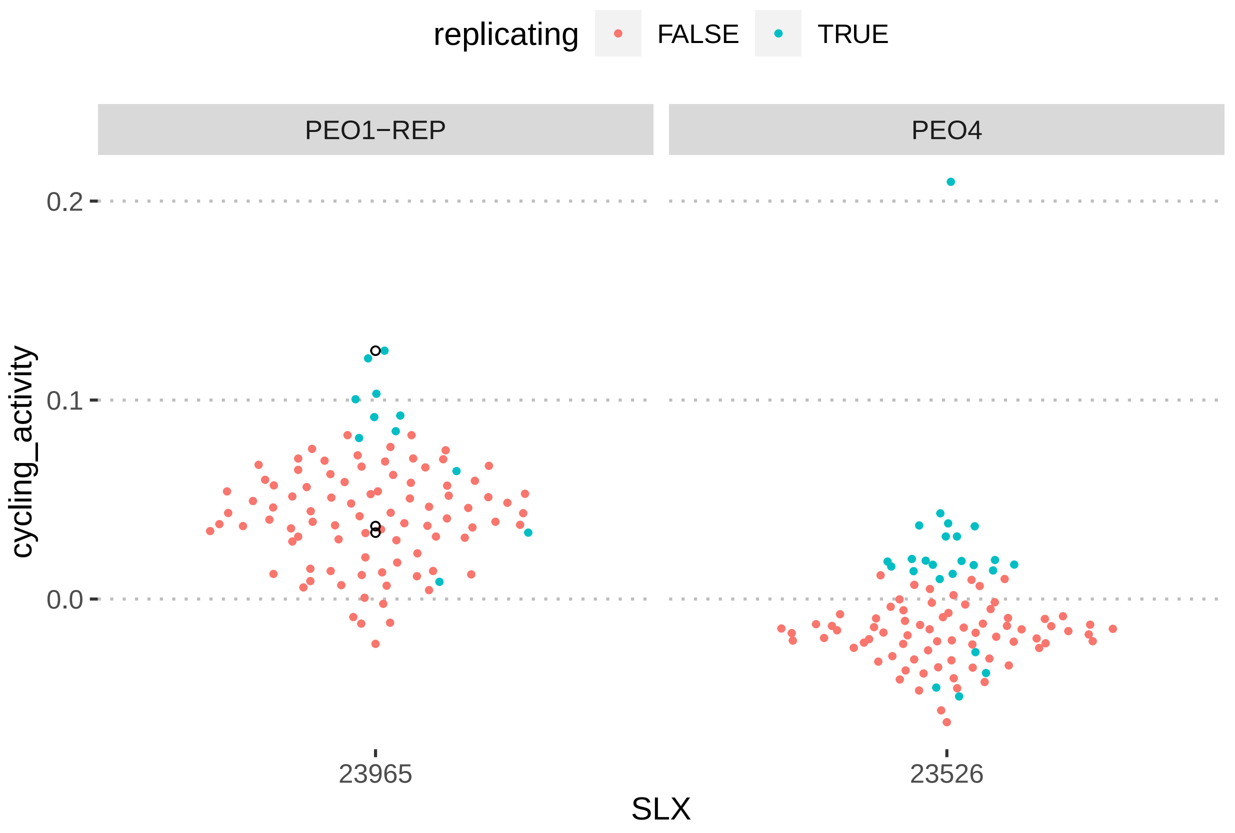
3.1 Loading libraries and data for interactive analysis

We recommend using the copynumber conda environment provided in config/envs.yml. To visualize results and conduct the quality controls, we first need to load dependencies.

```
  WORKFLOW_PATH="~/scDNAseq-workflow/"
  BASEDIR="~/"
  source(paste0(WORKFLOW_PATH, "envs/dependencies.R"))
#> [1] "BASEDIR is set to: ~/"

  df = dplyr::bind_rows(lapply(
    c("vignettes/data/PEO1_500.rds", "vignettes/data/PEO4_500.rds"), function(x){
      return(Biobase::pData(readRDS(paste0(WORKFLOW_PATH, x))))
  })) %>%
    tidyr::separate(name, sep="_", into=c("UID", "SLX", "cellid", "celltag"), remove=FALSE) %>%
    dplyr::mutate(UID = gsub("UID-FML-", "", UID), SLX=gsub("SLX-", "", SLX))
```

