Starfish tutorial

Oct 25, 2021

“Starfish” is an R package which takes structural variations (SVs) and copy number variations (CNVs) detected from the whole-genome sequencing data, identifies complex genomic rearrangements (CGRs), and then infers CGR signatures based on the patterns of CNVs and SV distributions. Here, we define “CGRs” broadly as complex events formed via one-time events rather than accumulation of multiple individual events over time. Starfish can perform *de novo* CGR signature decomposition based on many genomes. It can also infer the CGR signatures from a single genome using a classifier built upon 2428 tumors from Pan-cancer Analysis of Whole Genomes (PCAWG). The current version can only be used on human genome.

1. Prerequisites

1.1 Software environment

Starfish is entirely written in R and has been tested on R 3.5.0, 3.6.0 and 4.0.0, but any version above 3.0.1 should be fine to use.

1.2 Package dependencies

Starfish depends on several R packages including graph, BiocGenerics, S4Vectors, foreach, GenomeInfoDb, GenomicRanges, IRanges, ConsensusClusterPlus, neuralnet, plyr, data.table, MASS, ggplot2, gridExtra, dplyr, factoextra, dendextend, gplots, ggpubr, reshape2, cowplot, patchwork, Cairo and ggforce. It also requires a modified version of ShatterSeek (ShatterSeeky) which is provided as ShatterSeeky_0.4.tar.gz. As the installation of ShatterSeeky depends on the other packages, please install it after the others. The example data can be found under data/ folder and the expected output files are provided under example/ folder. In case you cannot reproduce the example results, you may need to install the versions of packages specified below. It is possible that the newer versions are not compatible. The versions of tested packages are: graph 1.68.0, BiocGenerics 0.36.0, S4Vectors 0.28.0, foreach 1.5.1, Genome-InfoDb 1.26.0, GenomicRanges 1.42.0, IRanges 2.24.0, ConsensusClusterPlus 1.54.0, neuralnet 1.44.2, plyr 1.8.6, data.table 1.14.0, MASS 7.3.53.1, ggplot2 3.3.3, gridExtra 2.3, dplyr 1.0.5, factoextra 1.0.7, dendextend 1.15.1, gplots 3.1.1, ggpubr 0.4.0, reshape2 1.4.4, cowplot 1.1.1, patchwork 1.1.1, Cairo 1.5.12.2, ggforce 0.3.3.

graph, BiocGenerics, S4Vectors, GenomeInfoDb, GenomicRanges, IRanges, ConsensusClusterPlus can be installed as:

```
if (!requireNamespace("BiocManager", quietly = TRUE)) install.packages("BiocManager")
star_pkgs <- c("graph", "BiocGenerics", "S4Vectors", "GenomeInfoDb",
    "GenomicRanges", "IRanges", "ConsensusClusterPlus")
BiocManager::install(star_pkgs)
```

foreach, neuralnet, plyr, data.table, MASS, ggplot2, gridExtra, dplyr, factoextra, dendextend, gplots, ggpubr, reshape2, cowplot, patchwork, Cairo, ggforce can be installed as:

```
install.packages(c("foreach", "neuralnet", "plyr", "data.table", "MASS",
    "ggplot2", "gridExtra", "dplyr", "factoextra", "dendextend", "gplots",
    "ggpubr", "reshape2", "cowplot", "patchwork", "Cairo", "ggforce"))
```

ShatterSeeky could be installed as:

```
git clone https://github.com/yanglab-computationalgenomics/Starfish.git
cd Starfish
R CMD INSTALL ShatterSeeky_0.4.tar.gz
```

The users can load all packages at once as:

```
Packages <- c("ShatterSeeky", "GenomeInfoDb", "plyr", "data.table",
    "GenomicRanges", "IRanges", "MASS", "ggplot2", "grid", "gridExtra",
    "dplyr", "ConsensusClusterPlus", "factoextra", "gplots", "ggpubr",
    "reshape2", "cowplot", "scales", "patchwork", "Cairo", "ggforce")
lapply(Packages, library, character.only = TRUE)
```

2. How to use Starfish

In the following sections, we will illustrate how to install and use Starfish to detect, classify and visualize CGRs using SV and CNV data.

2.1 Installation

Starfish could be installed remotely by:

```
if (!requireNamespace("devtools", quietly = TRUE)) install.packages("devtools")
library(devtools)
install_github("yanglab-computationalgenomics/Starfish")
```

Alternatively, one can download the latest release of Starfish by running the following command in a bash terminal:

```
git clone https://github.com/yanglab-computationalgenomics/Starfish.git
R CMD INSTALL Starfish
```

2.2 Load SV, CNV and sample data into R

One can first load Starfish and the test data provided by the package. The test data correspond to the SV, CNV and gender data we used to generate Figure 1c of our manuscript[1].

```
library(Starfish)
data("example_sv")
data("example_cnv")
data("example_sample")
```

Running this command loads three R data frames, corresponding to the somatic SVs, CNVs and gender information for 6 tumors. The somatic variants are generated by the PCAWG project. Starfish accepts SV and CNV calls from any variant caller, as long as they are encoded in the required format (see below). For bulk tumor sequencing samples, it is highly recommended to use algorithms, such as Batternberg and Sequenza, to derive integer copy numbers, rather than using copy ratios between tumor and normal.

2.2.1 SV data:

Starfish requires the SV data to be stored in a data frame with the following columns:

- “chrom1” (character): chromosome for the first breakpoint
- “pos1” (numeric): genomic coordinate for the first breakpoint, and pos1 should be smaller than pos2 for intrachromosomal SVs (i.e., DEL, DUP, h2hINV and t2tINV)
- “chrom2”(character): chromosome for the second breakpoint
- “pos2” (numeric): genomic coordinate for the second breakpoint
- “svtype” (character): type of SV, encoded as DEL (deletion-like; +/-), DUP (duplication-like; -/+), h2hINV (head-to-head inversion; +/+), t2tINV (tail-to-tail inversion; -/-) and TRA (translocation)
- “strand1” (character): strand information for the first breakpoint (e.g., + for DEL)
- “strand2” (character): strand information for the second breakpoint (e.g., - for DEL)
- “sample” (character): sample ID

Chromosome names could be either Ensembl style or UCSC style, e.g. “Chr1”, “chr1” and “1” are all accepted, and only chromosomes 1-22, X and Y are considered.

```
print(head(example_sv), row.names = FALSE)
```

```
##  chrom1     pos1 chrom2     pos2 strand1 strand2 svtype
##      11 36219433      2 52018778       +       -    TRA
##      11 39939312     11 39941515       +       -    DEL
##      12 40811811     12 63685824       +       + h2hINV
##      12 40838496     12 85876878       -       - t2tINV
##      12 57880298     12 79679264       +       -    DEL
##      12 57913369     12 75990126       -       +    DUP
##                                sample
##  07f16397-71bb-4594-ad4d-caa7d2baeabd
##  07f16397-71bb-4594-ad4d-caa7d2baeabd
##  07f16397-71bb-4594-ad4d-caa7d2baeabd
##  07f16397-71bb-4594-ad4d-caa7d2baeabd
##  07f16397-71bb-4594-ad4d-caa7d2baeabd
##  07f16397-71bb-4594-ad4d-caa7d2baeabd
```

2.2.2 CNV data:

Starfish requires the CNV data to be stored in a data frame with the following columns:

- “chromosome” (character): chromosome
- “start” (numeric): start coordinate for the CN segment
- “end” (numeric): end coordinate for the CN segment
- “total_cn” (numeric): total copy number, which could be either integer or decimal
- “sample” (character): sample ID, which should match the ones in SV and gender data frames

Chromosome names could be either Ensembl style or UCSC style, e.g. “Chr1”, “chr1” and “1” are all accepted, and only chromosomes 1-22, X and Y are considered.

```
print(head(example_cnv), row.names = FALSE)
```

```
##  chromosome     start       end total_cn                               sample
##           4  68582077 191154275        2 07f16397-71bb-4594-ad4d-caa7d2baeabd
##          12 112420925 112603567        1 07f16397-71bb-4594-ad4d-caa7d2baeabd
##           6  68336278  69355562        2 07f16397-71bb-4594-ad4d-caa7d2baeabd
##           7   7820155   8996369        2 07f16397-71bb-4594-ad4d-caa7d2baeabd
##           3 167181960 168958768        2 07f16397-71bb-4594-ad4d-caa7d2baeabd
##          12  84251201  84277057        1 07f16397-71bb-4594-ad4d-caa7d2baeabd
```

Starfish will calculate the copy number (CN) baseline for each chromosome to identify loss and gain fragments in a chromosome-wise manner. It will use the gender information to call losses and gains on chromosome X.

