SCclust T10 Tutorial

Alex Krasnitz, Jude Kendall, Junyan Song, Lubomir Chorbadjiev

2018-06-06

Contents

1 Introduction **1**

2 Data **1**
- 2.1 Data for the T10 case 1
- 2.2 Collect the Neccessary Data 2
- 2.3 Explore the Dowloaded Data 2

3 Segmentation of Varbin Data **4**
- 3.1 Prepare list of bins that are inside or intersect with centromeres regions 4
- 3.2 Exclude centromeres bins from binning scheme 5
- 3.3 Collect Varbin files for all samples 5
- 3.4 Segment varbin files 5
- 3.5 Construct results filenames and store segmentation and ratio results 6

4 Construct features and feature matrix **6**

5 Fisher FDR **7**

6 Hierarchical clustering **7**

7 Finding clones and subclones **7**

1 Introduction

The *SCclust* package implements feature selection based on breakpoints, permutations for FDRs for Fisher test p-values and identification of the clone structure in single cell copy number profiles.

In this tutorial we show how to use *SCclust* package using data, prepared by *sgains* pipeline as described in Example usage of sGAINS pipeline. *SCclust* package is called as the last step in processing data from *sgains* pipeline. In this tutoral we show how *SCclust* package could be used independently from *sgains* pipeline.

We assume that you have an R environment and have installed *SCclust* package as described in the `README.md`.

2 Data

2.1 Data for the T10 case

This tutorial is based on data published in: Navin N, Kendall J, Troge J, et al. Tumor Evolution Inferred by Single Cell Sequencing. Nature. 2011;472(7341):90-94. doi:10.1038/nature09807. In particular we will use the data for polygenomic breast tumor T10 case available from SRA. Description of samples for T10 could be found in Supplementary Table 1 | Summary of 100 Single Cells in the Polygenomic Tumor T10

We are going to run *SCclust* package on prepared by *sgains* pipeline `varbin` step. You can go through all the step in *sgains* T10 tutorial and prepare this data.

For the purposes of this tutorial we recomend you to download already prepared `varbin` data from example data. Apart from `varbin` T10 data you will need the binning scheme used in the analysis, that could be found here. And also we will need `cytoBand.txt` for HG19 that you can download it from UCSC Genome Browser.

2.2 Collect the Neccessary Data

Let us create a directory, where to store all the data used in this tutorial:

```
mkdir T10data
cd T10data
```

and let us download and extract T10 `varbin` data:

```
wget -c \
  https://github.com/KrasnitzLab/SCclust/releases/download/v1.0.0RC3/navin_t10_varbin_data.tar.gz
tar zxvf navin_t10_varbin_data.tar.gz
rm navin_t10_varbin_data.tar.gz
```

Let us also download and extract the binning scheme used in preparation of `varbin` data:

```
wget -c \
  https://github.com/KrasnitzLab/SCclust/releases/download/v1.0.0RC3/hg19_R50_B20k_bins_boundaries.txt.g
gunzip hg19_R50_B20k_bins_boundaries.txt.gz
```

And finally let us download the `cytoBand.txt` for Human reference genome *hg19*:

```
wget -c \
  http://hgdownload.cse.ucsc.edu/goldenPath/hg19/database/cytoBand.txt.gz
gunzip cytoBand.txt.gz
```

Our data directory should have following structure:

```
.
|-- T10data
    |-- cytoBand.txt
    |-- hg19_R50_B20k_bins_boundaries.txt
    |-- varbin
        |-- SRR052047.varbin.20k.txt
        |-- SRR052148.varbin.20k.txt
        |-- SRR053437.varbin.20k.txt
        ...
```

2.3 Explore the Dowloaded Data

We are going to use *SCclust* package so let us load it:

```
library("SCclust")
```

```
library(futile.logger)
flog.threshold(ERROR)
#> NULL
```

2.3.1 Binning schema

Describe the data.

```
gc_df <- read.csv("T10data/hg19_R50_B20k_bins_boundaries.txt", header = T, sep='\t')
gc_df$chrom.numeric <- chrom_numeric(gc_df$bin.chrom)
knitr::kable(head(gc_df))
```

bin.chrom	bin.start	bin.start.abspos	bin.end	bin.length	mappable.positions	gc.content	chrom.numeric
chr1	0	0	859077	859077	131390	0.4357746	1
chr1	859077	859077	999002	139925	131390	0.6280936	1
chr1	999002	999002	1141973	142971	131391	0.6026537	1
chr1	1141973	1141973	1280121	138148	131390	0.6284347	1
chr1	1280121	1280121	1435418	155297	131390	0.5757548	1
chr1	1435418	1435418	1603686	168268	131391	0.5690862	1