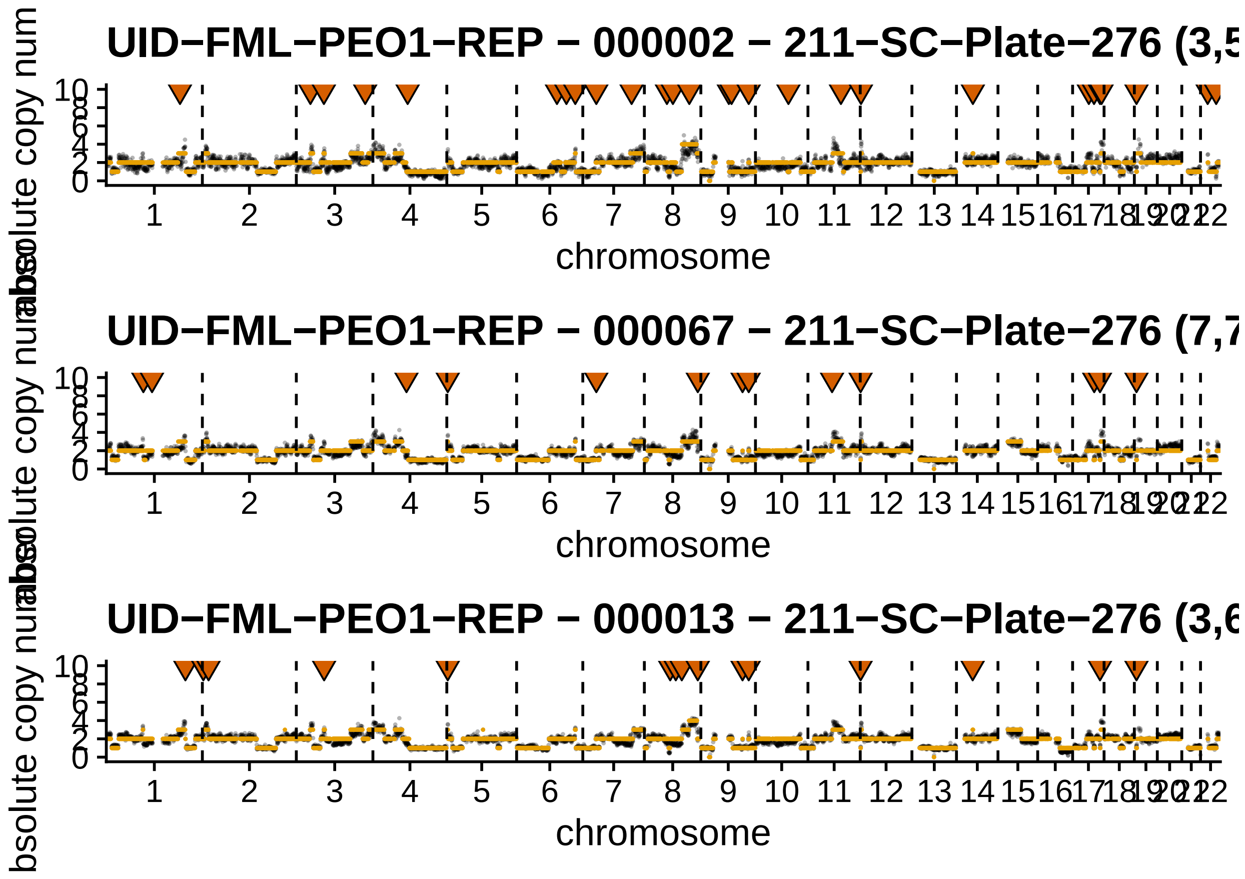
3.2 Predicting replicating cells

```
df = predict_replicating(df, cutoff_value=1.5, iqr_value = 1.5)
fig_cellcycle = ggplot(data = df) + geom_quasirandom(aes(x=SLX, y=cycling_activity,
  color=replicating, name=name), size=0.8, alpha=1.0) +
  geom_point(data = df %>% dplyr::filter(cellid %in% c("000002", "000013", "000067"), SLX=="23965"), aes
  facet_wrap(~UID, scales = "free_x") + theme_pubclean()
fig_cellcycle
```



