Usage Tips

Junyan Song

2017-05-25

Introduction

This is an example of using SCclust package in R. SCclust implements determination of variable-length bin boundaries, computation of bin counts, GC normalization and segmentation on bin counts, 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 some example codes and example data.

Preprocessing

Before the implementation of SCclust, there are some preprocessing steps: download reference genome, run three python programs and index/mapping commands.

Download and unzip the reference genome files. The reference genome files are saved in a directory `/filepath/chromFa`.

Take hg19 and hg38 as examples:

http://hgdownload.cse.ucsc.edu/goldenPath/hg19/bigZips/chromFa.tar.gz
(http://hgdownload.cse.ucsc.edu/goldenPath/hg19/bigZips/chromFa.tar.gz)
http://hgdownload.cse.ucsc.edu/goldenPath/hg38/bigZips/hg38.chromFa.tar.gz
(http://hgdownload.cse.ucsc.edu/goldenPath/hg38/bigZips/hg38.chromFa.tar.gz)

(1) build index, mask pseudoautosomal regions

```
cd filepath/ChromFa
python hg19.chrY.psr.py(hg38.chrY.psr.py)
bash bowtie.build.bash
```

(2) Simulation for mappable regions

parameters: read length, e.g. 100; genome, e.g. hg or hgdm

```
mkdir k100
python generate.reads.py 100 hg| /filepath/bowtie-0.12.7/bowtie -S -t -v 0 -m 1 -f hg - |
python mappable.regions.py > /k100/mappable.regions.txt 2> /k100/mappable.regions.stderr &
```

(3) Prepare SAM files for all cells in one directory

```
/filepath/cellSAMdir
```

Directory for SAM files of all single cells, e.g. `example.rmdup.sam`

Usage of SCclust

(1) initialization

Make sure that you have installed the bioconductor package DNAcopy first.

```
install.packages("SCclust_0.1.4.tar.gz", repos = NULL, type="source")
```

Specify some work directories. `chromFa_dir` : the directory for reference genome; `k100_dir` : the directory for mappable regions.

```
library(SCclust)
chromFa_dir <- "/filepath/ChromFa"
k100_dir <- "/filepath/ChromFa/k100"
```

(2) Determine bin boundaries

The count of bins can be set as any positive integer such as 5000, 10000, 20000.

```
bin_boundaries(k100_dir, bincount = 20000)
```

(3) Compute GC content

```
varbinGC(chromFa_dir, k100_dir, Nk = "20k")
```

(4) Compute bin counts

```
SAM_dir <- "/filepath/cellSAMdir"
cellname <-"example"
bin_counts(SAM_dir,k100_dir, cellname, Nk = "20k")
```

(5) GC normalization and CBS segmentation for bin counts of a single cell

```
bin_mat <- read.table("/filepath/cellSAMdir/example.varbin.20k.txt", header = T, sep = "\t")
gc <- read.table("/filepath/k100_dir/varbin.gc.content.20k.txt", header = T, sep = "\t")
```

One example dataset is included in the SCclust package.

```
bin_mat <- example1.varbin.20k
gc <- hg19.varbin.gc.20k
```