2.2.3 Gender data:

Starfish requires the gender data to be stored in a data frame with following columns:

- “sample” (character): sample ID, which should match SVs and CNVs
- “gender” (character): gender for the sample, which could be “Female,”female“,”F“,”f“,”Male” ,“male”,“M”, or “m”. If the gender is unknown, any other characters could be given, such as “unknown”, and the gender will be inferred by the CN baseline of chromosome X (see details in step 2.3.3)

```
print(head(example_sample), row.names = FALSE)
```

```
##                                  sample gender
##  3db6e6cc-1a06-49b9-834e-b6611cde4c4b   Male
##  07f16397-71bb-4594-ad4d-caa7d2baeabd   Male
##  c00de7a0-0b09-4e07-988c-ef2a7f8e932a   Male
##  2b6d4d66-7f0b-4bc0-b3d6-171956a937c5 Female
##  18f5e75e-c623-11e3-bf01-24c6515278c0   Male
##  fc8130df-897d-5404-e040-11ac0d485e0a Female
```

2.3 Running Starfish

There are four components of Starfish: **starfish_link**, **starfish_feature**, **starfish_sig** and **starfish_plot**. Once the input data are loaded, we could run them step by step to obtain and examine intermediate outputs or we could use **starfish_all** to run them all at once.

2.3.1 starfish_link

starfish_link loads an SV data frame and identifies "seed" CGR regions, "linked" CGR regions, and complex SVs using modified ShatterSeek[2]. Oscillating-copy-state criteria is removed from the original ShatterSeek package. In each sample, linked regions are identified if they are connected by at least two translocations within 10 kb of any seed regions. The search is performed iteratively until no new linked regions could be found. A CGR event is defined as all connected seed and linked regions combined (Figure 1).

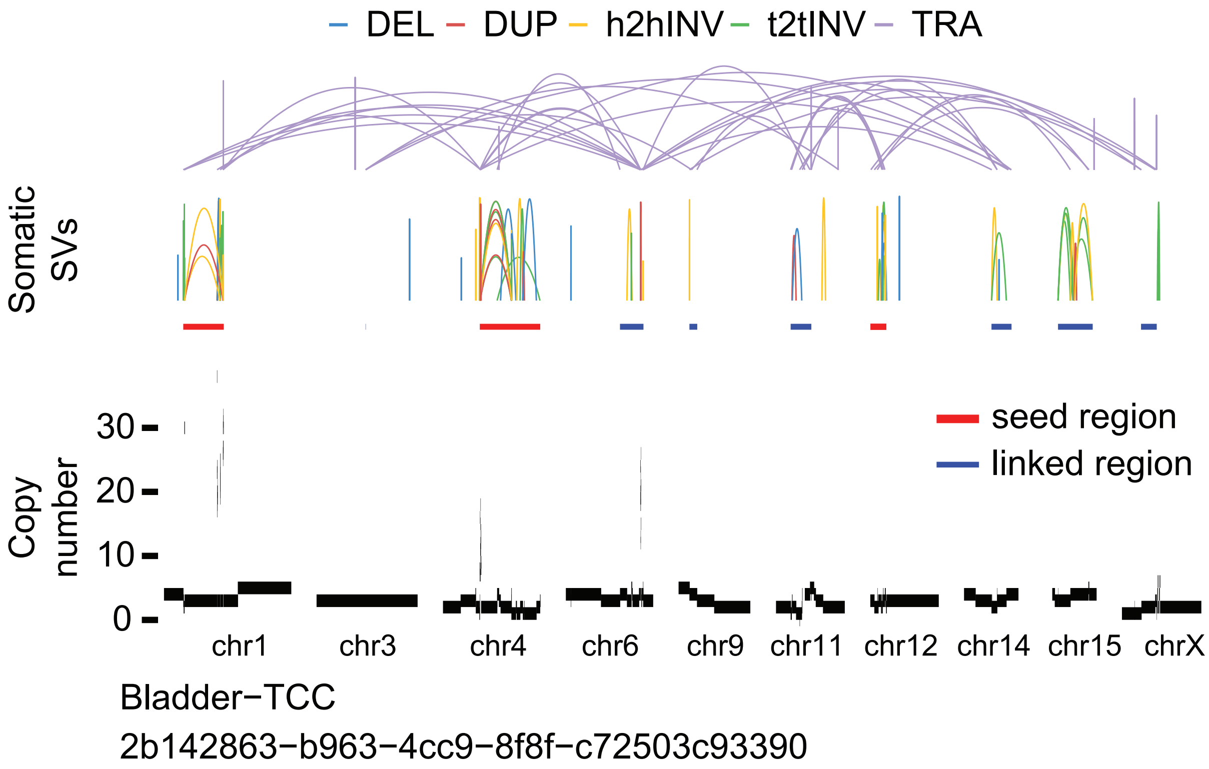

Figure 1: "seed" regions and "linked" regions in a CGR of a Bladder-TCC sample "2b142863-b963-4cc9-8f8f-c72503c93390". SVs are shown as curved lines, and SV types are shown in different colors. "Seed" and "linked" regions are plotted as colored bars.

Usage:

```
starfish_link(sv_file, prefix = "")
```

Input:

- "sv_file": the SV data frame defined previously
- "prefix": the prefix for all intermediate files, default is none

Example:

Starfish will output the progress of "seed" region and "linked" region calling.

```
starfish_link_out = starfish_link(example_sv, prefix = "example")
```

```
## Running..
##
##
## Evaluating the statistical criteria
## Successfully finished!
## Running..
##
##
## Evaluating the statistical criteria
## Successfully finished!
## Running..
##
##
## Evaluating the statistical criteria
## Successfully finished!
## Running..
##
##
## Evaluating the statistical criteria
## Successfully finished!
## Running..
##
##
## Evaluating the statistical criteria
## Successfully finished!
## Running..
##
##
## Evaluating the statistical criteria
## Successfully finished!
## Running..
```

```
##
##
## Evaluating the statistical criteria
## Successfully finished!
## Running..
##
##
## Evaluating the statistical criteria
## Successfully finished!
## Running..
##
##
## Evaluating the statistical criteria
## Successfully finished!
## Running..
##
##
## Evaluating the statistical criteria
## Successfully finished!
## Running..
##
##
## Evaluating the statistical criteria
## Successfully finished!
## Running..
##
##
## Evaluating the statistical criteria
## Successfully finished!
## [1] "Iteration 1"
## [1] "Starfish is connecting chromothriptic regions..."
```

Output:

The function **starfish_link** returns an instance that contains three data frames: *interleave_tra_complex_sv*, *mergecall*, and *starfish_call*.