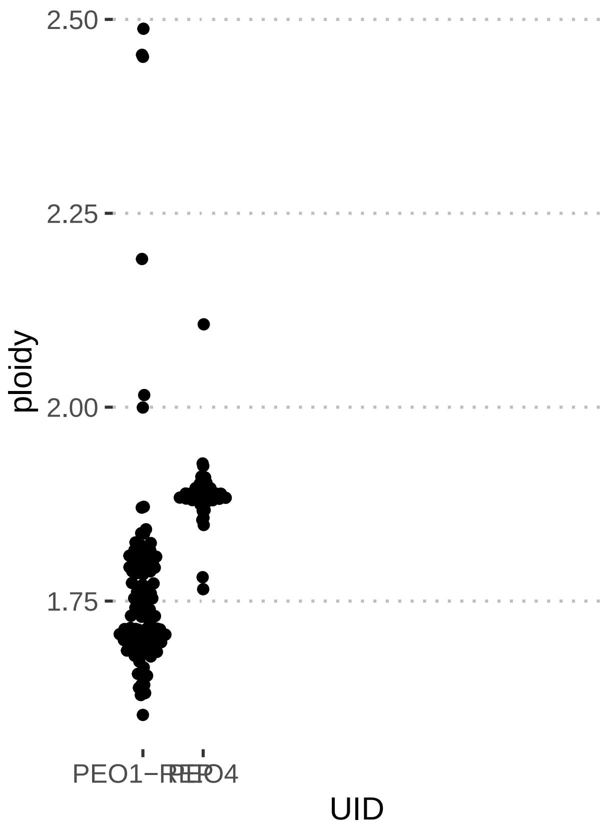
The choice of cutoff depends on whether the cells have previously been FACS sorted and how stringent you want to be. Here, we can be relatively relaxed, as the cells have already been previously sorted using Gemini staining.

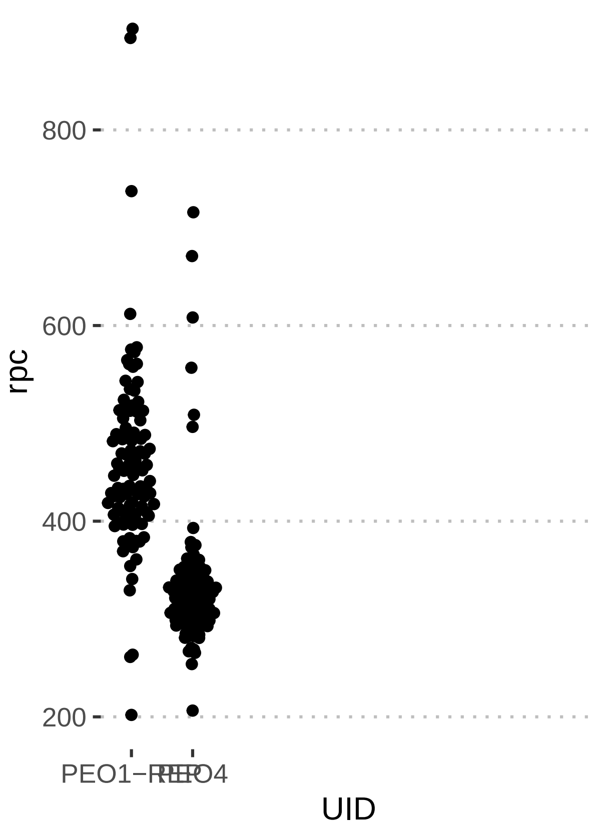
```
CN = readRDS(paste0(WORKFLOW_PATH, "vignettes/data/PEO1_500.rds"))
pc1 = plotCopynumber(CN[, grepl("000002", colnames(CN))])
#> [1] "Unique items:"
#> unique_index
#> FALSE  TRUE
#>  4338   388
#> [1] 27
pc2 = plotCopynumber(CN[, grepl("000067", colnames(CN))])
#> [1] "Unique items:"
#> unique_index
#> FALSE  TRUE
#>  4592   134
#> [1] 13
pc3 = plotCopynumber(CN[, grepl("000013", colnames(CN))])
#> [1] "Unique items:"
#> unique_index
#> FALSE  TRUE
#>  4590   136
#> [1] 15

ggpubr::ggarrange(pc1, pc2, pc3, nrow=3)
```