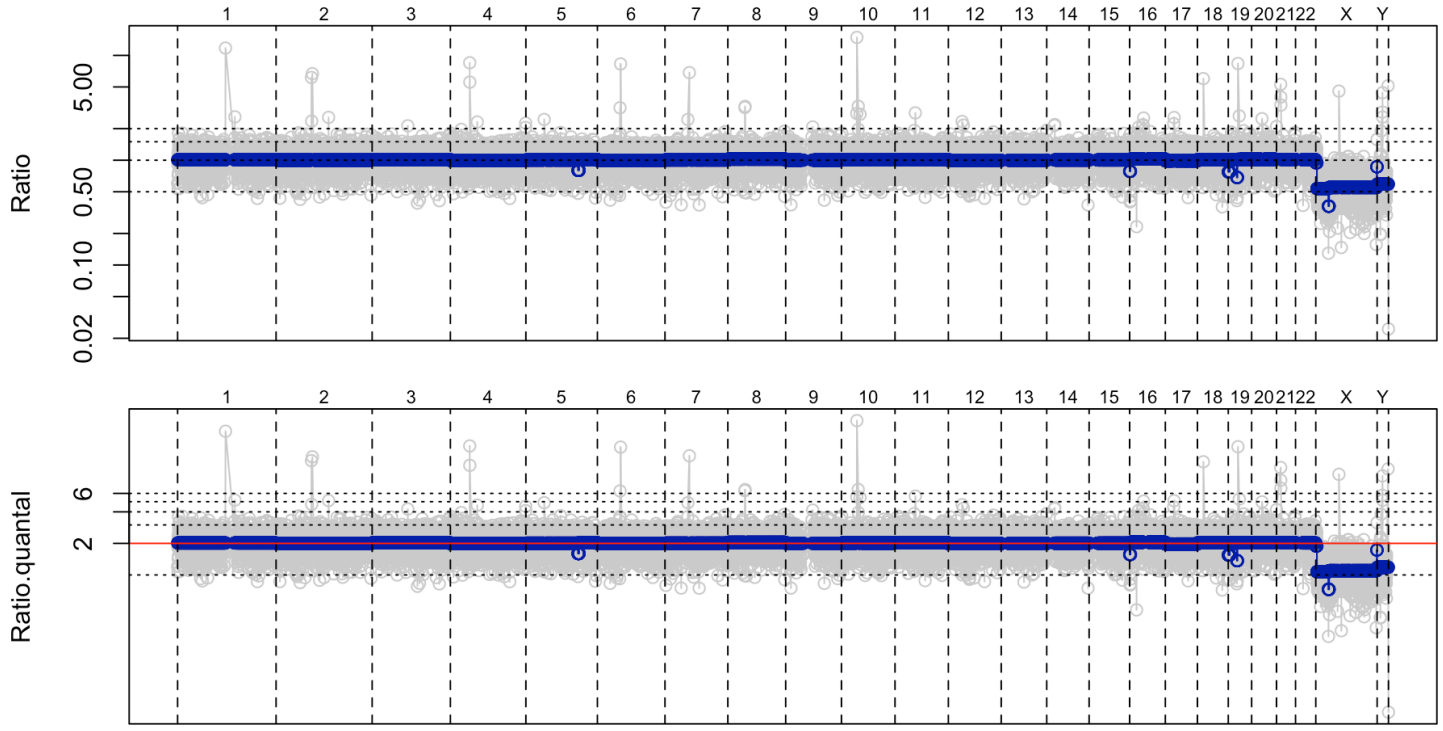
Given the bin count of one cell: `bin_mat` and the GC content table `gc` , then do GC normalization and segmentation. The package can also generate plots here: the upper plot is GC-normalized and ratio-normalized bin counts; the other plot is integer CN using “multiplier” method.

```
bin_mat_normalized <- gc_one(bin_mat,gc)
bin_mat_segmented <- cbs.segment_one(bin_mat_normalized, alpha = 0.05, nperm = 1000, undo.SD = 1.0, min.width = 5
, method = "multiplier", genome = "hg" , graphic = TRUE)
```

bin_mat

chrom	chrompos	abspos	bincount	ratio
chr1	0	0	23	0.6059369
chr1	892036	892036	12	0.3161410
chr1	1033528	1033528	14	0.3688311
chr1	1172212	1172212	15	0.3951762
chr1	1311902	1311902	10	0.2634508
chr1	1487403	1487403	16	0.4215213

Analyzing: Sample.1



bin_mat_normalized

chrom	chrompos	abspos	bincount	ratio	gc.content	lowratio
chr1	0	0	23	0.6160520	0.4450983	0.6027112
chr1	892036	892036	12	0.3336948	0.6258870	0.9486738
chr1	1033528	1033528	14	0.3850325	0.6012950	0.9043198
chr1	1172212	1172212	15	0.4107013	0.6247978	1.1576752
chr1	1311902	1311902	10	0.2823572	0.5731021	0.5339105
chr1	1487403	1487403	16	0.4363702	0.5536736	0.7109656

bin_mat_segmented

chrom	chrompos	abspos	bincount	ratio	gc.content	lowratio	chrom.numeric	seg.mean.LOWESS	seg.quantal	ratio.quantal
chr1	0	0	23	0.6160520	0.4450983	0.6027112	1	1.009612	2.019224	1.205422
chr1	892036	892036	12	0.3336948	0.6258870	0.9486738	1	1.009612	2.019224	1.897348
chr1	1033528	1033528	14	0.3850325	0.6012950	0.9043198	1	1.009612	2.019224	1.808639
chr1	1172212	1172212	15	0.4107013	0.6247978	1.1576752	1	1.009612	2.019224	2.315350
chr1	1311902	1311902	10	0.2823572	0.5731021	0.5339105	1	1.009612	2.019224	1.067821
chr1	1487403	1487403	16	0.4363702	0.5536736	0.7109656	1	1.009612	2.019224	1.421931

(6) GC normalization and CBS segmentation for all cells together into one combined table

```
segfile <- cbs.segment_all(SAM_dir, Nk = "20k", gc, alpha = 0.05, nperm = 1000, undo.SD = 1.0, min.width = 5, met
hod = "multiplier", genome = "hg")
seg.quantal <- segfile$seg.quantal
ratio.quantal <- segfile$ratio.quantal
```

(7) Determination of breakpoints

Determine the position of breakpoints and the sign of CN discontinuity. Output two tables `breakpoint_table` and `ploidies_table`. Some bad cells can be deleted. Here we suppose CJA1024 and CJA1025 are the names of bad cells.

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

(8) 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
```

(9) 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
```

(10) 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
```

(11) 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)
```

(12) 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)
```

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

Create an output directory `/filepath/viewerInput` for the output files. The directory will also be the input directory for the viewer.

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

More Information

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