interleave_tra_complex_sv contains complex SVs in the CGR regions and "complex = 2" means both breakpoints are in the CGR regions. A CSV file "example_connected_CGR_complex_SV.csv" will be generated under the working space:

```
print(head(starfish_link_out$interleave_tra_complex_sv), row.names = FALSE)
```

```
##  chrom1     pos1 chrom2     pos2 svtype complex
##      19 22936736      4  4188431    TRA       2
##      19 22937258      4  4188272    TRA       2
##      19 19053346     19 19817150 h2hINV       2
##      19 19368099     19 22447855 t2tINV       2
##      19 19769579      19 19817431 t2tINV       2
##      19 19792588      19 21652467 h2hINV       2
##                                 sample
##  18f5e75e-c623-11e3-bf01-24c6515278c0
##  18f5e75e-c623-11e3-bf01-24c6515278c0
##  18f5e75e-c623-11e3-bf01-24c6515278c0
##  18f5e75e-c623-11e3-bf01-24c6515278c0
##  18f5e75e-c623-11e3-bf01-24c6515278c0
##  18f5e75e-c623-11e3-bf01-24c6515278c0
```

mergecall contains "seed" CGR regions which is identified by the modified ShatterSeek. "Call3" refers to the criteria of "at least 3 interleaved intrachromosomal SVs and 4 or more interchromosomal SVs" to call CGRs used by ShatterSeek, and "call6" refers to the criteria of "at least 6 interleaved intrachromosomal SVs". The regions pass either call3 or call6 criteria will be defined as "seed" CGR regions.

```
print(head(starfish_link_out$mergecall), row.names = FALSE)
```

```
##  chr     start       end                               sample call3 call6
##    3 162683868 169237872 07f16397-71bb-4594-ad4d-caa7d2baeabd    no   CGR
##    5  30217747 163056821 07f16397-71bb-4594-ad4d-caa7d2baeabd    no   CGR
##    6  64700081 152466236 07f16397-71bb-4594-ad4d-caa7d2baeabd    no   CGR
##    7   7799683  14177776 07f16397-71bb-4594-ad4d-caa7d2baeabd    no   CGR
##   12  40811811 113578120 07f16397-71bb-4594-ad4d-caa7d2baeabd    no   CGR
##    X 126784598 152948223 07f16397-71bb-4594-ad4d-caa7d2baeabd    no   CGR
```

starfish_call contains final CGR regions (both seed and linked regions) and a CSV file "example_connected_CGR_event.csv" will be generated under the working space:

```
print(head(starfish_link_out$starfish_call), row.names = FALSE)
```

```
##  chr     start       end                               sample CGR_status
##   19  19053346  23741257 18f5e75e-c623-11e3-bf01-24c6515278c0        CGR
##    4   4188272   4188431 18f5e75e-c623-11e3-bf01-24c6515278c0       link
##    1 145028118 172302421 3db6e6cc-1a06-49b9-834e-b6611cde4c4b        CGR
##   12   1653428  91454275 3db6e6cc-1a06-49b9-834e-b6611cde4c4b        CGR
##    6  82279193 105887226 3db6e6cc-1a06-49b9-834e-b6611cde4c4b       link
##    7  65538566 150670275 3db6e6cc-1a06-49b9-834e-b6611cde4c4b        CGR
##  link_chromosome                                      cluster_id
##             4_19       18f5e75e-c623-11e3-bf01-24c6515278c0_4_19
##             4_19       18f5e75e-c623-11e3-bf01-24c6515278c0_4_19
##       1_6_7_12_X 3db6e6cc-1a06-49b9-834e-b6611cde4c4b_1_6_7_12_X
##       1_6_7_12_X 3db6e6cc-1a06-49b9-834e-b6611cde4c4b_1_6_7_12_X
##       1_6_7_12_X 3db6e6cc-1a06-49b9-834e-b6611cde4c4b_1_6_7_12_X
##       1_6_7_12_X 3db6e6cc-1a06-49b9-834e-b6611cde4c4b_1_6_7_12_X
```

The columns are:

- "chr": chromosome of the CGR region
- "start": start coordinate of the CGR region
- "end": end coordinate of the CGR region
- "sample": sample ID
- "CGR_status": "CGR" means the region is the "seed" region identified by ShatterSeek, and "link" means the region is the "linked" region connected to "seed" regions
- "link_chromosome": the chromosomes linked together in the CGRs
- "cluster_id": sample ID plus link_chromosome, which is the unique CGR event ID

2.3.2 starfish_feature

This function loads CGR regions and SVs in the CGRs reported by starfish_link, then combines CNV calls and gender to construct a feature matrix for clustering and classification in step 2.3.3. In Figure 2, the size of CGR region [s] equals to $\sum a : g$. Breakpoint dispersion score is defined as mean absolute deviation of [a:g] which measures the randomness of breakpoint distribution of a CGR. Copy loss and copy gain percentages are calculated by $\sum b, f/$[s] and $\sum d, g/$[s]. Telomere loss percentage is t/L. The maximum CN is 10 in this example.

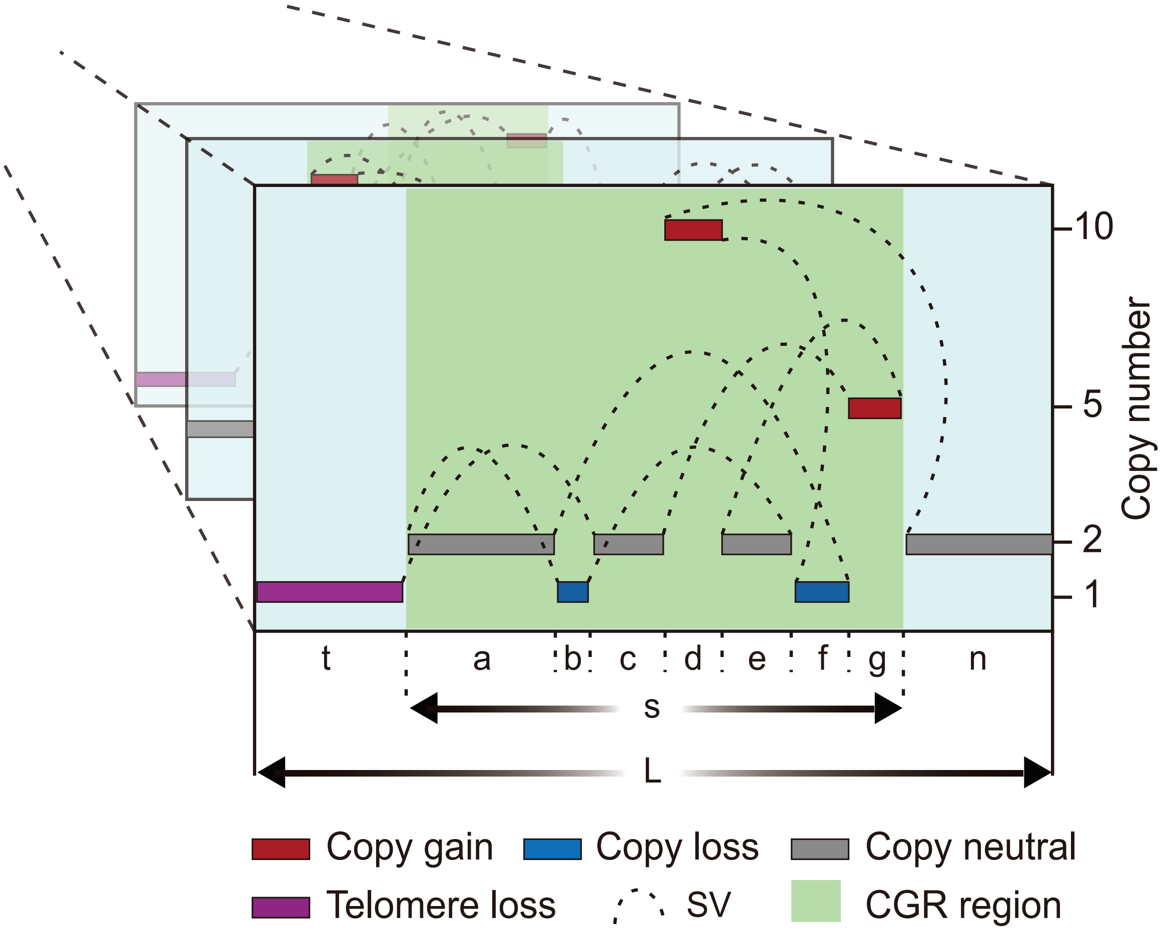

Figure 2: Genomic features in CGR regions. Each layer represents a unique CGR event with CGR region highlighted in green. Different CN fragments are painted in different colors. Letters ”a” to ”g”, ”n”, ”t”, ”s” and ”L” denote the lengths of DNA segments, which are used for feature computation.