3.3 Ploidy values double check and rpc value

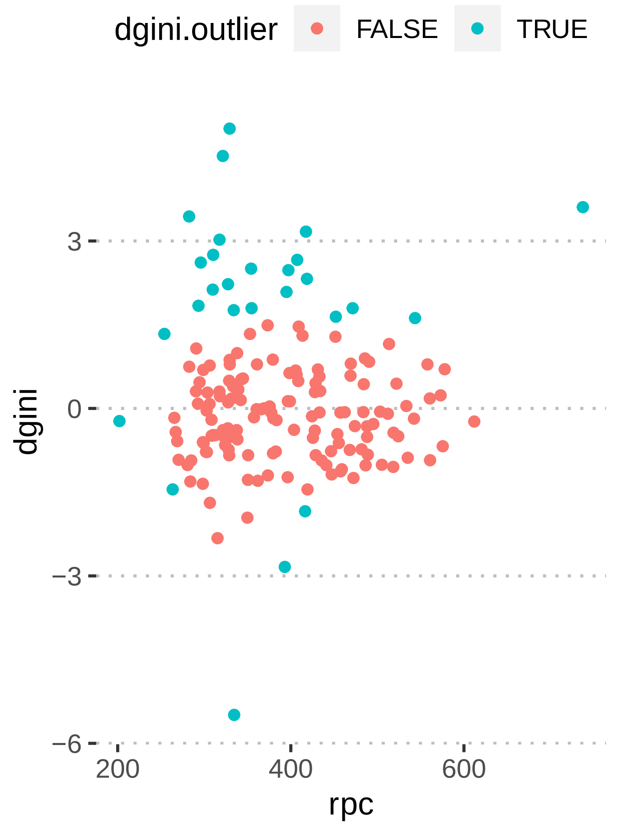
```
p1 = ggplot(data = df) + geom_quasirandom(aes(x=UID, y=ploidy)) + coord_cartesian(xlim=c(1.1, 8.0)) + tl

p2 = ggplot(data = df) + geom_quasirandom(aes(x=UID, y=rpc)) + coord_cartesian(xlim=c(1.1, 8.0)) + theme

ggpubr::ggarrange(p1, p2)
```