We are using binning scheme with 20000 bins:

```
dim(gc_df)
```

[1] 20000 8

2.3.2 Cytobands and Centromeres for HG19

Describe the data.

```
cytobands <- read.csv("T10data/cytoBand.txt", header = F, sep='\t')
knitr::kable(head(cytobands))
```

V1	V2	V3	V4	V5
chr1	0	2300000	p36.33	gneg
chr1	2300000	5400000	p36.32	gpos25
chr1	5400000	7200000	p36.31	gneg
chr1	7200000	9200000	p36.23	gpos25
chr1	9200000	12700000	p36.22	gneg
chr1	12700000	16200000	p36.21	gpos50

The main reason we need `cytoBand.txt` is to get the location of centromeres. Since centromere areas contain a lot of repetitive sequencies they are excluded from analysis when segmenting and clustering samples.

To find regions where centromeres are located we are using `calc_centroareas` function:

```
centroareas <- calc_centroareas(cytobands)
knitr::kable(head(centroareas))
```

	chrom	from	to
33	1	120600000	128900000
393	2	83300000	102700000
508	3	87200000	98300000
556	4	48200000	52700000
604	5	46100000	58900000

	chrom	from	to
655	6	57000000	63400000

So, in `centroareas` for each chromosome we have the region where the centromere is located.

2.3.3 Varbin Samples Data

Describe the data.

For each `varbin` sample

```
sample_df <- read.csv("T10data/varbin/SRR052047.varbin.20k.txt", header=T, sep='\t')
knitr::kable(head(sample_df))
```

chrom	chrompos	abspos	bincount	ratio
chr1	0	0	51	0.3327000
chr1	859077	859077	57	0.3718412
chr1	999002	999002	89	0.5805941
chr1	1141973	1141973	53	0.3457471
chr1	1280121	1280121	99	0.6458294
chr1	1435418	1435418	63	0.4109824

```
sample_df <- read.csv("T10data/varbin/SRR052148.varbin.20k.txt", header=T, sep='\t')
knitr::kable(head(sample_df))
```

chrom	chrompos	abspos	bincount	ratio
chr1	0	0	125	0.5530061
chr1	859077	859077	69	0.3052594
chr1	999002	999002	90	0.3981644
chr1	1141973	1141973	57	0.2521708
chr1	1280121	1280121	98	0.4335568
chr1	1435418	1435418	84	0.3716201

3 Segmentation of Varbin Data

3.1 Prepare list of bins that are inside or intersect with centromeres regions

Using centromere regions calculated from `cytoBand.txt` we calculate which bins are inside or intersect with centromere regions using `calc_regions2bins` function:

```
centrobins <- calc_regions2bins(gc_df, centroareas)
length(centrobins)
```

[1] 1513

3.2 Exclude centromeres bins from binning scheme

After excluding centromeres bins from binning scheme, the new binning scheme has smaller number of bins:

```
gc_df <- gc_df[-centrobins, ]
dim(gc_df)
```

[1] 18487 8

3.3 Collect Varbin files for all samples

The function `varbin_input_files` helps us to collect varbin files for all samples from our `T10data/varbin` directory:

```
varbin_files <- varbin_input_files("T10data/varbin", "*.varbin.20k.txt")
knitr::kable(head(varbin_files))
```

names	cells	paths
SRR052047.varbin.20k.txt	SRR052047	T10data/varbin/SRR052047.varbin.20k.txt
SRR052148.varbin.20k.txt	SRR052148	T10data/varbin/SRR052148.varbin.20k.txt
SRR053437.varbin.20k.txt	SRR053437	T10data/varbin/SRR053437.varbin.20k.txt
SRR053600.varbin.20k.txt	SRR053600	T10data/varbin/SRR053600.varbin.20k.txt
SRR053602.varbin.20k.txt	SRR053602	T10data/varbin/SRR053602.varbin.20k.txt
SRR053604.varbin.20k.txt	SRR053604	T10data/varbin/SRR053604.varbin.20k.txt

Let us use not all the 100 samples but a subset of first 10 samples found by `varbin_input_files` function:

```
varbin_files <- varbin_files[seq(10),]
dim(varbin_files)
#> [1] 10  3
```