Usage:

```
starfish_feature(cgr, complex_sv, cnv_file, gender_file, prefix = "",
    genome_v = "hg19", cnv_factor = "auto", arm_del_rm = TRUE)
```

Input:

- “cgr”: the output of starfish_link_out$starfish_call
- “complex_sv”: the output of starfish_link_out$interleave_tra_complex_sv
- “cnv_file”: the CNV data frame defined previously
- “gender_file”: the sample gender defined previously
- “prefix”: the prefix for all intermediate files, default is none
- “genome_v”: the genome assembly used to call SV and CNV. It should be “hg19” or “hg38”, default is “hg19”

- "cnv_factor": the CN fluctuation beyond or below baseline to identify loss and gain fragments for samples with decimal CN, default is "auto", or users can provide a value between 0 and 1
- "arm_del_rm": logical value for whether or not arm level deletions should be removed, default is TRUE

For each chromosome, **starfish_feature** will calculate the longest CN and set it as the chromosome CN baseline. In tumors, aneuploidy is common, so each chromosome can have a different baseline. By doing this, chromosome-specific copy-loss and copy-gain fragments can be identified. Sometimes the CNV data may be noisy, such as from single-cell WGS. It would be challenging to determine the CN baselines and CNVs. For example, Figure 3 shows a noisy CN profile from single cell sequencing data of an experimentally induced chromothriptic cell[3]. The "cnv_factor" parameter is designed particularly for these samples. Copy loss fragments are defined as fragments with CN less than (CN baseline)*(1-cnv_factor), and copy gain fragments are those with CN greater than (CN baseline)*(1+cnv_factor). The default value is "auto", and Starfish will decide the value automatically based on the CN profile. In "auto" mode, if the CN segments all have integer CNs, the algorithm will assume the CN profile comes from bulk tumor sequencing samples with high quality CN profiles, such as PCAWG samples, cnv_factor will be automatically set to 0. If the CN segments are decimal values, the algorithm will assume the CN profile is noisy, such as single cell sequencing data in Figure 3. Cnv_factor will be automatically set to 0.15. The users may set this parameter manually with a value ranging from 0 to 1 based on the quality of CN profiles. For each sample without gender data, Starfish will infer the gender based on the CN baseline of chromosome X. If (1-cnv_factor)≤(CN baseline of chromosome X)≤(1+cnv_factor), the gender will be inferred as "Male", otherwise it will be inferred as "Female".

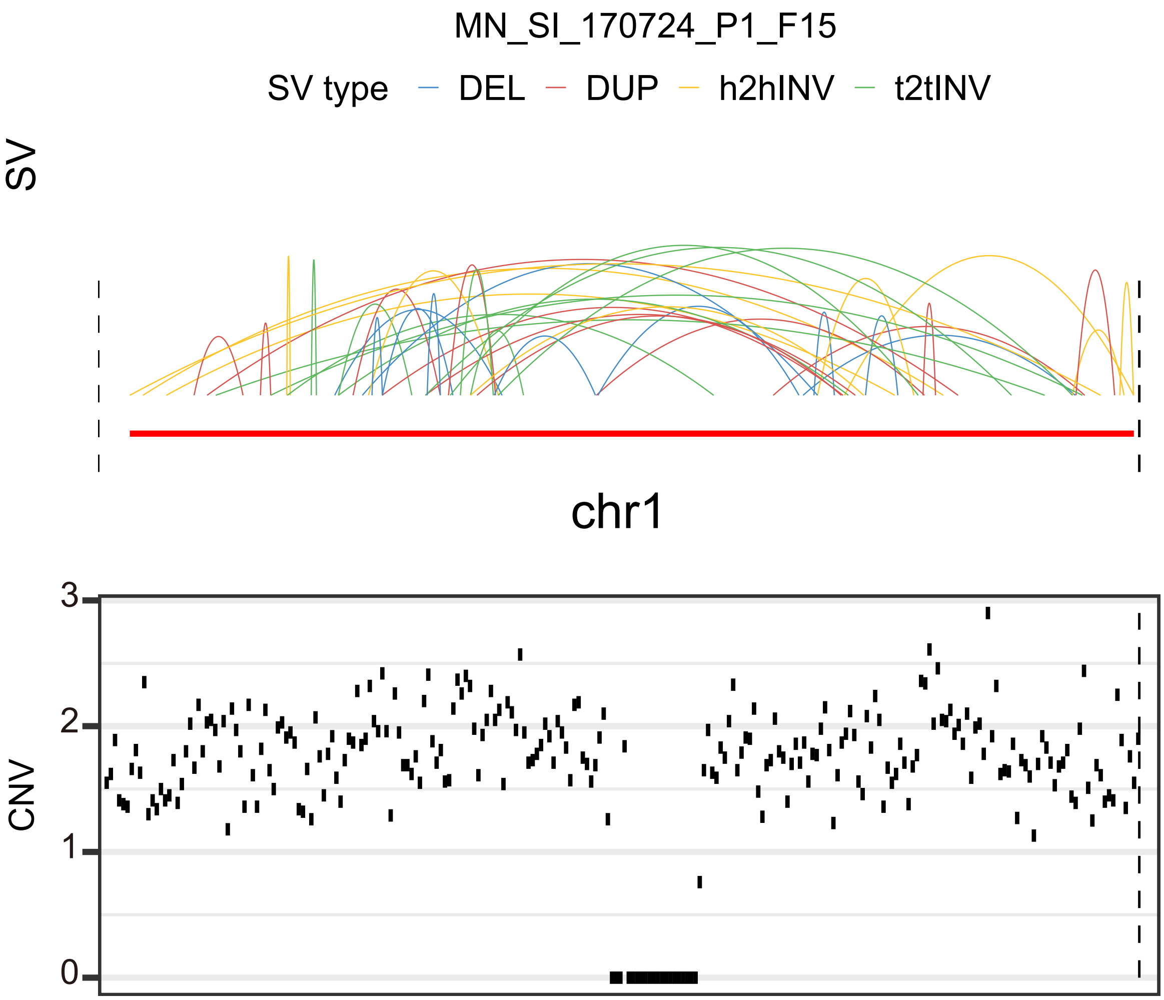

Figure 3: SVs and CNVs identified from single cell sequencing data with experimentally induced micronuclei. CN data is noisy with single cell sequencing.

“arm_del_rm” controls whether arm level deletions (at least 95% of one chromosome arm is lost) should be removed or not. To avoid the impact of aneuploidy on identifying telomere loss, arm level deletions are removed from bulk tumor sequencing data by default, since arm-level copy losses are common in cancer and independent of CGRs. However, in single cell sequencing of experimentally induced CGRs (Figure 4), all alterations are generated in one cell cycle including the somatic SVs and arm-level CNVs. This function should be turned off in that case.