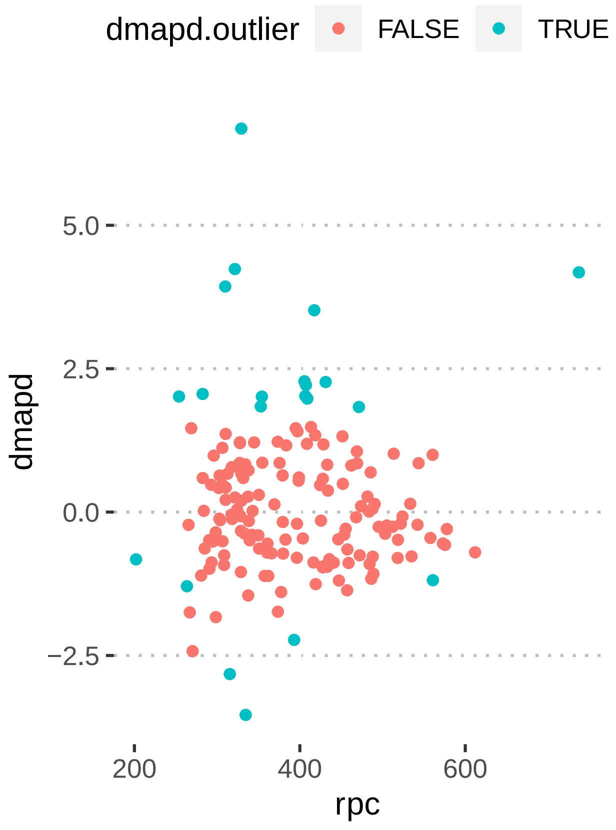




3.4 Using metrics of read distribution to detect outlier cells

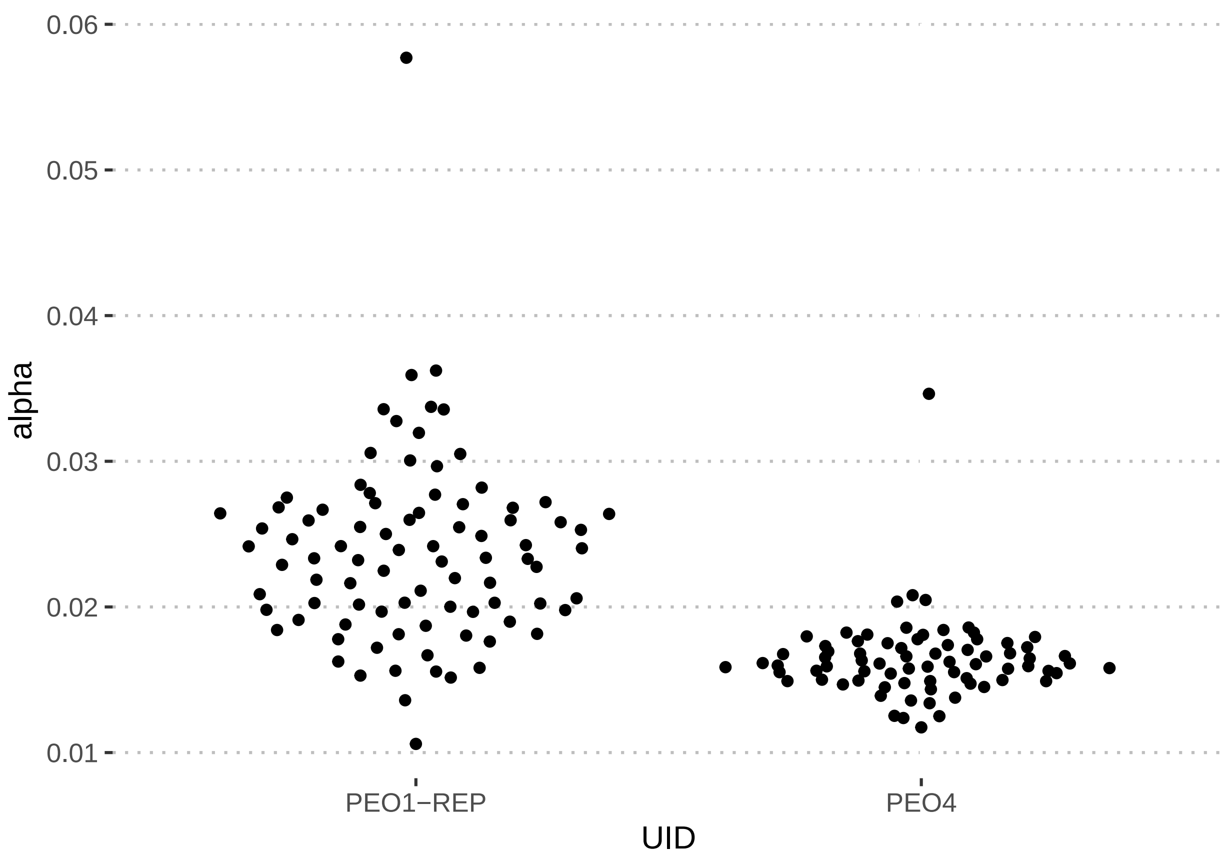
```
df <- qc_gini_norm(qc_mapd(df %>% dplyr::filter(!replicating, rpc >= 25), # minimum rpc value of 25 is
                           cutoff_value = 1.5, densityControl=TRUE), # MAPD cutoffs
                           cutoff_value = 1.5, densityControl=TRUE) # GINI cutoffs

pg = ggplot(data = df) + geom_point(aes(x=rpc, y=dgini, color=dgini.outlier)) + theme_pubclean()
pm = ggplot(data = df) + geom_point(aes(x=rpc, y=dmapd, color=dmapd.outlier)) + theme_pubclean()
ggpubr::ggarrange(pg, pm)
```





3.5 Using alpha values to detect outliers cells and sample issues

```
df = qc_alpha(df, cutoff_value=2.0, hard_cutoff = 0.05)

ggplot(data=df) + geom_quasirandom(aes(x=UID, y=alpha)) + theme_pubclean()
```



Above, we have demonstrated different quality assessments that can be used to identify a subset of cells that is reliable. In this workflow, the TODO

4 scUnique - recent copy number aberrations and joint copy number analysis

Again, this part of the pipeline is highly automated. Run the following command to obtain per sample results in