3.4 Segment varbin files

Explain segmentation

```
res <- segment_varbin_files(varbin_files, gc_df, centrobins)
#> Analyzing: Sample.1
#> Analyzing: Sample.1
#> Analyzing: Sample.1
#> Analyzing: Sample.1
#> Analyzing: Sample.1
#> Analyzing: Sample.1
#> Analyzing: Sample.1
#> Analyzing: Sample.1
#> Analyzing: Sample.1
#> Analyzing: Sample.1
```

The `segment_varbin_files` function returns list of segmentation and ratios for the samples

```
knitr::kable(head(res$seg))
```

chrom	chrompos	abspos	SRR052047	SRR052148	SRR053437	SRR053600	SRR053602	SRR053604	SRR05
1	0	0	1.914278	1.961422	1.981773	1.959143	1.96083	1.957483	1.98
1	859077	859077	1.914278	1.961422	1.981773	1.959143	1.96083	1.957483	1.98
1	999002	999002	1.914278	1.961422	1.981773	1.959143	1.96083	1.957483	1.98
1	1141973	1141973	1.914278	1.961422	1.981773	1.959143	1.96083	1.957483	1.98
1	1280121	1280121	1.914278	1.961422	1.981773	1.959143	1.96083	1.957483	1.98
1	1435418	1435418	1.914278	1.961422	1.981773	1.959143	1.96083	1.957483	1.98

```
knitr::kable(head(res$ratio))
```

chrom	chrompos	abspos	SRR052047	SRR052148	SRR053437	SRR053600	SRR053602	SRR053604	SRR05
1	0	0	0.6593663	1.079580	1.222141	1.108080	1.079869	1.353035	0.79
1	859077	859077	1.8631489	1.781506	2.062367	1.765271	1.896897	2.534985	1.87
1	999002	999002	2.4473278	1.908000	2.396548	2.878972	2.471616	1.510905	2.12
1	1141973	1141973	1.7385504	1.479967	1.654073	1.247160	2.736819	1.284730	2.77
1	1280121	1280121	2.2813321	1.692508	1.706780	1.688706	2.188923	1.575746	1.82
1	1435418	1435418	1.3992734	1.382124	1.448576	1.665827	1.627200	1.344241	1.28

3.5 Construct results filenames and store segmentation and ratio results

```
filenames <- case_filenames("T10data/results", "NavinT10")
```

```
save_table(filenames$seg, res$seg)
save_table(filenames$ratio, res$ratio)
```

```
cells <- uber_cells(res$seg)$cells
save_table(filenames$cells, data.frame(cell=cells))
```

```
dir("T10data/results")
#> [1] "NavinT10.cells.txt"      "NavinT10.featuremat.txt"
#> [3] "NavinT10.features.txt"   "NavinT10.ratio.txt"
#> [5] "NavinT10.seg.txt"        "NavinT10.sim_pv.txt"
#> [7] "NavinT10.true_pv.txt"
```

4 Construct features and feature matrix

Explain how we calculate features

```
pins <- calc_pinmat(gc_df, res$seg, dropareas=centroareas)
pinmat_df <- pins$pinmat
pins_df <- pins$pins
save_table(filenames$featuremat, pinmat_df)
save_table(filenames$features, pins_df)
```

```
dir("T10data/results")
#> [1] "NavinT10.cells.txt"      "NavinT10.featuremat.txt"
#> [3] "NavinT10.features.txt"   "NavinT10.ratio.txt"
#> [5] "NavinT10.seg.txt"        "NavinT10.sim_pv.txt"
#> [7] "NavinT10.true_pv.txt"
```

5 Fisher FDR

```r
fisher <- sim_fisher_wrapper(
      pinmat_df, pins_df, njobs=30, nsim=150, nsweep=10)
true_pv <- fisher$true
sim_pv <- fisher$sim
```

```r
# devtools::load_all()
# flog.threshold(DEBUG)
mfdr <- fisher_fdr(true_pv, sim_pv, cells)
mdist <- fisher_dist(true_pv, cells)
```

```r
save_mat(filenames$true_pv, true_pv)
save_mat(filenames$sim_pv, sim_pv)
```

6 Hierarchical clustering

```r
hc <- hclust_tree(pinmat_df, mfdr, mdist)
tree_df <- tree_py(mdist, method='average')
```

```r
save_table(filenames$tree, tree_df)
```

7 Finding clones and subclones

```r
hc <- find_clones(hc)
subclones <- find_subclones(hc, pinmat_df, pins_df, nsim=nsim)
```

```r
save_table(filenames$clone, subclones)
```