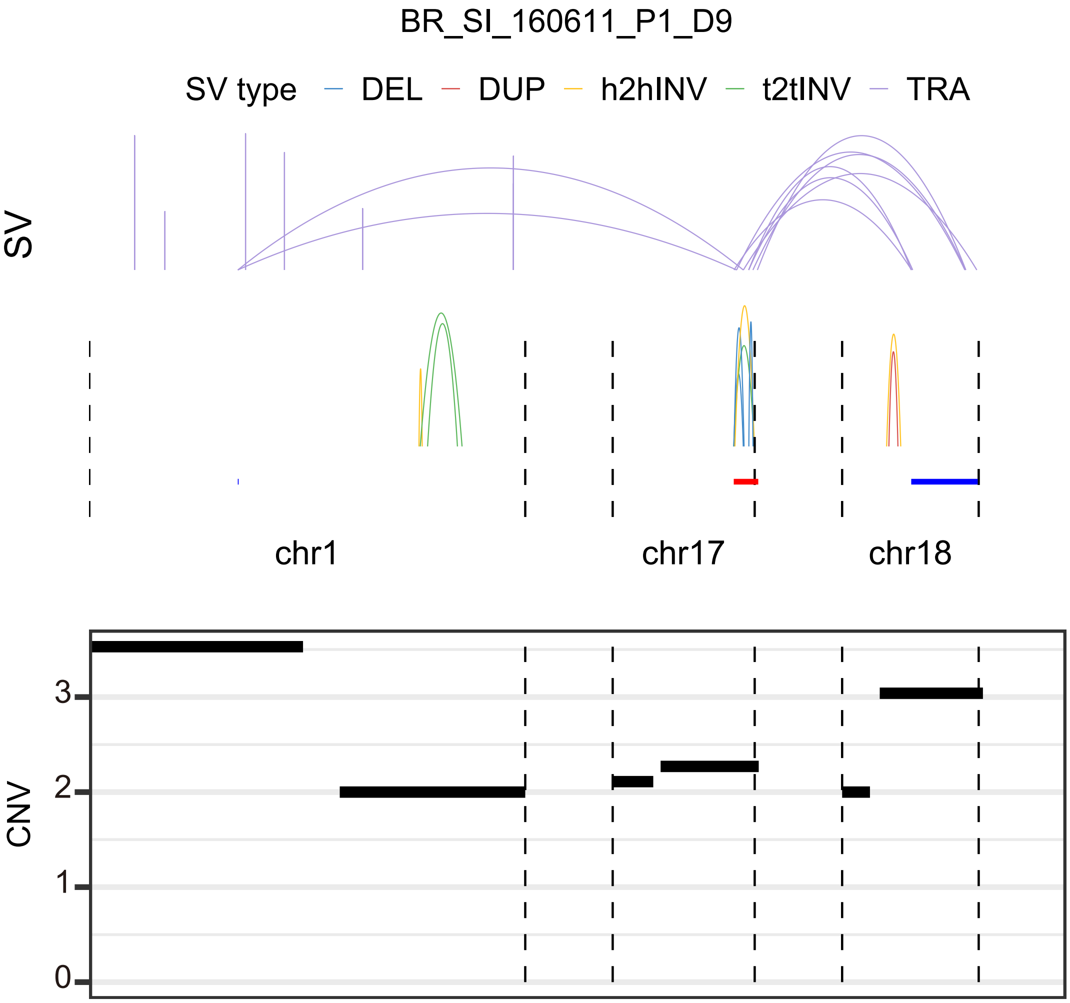

Figure 4: SVs and CNVs identified from single cell sequencing data with experimentally induced chromatin bridges. Induced chromatin bridge generates the telomere loss signature in the CGR region.

Example:

Starfish will output the progress of CGR feature computation.

```
starfish_feature_out = starfish_feature(starfish_link_out$starfish_call,
    starfish_link_out$interleave_tra_complex_sv, example_cnv, example_sample,
    prefix = "example", genome_v = "hg19", cnv_factor = "auto", arm_del_rm = TRUE)
```

```
## [1] "Starfish is computing the feature matrix, please be patient..."
## [1] "6% is done..."
## [1] "Event 07f16397-71bb-4594-ad4d-caa7d2baeabd_12"
## [1] "12% is done..."
## [1] "Event 07f16397-71bb-4594-ad4d-caa7d2baeabd_3"
## [1] "19% is done..."
## [1] "Event 07f16397-71bb-4594-ad4d-caa7d2baeabd_6"
## [1] "25% is done..."
## [1] "Event 07f16397-71bb-4594-ad4d-caa7d2baeabd_5"
## [1] "31% is done..."
## [1] "Event 07f16397-71bb-4594-ad4d-caa7d2baeabd_7"
## [1] "38% is done..."
## [1] "Event 07f16397-71bb-4594-ad4d-caa7d2baeabd_X"
## [1] "44% is done..."
## [1] "Event 18f5e75e-c623-11e3-bf01-24c6515278c0_4_19"
## [1] "50% is done..."
## [1] "Event 2b6d4d66-7f0b-4bc0-b3d6-171956a937c5_10"
## [1] "56% is done..."
## [1] "Event 2b6d4d66-7f0b-4bc0-b3d6-171956a937c5_18"
## [1] "62% is done..."
## [1] "Event 2b6d4d66-7f0b-4bc0-b3d6-171956a937c5_12"
## [1] "69% is done..."
## [1] "Event 2b6d4d66-7f0b-4bc0-b3d6-171956a937c5_6"
## [1] "75% is done..."
## [1] "Event 2b6d4d66-7f0b-4bc0-b3d6-171956a937c5_X"
## [1] "81% is done..."
## [1] "Event 3db6e6cc-1a06-49b9-834e-b6611cde4c4b_1_6_7_12_X"
## [1] "88% is done..."
## [1] "Event c00de7a0-0b09-4e07-988c-ef2a7f8e932a_5_13"
## [1] "94% is done..."
## [1] "Event fc8130df-897d-5404-e040-11ac0d485e0a_6_9_10_14_16_20"
## [1] "100% is done..."
## [1] "Event fc8130df-897d-5404-e040-11ac0d485e0a_2_3_7_17"
## [1] "CGR feature computing is done!"
```

Output:

The function **starfish_feature** returns an instance that contains a data frame: *cluster_feature*. *cluster_feature* is the feature matrix. A CSV file "example_CGR_feature_matrix.csv" will be generated under the working space:

```
print(head(starfish_feature_out$cluster_feature), row.names = FALSE)
```

```
##                                 sample                              cluster_id
##  07f16397-71bb-4594-ad4d-caa7d2baeabd 07f16397-71bb-4594-ad4d-caa7d2baeabd_12
##  07f16397-71bb-4594-ad4d-caa7d2baeabd  07f16397-71bb-4594-ad4d-caa7d2baeabd_3
##  07f16397-71bb-4594-ad4d-caa7d2baeabd  07f16397-71bb-4594-ad4d-caa7d2baeabd_6
##  07f16397-71bb-4594-ad4d-caa7d2baeabd  07f16397-71bb-4594-ad4d-caa7d2baeabd_5
##  07f16397-71bb-4594-ad4d-caa7d2baeabd  07f16397-71bb-4594-ad4d-caa7d2baeabd_7
##  07f16397-71bb-4594-ad4d-caa7d2baeabd  07f16397-71bb-4594-ad4d-caa7d2baeabd_X
##  max_CNV Loss_size_percentage Gain_size_percentage max_telo_loss_percentage
##        2          0.072315128                    0                        0
##        2          0.180587137                    0                        0
##        2          0.968516759                    0                        0
##        2          0.008745205                    0                        0
##        2          0.185858503                    0                        0
##       1         0.000000000                    0                        0
##  Brk_dispersion_MAD_mean_total
##                      1.4281075
##                      0.9620036
##                      1.7119816
##                      1.7628424
##                      1.0923239
##                      1.5943038
```

The columns are:

- "sample": sample ID
- "cluster_id": the unique CGR event ID
- "max_CNV": maximum copy number (log scale)
- "Loss_size_percentage": copy loss percentage
- "Gain_size_percentage": copy gain percentage
- "max_telo_loss_percentage": highest telomere loss percentage
- "Brk_dispersion_MAD_mean_total": breakpoint dispersion score

2.3.3 starfish_sig

This function loads the feature matrix and performs either signature classification (classifier) or *de novo* signature decomposition (clustering).