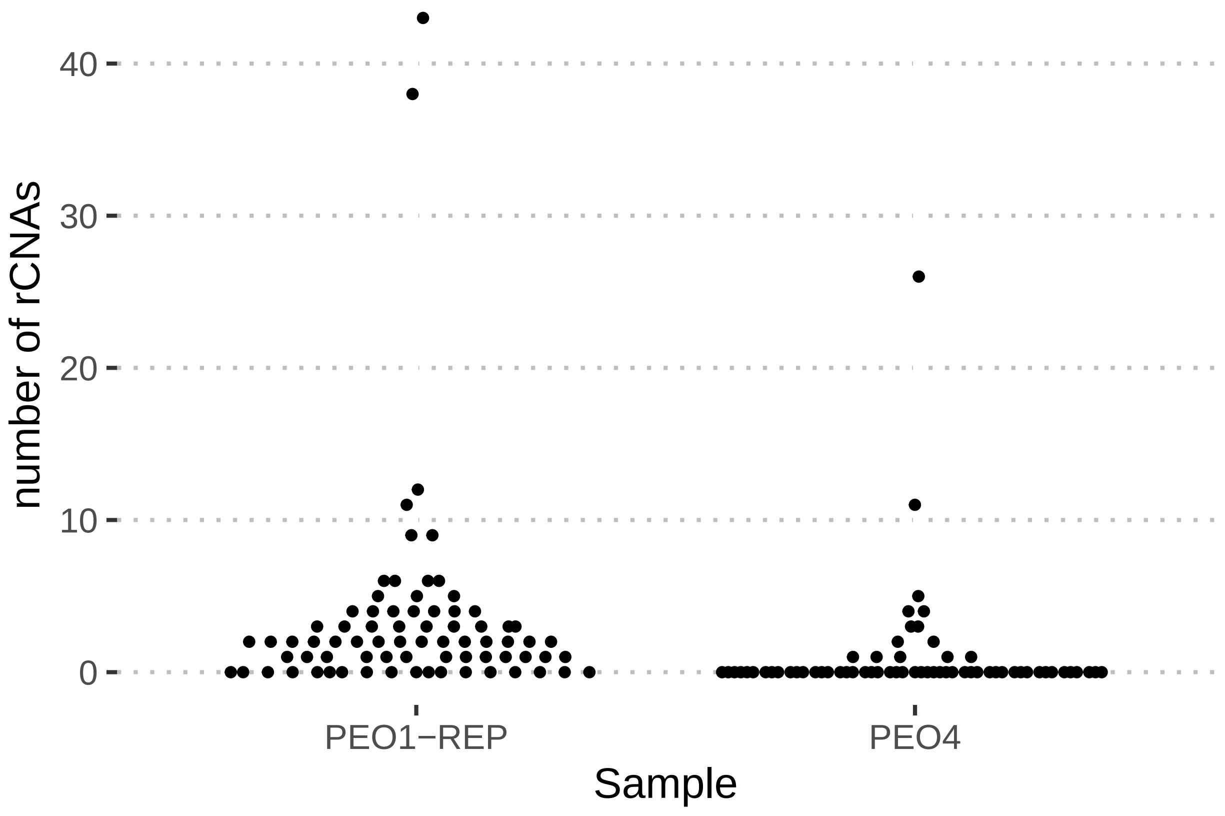
We have provided example results for running the scDNAseq workflow on the PEO1/PEO4 data sets in in vignettes/data/PEO1_500 and vignettes/data/PEO4_500 folders.

```
sampleFiles = paste0(WORKFLOW_PATH,
  c("vignettes/data/PEO1_500/PEO1_500", "vignettes/data/PEO4_500/PEO4_500"))

CN = combineQDNASets(lapply(paste0(sampleFiles, ".finalCN.RDS"), readRDS))
anno = base::factor(substr(colnames(CN), start=9, stop=12))

cn_plot = plotCopynumberHeatmap(CN, row_split = anno, show_cell_names = FALSE)
#> 'magick' package is suggested to install to give better rasterization.
#>
#> Set `ht_opt$message = FALSE` to turn off this message.
```

