Usage Tips

Junyan Song

2017-03-11

Introduction

This is an example of using SCclust package in R. For whole genome segmented copy number profiles, SCclust implements determination of breakpoints, derivation of new feature set, hierarchical clustering on the binary matrix and identification of clone structure. This vignette aims to demonstrate the usage of SCclust through an example data.

Usage

(1) Generate and combine the segmented profiles

Make sure you have installed DNAcopy (Bioconductor package).The bin count profiles for all cells are saved in `/mnt/wigclust17/data/safe/jsong/SCexampleData/example_nyu001`. Now we generate segmented copy number profiles with GC bias correction for each cell. The output is the combined segmented profiles for all samples (cells). Note: we have three choices for count of bins (num_bin): 5000 (5k), 20000 (20k) and 50000 (50k).

```
segfile <-
  cbs.segment_all(input_file_dir = "/mnt/wigclust17/data/safe/jsong/SCexampleData/example_GL9.2",
                            num_bin = "20k", alpha = 0.05, nperm = 1000, undo.SD = 1.0, min.width = 5)
seg.quantal <- segfile$seg.quantal
ratio.quantal <- segfile$ratio.quantal
```

(2) Determination of breakpoints

Determine the position of breakpoints and the sign of CN discontinuity. Output two tables `breakpoint_table` and `ploidies_table`.

```
res1 <- preprocess_segfile(seg.quantal, num_bin = "20k",
                           eviltwins = c("CJA1024", "CJA1025"), ploidies = TRUE)
breakpoint_table <- res1$breakpoint_table
ploidies_table <- res1$ploidies_table
```

(3) Derivation of new feature set

The breakpoints are extended to an interval spanning 2*`smear`+1 bins. The centromere areas are also filtered out. This results in the `smear_table`.

The function `findpins` implement the procedure to find a minimum set of "piercing points" for the extended intervals as the new feature set and derive the incidence table. The piercing points are a smallest set of points such that each interval contains at least one of them. `pins` and `pinmat` provide the bin location of new feature set and the incidence table. We extract `cell_names` for stages later.

```
smear_table <- findsmears(breakpoint_table, smear = 1, keepboundaries = FALSE, mask_XY = TRUE)

res2 <- findpins(breakpoint_table, smear_table)
pins <- res2$pins
pinmat <- res2$pinmat
cell_names <- res2$cell_names
```

(4) Perform Fisher's tests on incidence table

We perform Fisher's tests for pairwise dissimilarity on both observed incidence table and permutated incidence tables. This procedure `simFisher_parallel` generates two vectors of p-values `true_fisherPV` and `sim_fisherPV` for observed and permutated data respectively.

```
res3 <- simFisher_parallel(pins, pinmat, sim_round = 500)
true_fisherPV <- res3$true_fisherPV
sim_fisherPV <- res3$sim_fisherPV
```

(5) Significance assessment of pairwise dissimilarity

Compute the FDRs `mat_fdr` for the observed Fisher's test p-values. Also output the dissimilarity matrix `mat_dist` based on pairwise Fisher's test p-values.

```
res4 <- fdr_fisherPV(true_fisherPV, sim_fisherPV, cell_names, lm_max = 0.001, graphic = TRUE)
mat_fdr <- res4$mat_fdr
mat_dist <- res4$mat_dist
```

(6) Identify the clone in the hierarchical clustering tree

This procedure identify the hard and soft clones and display the hierarchical tree. The clones node information can be output from `hc_clone$softclones`.

```
hc <- hclust_tree(pinmat, mat_fdr, mat_dist, hc_method = "average")
hc_clone <- find_clone(hc, fdr_thresh = -2, share_min = 0.85, n_share = 3, bymax = TRUE,
                       climb_from_size = 2, climb_to_share = 3, graphic = TRUE)
```

(7) Identify the subclones.

```
sub_hc_clone <- find_subclone(hc_clone, pinmat, pins, min_node_size = 6, sim_round = 500,
                              lm_max = 0.001, hc_method = "average",base_share = 3,
                              fdr_thresh = -2, share_min = 0.90, bymax = TRUE, climb_from_size = 2,
                              climb_to_share = 3, graphic = TRUE)
```

(8) Generate the output files for visualization using Viewer.

Create an output directory for the output files.

```
output_viewer(output_file_dir, seg.quantal, ratio.quantal, pins, pinmat,
              mat_dist, hc_clone, sub_hc_clone, subcloneTooBig = 0.8,
              num_bin = "20k", smear = 1, study="GL9.2")
```

Additional example

```
segfile <- cbs.segment_all("/mnt/wigclust17/data/safe/jsong/SCexampleData/example_nyu001",
num_bin = 5000, alpha = 0.05, nperm = 1000, undo.SD = 1.0, min.width = 5)
```

More Information

Details for arguments and functions can be found by typing e.g. `help(package="SCclust")`, `?fdr_fisherPV`.