Usage:

```
starfish_sig(cluster_feature, prefix = "", cmethod = "class")
```

Input:

- "cluster_feature": feature matrix, which is the output from starfish_feature_out$cluster_feature.
- "prefix": the prefix for intermediate files, default is none.
- "cmethod": method to infer signatures from the CGR feature matrix. It could be either "class", pre-constructed classifier of PCAWG dataset, or "cluster", de novo unsupervised clustering. Default is "class".

Example 1:

If the user selects "class" method, Starfish will run the signature classification based on a pre-constructed classifier built upon 2428 tumors from PCAWG and output the progress:

```
starfish_sig_out = starfish_sig(starfish_feature_out$cluster_feature,
    prefix = "example", cmethod = "class")
```

```
## Using cluster_id as id variables
```

```
## [1] "Signature classification is done!"
```

Output 1:

The function **starfish_sig** returns a data frame which contains the normalized CGR feature values and the CGR signature classification. A CSV file "example_pcawg_6signatures_class.csv" will be generated under the working space:

```r
print(head(starfish_sig_out), row.names = FALSE)
```

```
##                                       cluster_id Brk_dispersion_MAD_mean_total
##  3db6e6cc-1a06-49b9-834e-b6611cde4c4b_1_6_7_12_X                     1.9878880
##        18f5e75e-c623-11e3-bf01-24c6515278c0_4_19                     0.1103537
##           07f16397-71bb-4594-ad4d-caa7d2baeabd_6                     2.9553173
##          2b6d4d66-7f0b-4bc0-b3d6-171956a937c5_10                     2.0866202
##          2b6d4d66-7f0b-4bc0-b3d6-171956a937c5_12                     1.7028798
##           07f16397-71bb-4594-ad4d-caa7d2baeabd_3                     0.3253912
##  Loss_size_percentage Gain_size_percentage log_max_CN max_telo_loss_percentage
##             0.0000000            0.8391539  3.7145040                0.0000000
##             0.0000000            2.6993327  1.7949709                4.0549723
##             3.5439186            0.0000000  0.5978452                0.0000000
##             3.3226341            0.0000000  0.5978452                0.0000000
##             3.0412921            0.0000000  0.5978452                0.5961708
##             0.6615674            0.0000000  0.5978452                0.0000000
##                    CGR_signature
##           1 ecDNA/double minutes
##  2 BFB cycles/chromatin bridge
##                     3 Large loss
##                     3 Large loss
##                     3 Large loss
##                    4 Micronuclei
```

The columns are:

- "cluster_id": the unique CGR event ID
- "log_max_CN": normalized maximum copy number (log scale)
- "Loss_size_percentage": normalized copy loss percentage
- "Gain_size_percentage": normalized copy gain percentage
- "max_telo_loss_percentage": normalized highest telomere loss percentage
- "Brk_dispersion_MAD_mean_total": normalized breakpoint dispersion score
- "CGR_signature": CGR signature classification

A signature plot will be generated for users to quickly check the proportion of 6 CGR signatures in the study cohort (Figure 5).

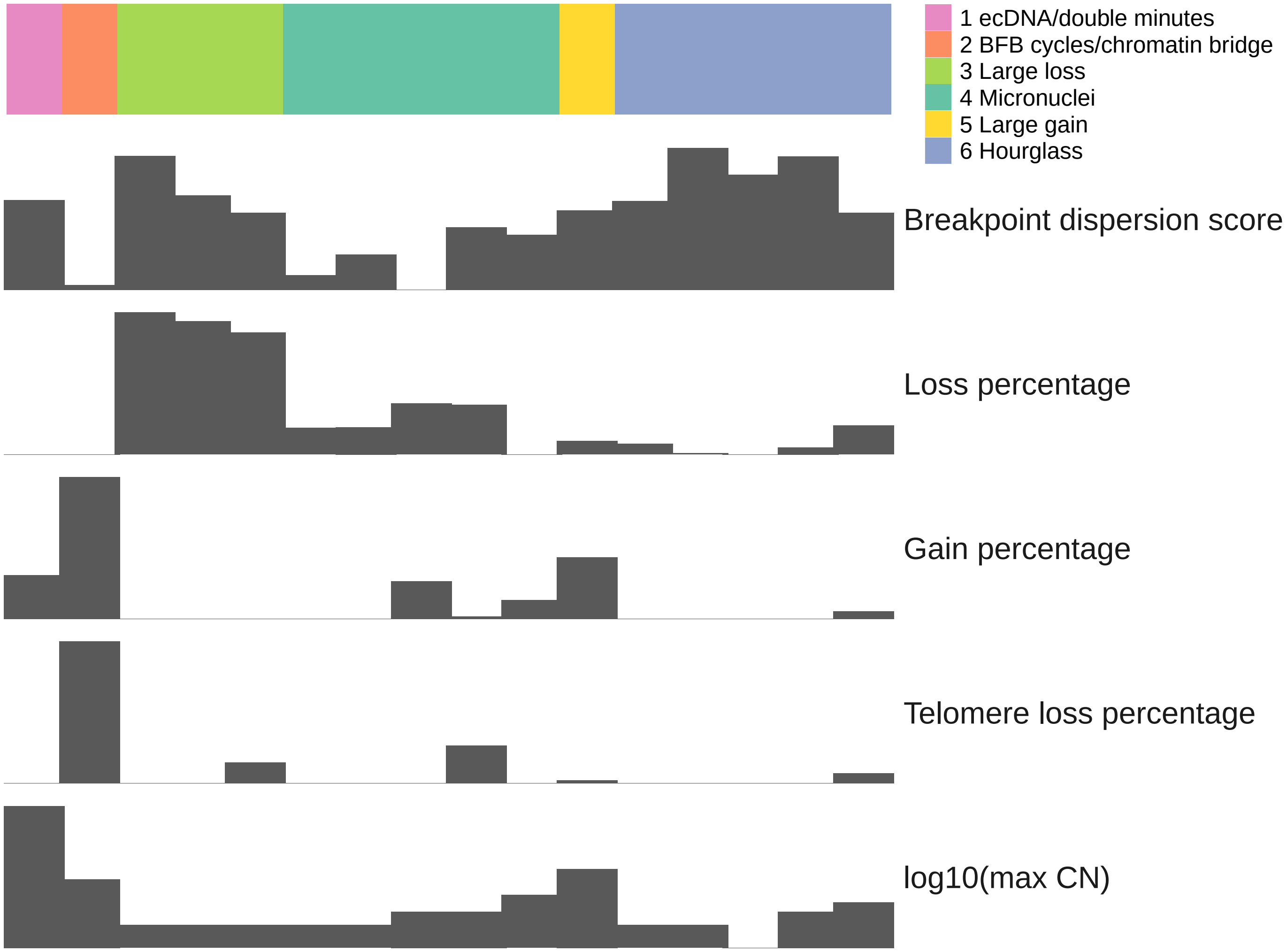

Figure 5: 6 CGR signatures with normalized feature values in the test data. Each column is a unique CGR event. Top panel shows the predicted signature of the event, and bottom panels show the normalized values of 5 features.