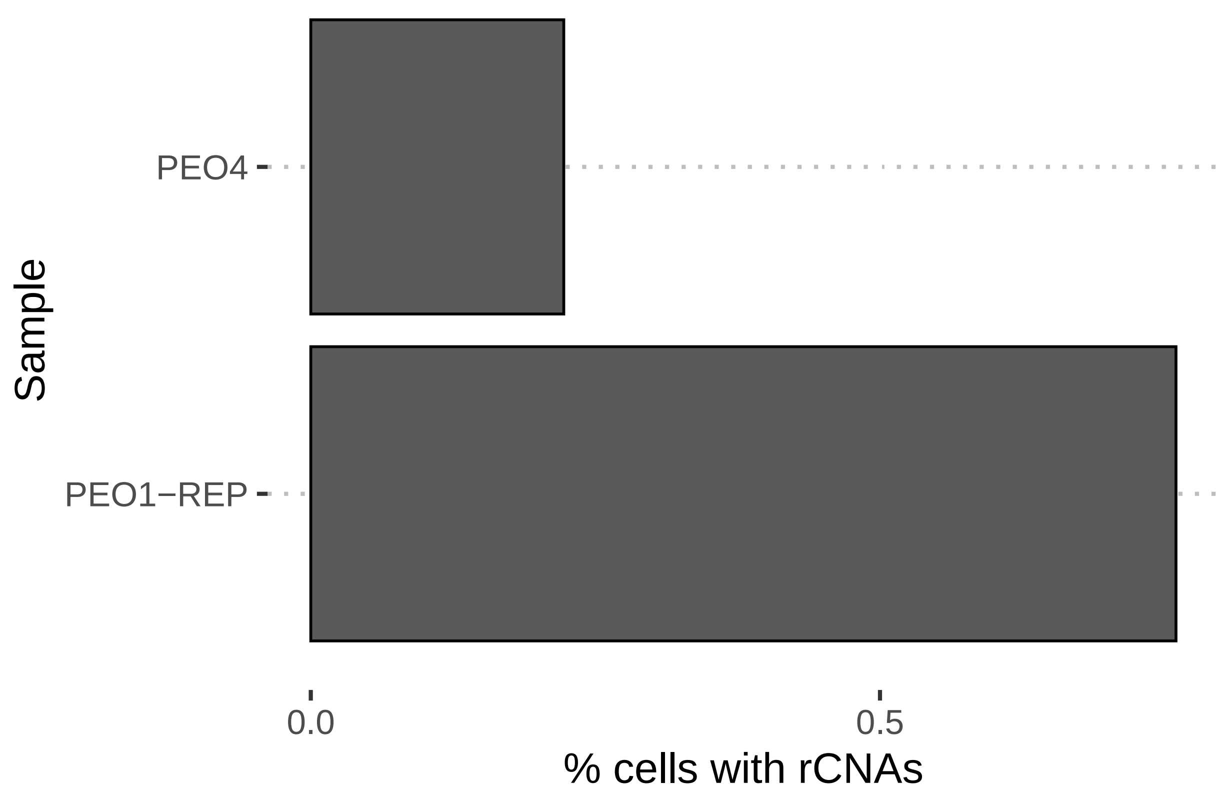
```
CN_PEO1 = readRDS(paste0(WORKFLOW_PATH, "vignettes/data/PEO1_500/PEO1_500", ".finalCN.RDS"))
tree_PEO1 = readRDS(paste0(WORKFLOW_PATH, "vignettes/data/PEO1_500/PEO1_500", ".tree.RDS"))
```

```r
dend_PEO1 = phylogram::as.dendrogram(tree_PEO1)
# NOTE: it's necessary to sort the copy number profiles according to the order of
# labels in the dendrogram
PEO_plot = plotCopynumberHeatmap(CN_PEO1[, labels(dend_PEO1)], cluster_rows = dend_PEO1, show_cell_names
#> 'magick' package is suggested to install to give better rasterization.
#>
#> Set `ht_opt$message = FALSE` to turn off this message.
#PEO_plot
```

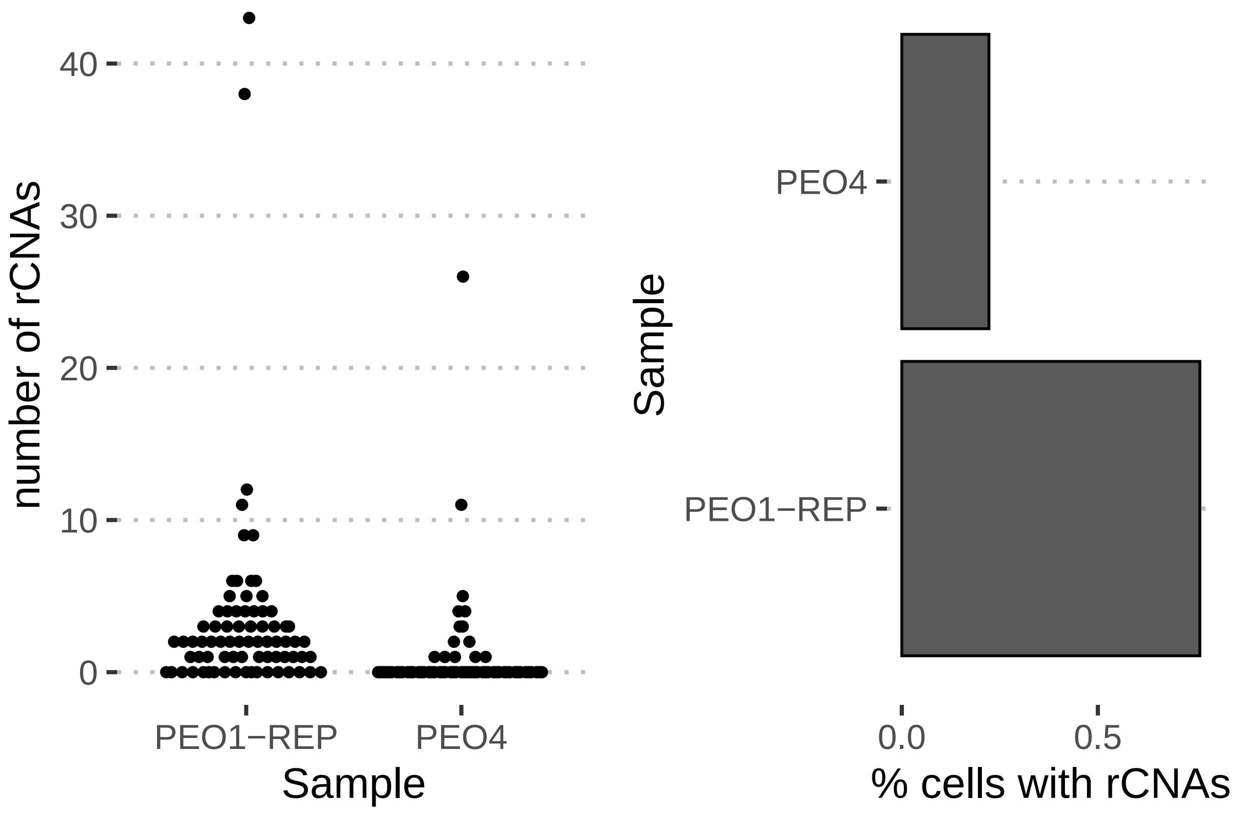
```r
pd = Biobase::pData(CN) %>%
  tidyr::separate(name, sep="_", into=c("UID", "SLX", "cellid", "celltag"), remove=FALSE) %>%
  dplyr::mutate(LIBRARY=gsub("SLX-", "", SLX), UID = gsub("UID-FML-", "", UID))

dfp = dplyr::bind_rows(future.apply::future_lapply(paste0(sampleFiles, ".df_pass_post.RDS"), readRDS)) 
  tidyr::separate(cellname, sep="_", into=c("UID", "SLX", "cellid", "celltag"), remove=FALSE) %>%
  dplyr::mutate(LIBRARY=gsub("SLX-", "", SLX), UID = gsub("UID-FML-", "", UID)) %>%
  dplyr::filter(freq == 1)

# dfp only contains cells who have rCNA ->
# create entries in data frame for rows not in df_pass
df_allcells2 = dfp %>% dplyr::group_by(UID, LIBRARY, cellname, cellindex) %>%
  dplyr::summarise(n=n(), total_size=sum(width), has_rCNA=TRUE)
#> `summarise()` has grouped output by 'UID', 'LIBRARY', 'cellname'. You can override using the
#> `.groups` argument.
df_allcells = dplyr::full_join(df_allcells2, pd %>% dplyr::select(UID, LIBRARY, cellid, name, rpc), by=
  dplyr::mutate_at(c("n", "total_size"), ~replace(., is.na(.), 0)) %>%
  dplyr::mutate(LIBRARY = ifelse(is.na(LIBRARY), LIBRARY.y, LIBRARY),
                UID = ifelse(is.na(UID), UID.y, UID),
                has_rCNA = ifelse(is.na(has_rCNA), FALSE, has_rCNA))
stopifnot(all(as.numeric(table(df_allcells$cellname))==1))


plot_nrcna = ggplot(data = df_allcells) +
  geom_quasirandom(aes(x=UID, y=n)) + xlab("Sample") + ylab("number of rCNAs") +
  theme_pubclean(base_size = 16)
plot_nrcna
```



```
plot_prcna = ggplot(data = df_allcells %>%
            dplyr::group_by(UID, LIBRARY) %>%
                              dplyr::summarise(perc = mean(has_rCNA)) %>%
                              dplyr::group_by(UID) %>%
                              dplyr::summarise(mz = mean(perc), sd=sd(perc))) +
    #geom_point(aes(x=GROUP, y=mz), color="black") +
    geom_col(aes(x=UID, y=mz), color="black") +
    scale_y_continuous(breaks=c(0.0, 0.5, 1.0)) +
    xlab("Sample") + ylab("% cells with rCNAs") +
    theme_pubclean(base_size = 16) +
  coord_flip()
#> `summarise()` has grouped output by 'UID'. You can override using the `.groups`
#> argument.
plot_prcna
```



```r
ggpubr::ggarrange(plot_nrcna, plot_prcna)
```


number of rCNAs
40
30
20
10
0
PEO1−REP
PEO4
Sample
Sample
PEO4
PEO1−REP
0.0
0.5
% cells with rCNAs