Example 2:

If the user selects “cluster” method, Starfish will invoke R package ConsensusClusterPlus[4] and run the unsupervised clustering for *de novo* signature decomposition and output the progress:

```r
starfish_sig(starfish_feature_out$cluster_feature, prefix = "example",
    cmethod = "cluster")
```

```
## end fraction

## clustered
## clustered
## clustered
## clustered
## clustered
## clustered
## clustered
## clustered
## clustered
## clustered
## clustered
## clustered

## [1] "Clustering is done! The clustering results are stored under 'CGR_cluster' folder!"
```

Output 2:

The clustering results are stored under "CGR_cluster" folder. Users could check either the "Delta Area" plot (Figure 6) or the "Consensus Cumulative Distribution Function (CDF) Plot" (Figure 7) to determine the optimal cluster number K:

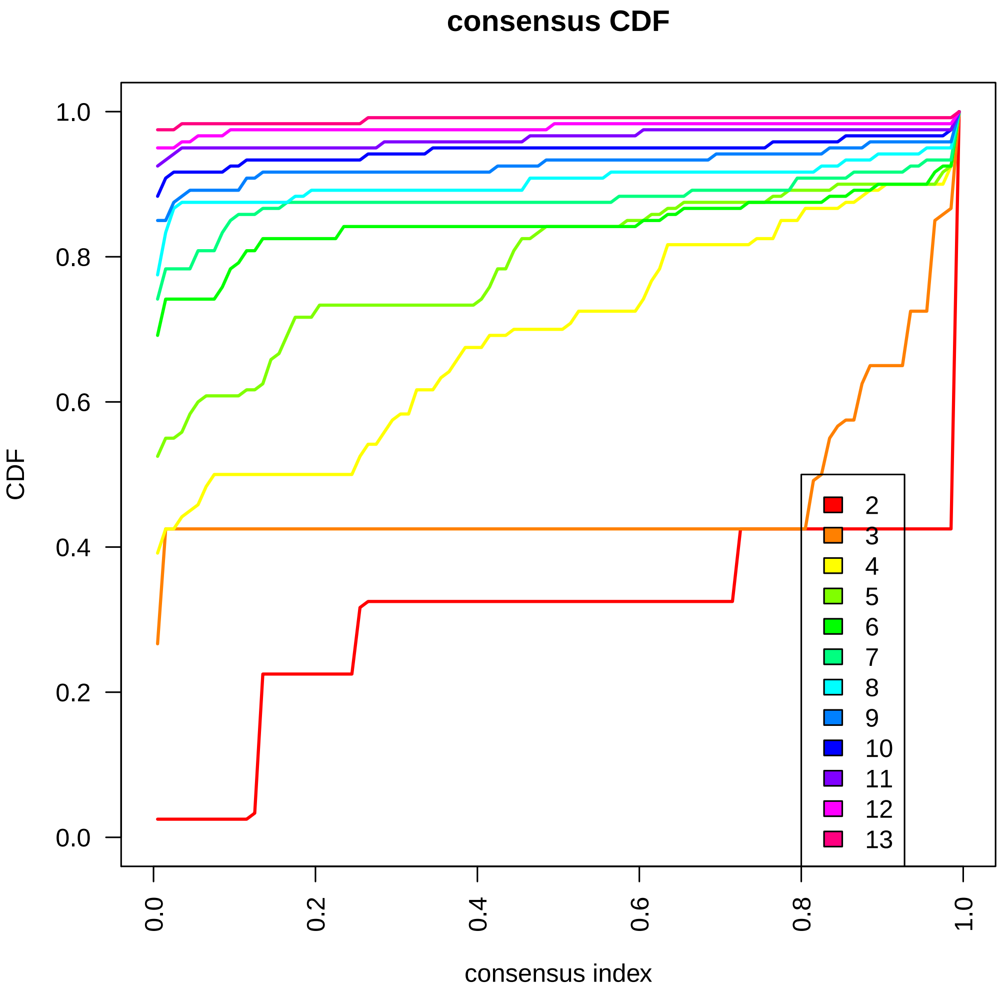

Figure 6: CDF plot for test data. This graphic shows the cumulative distribution functions of the consensus matrix for each cluster number K (indicated by colors). This figure allows the user to determine at what number of K, the CDF reaches an approximate maximum. The optimal K would be 6 as the curve approaches the plateau in this example.

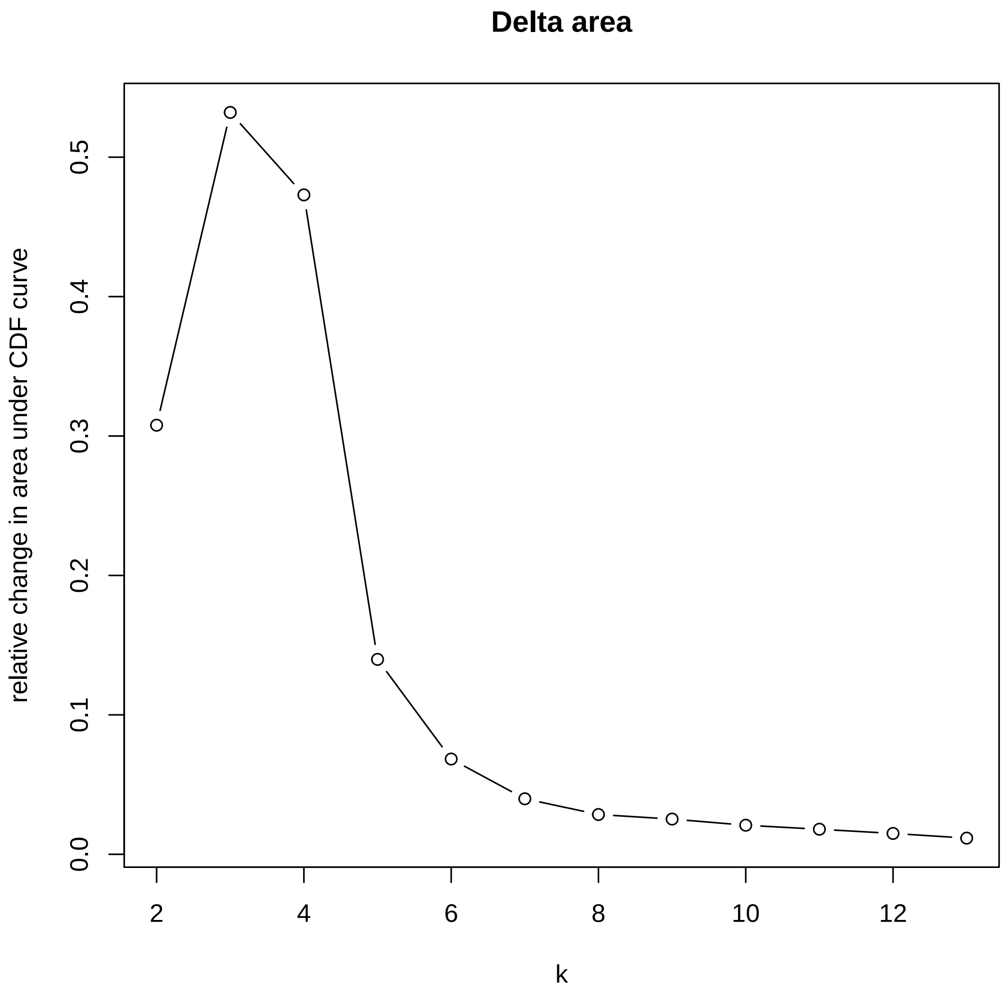

Figure 7: Delta Area plot for test data. This graphic shows the relative change in area under the CDF curve comparing cluster number K and K-1. This plot allows the user to determine the relative increase in consensus and determine K at which there is no appreciable increase. Beginning from cluster 6, there is no more dramatic increase, so 6 clusters would be optimal in this example.

The user can find clustering tables "CGR_cluster.k=K.consensusClass.csv" under the "CGR_cluster" folder, where different K represents different clustering numbers. Taking K=6 as an example, column "V1" is the unique CGR event ID, and column "V2" is the clustering ID:

```
starfish_sig_cluster = read.csv("CGR_cluster.k=6.consensusClass.csv",
    header = F)
print(head(starfish_sig_cluster), row.names = FALSE)
```

```
##                                       V1 V2
##  07f16397-71bb-4594-ad4d-caa7d2baeabd_12  1
##   07f16397-71bb-4594-ad4d-caa7d2baeabd_3  2
##   07f16397-71bb-4594-ad4d-caa7d2baeabd_6  3
##   07f16397-71bb-4594-ad4d-caa7d2baeabd_5  4
##   07f16397-71bb-4594-ad4d-caa7d2baeabd_7  2
##   07f16397-71bb-4594-ad4d-caa7d2baeabd_X  4
```

2.3.4 starfish_plot

This function loads an SV data frame, a CNV data frame and CGR regions reported by starfish_link, to draw the CGR regions in a linear setting.

```
starfish_plot(sv_file, cnv_file, cgr, genome_v = "hg19")
```

Input:

- "sv_file": the SV data frame defined previously
- "cnv_file": the CNV data frame defined previously
- "cgr": the output of starfish_link_out$starfish_call.
- "genome_v": the genome assembly version. It could be either "hg19" or "hg38", and default is "hg19".

Example:

```
starfish_plot(example_sv, example_cnv, starfish_link_out$starfish_call)
```

Output:

Linear plots of CGR regions (Figure 8).

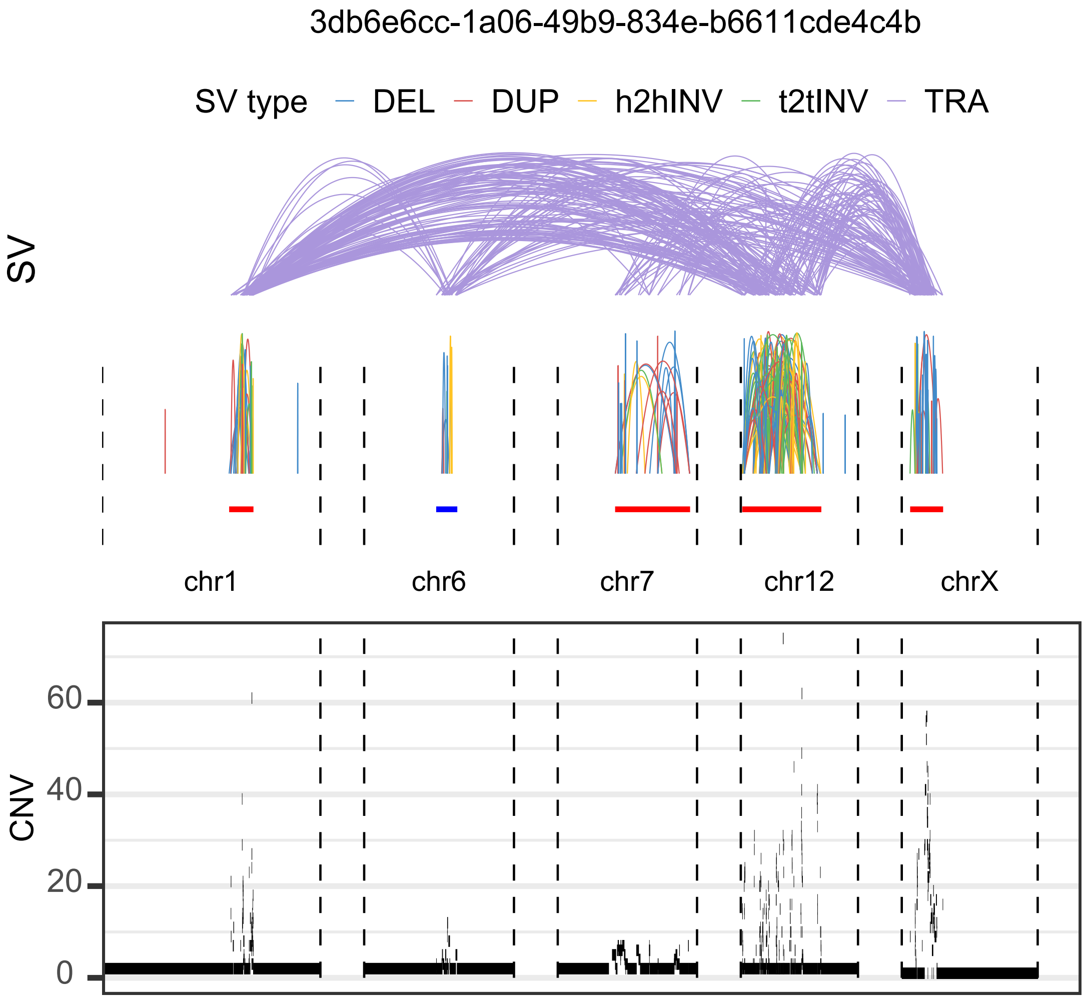

Figure 8: Example plot of SVs and CNVs in a CGR region involving chr1, 6, 7, 12 and X in tumor sample ”3db6e6cc-1a06-49b9-834e-b6611cde4c4b” with connected. ”seed” CGR regions are shown in red, and ”linked” CGR regions are shown in blue.

2.3.5 starfish_all

Starfish also provides a function “starfish_all” to run all previous four functions at once.

Usage:

```r
starfish_all(sv_file, cnv_file, gender_file, prefix = "", genome_v = "hg19",
    cnv_factor = "auto", arm_del_rm = TRUE, plot = TRUE, cmethod = "class")
```

Input:

- “sv_file”: the SV data frame defined previously
- “cnv_file”: the CNV data frame defined previously

- "gender_file": the gender data frame defined previously
- "prefix": the prefix for all intermediate files, default is none
- "genome_v": the genome assembly version. It should be "hg19" or "hg38", default is "hg19"
- "cnv_factor": the CN fluctuation beyond or below baseline to identify loss and gain fragments for samples with decimal CN, default is "auto", or users can provide a value between 0 and 1
- "arm_del_rm": logical value for whether or not arm level deletions should be removed, default is TRUE
- "plot": the logical value of plotting CGRs, default is TRUE.
- "cmethod": method to infer signatures from the CGR feature matrix, which can be "class" or "cluster", default is "class"

Example:

```r
starfish_all(example_sv, example_cnv, example_sample, prefix = "example",
    genome_v = "hg19", cnv_factor = "auto", arm_del_rm = TRUE, plot = TRUE,
    cmethod = "class")
```

3. Reference

1. Bao, L., Zhong, X., Yang, Y., Yang, L. Mutational signatures of complex genomic rearrangements in human cancer. https://www.biorxiv.org/content/10.1101/2021.05.16.444385v1
2. Cortés-Ciriano, I., Lee, J.JK., Xi, R. et al. Comprehensive analysis of chromothripsis in 2,658 human cancers using whole-genome sequencing. **Nat Genet** 52, 331–341 (2020).
3. Umbreit, N.T. et al. Mechanisms generating cancer genome complexity from a single cell division error. **Science** 368, (2020).
4. Wilkerson, D.M, Hayes, N.D. "ConsensusClusterPlus: a class discovery tool with confidence assessments and item tracking.". **Bioinformatics**, 26(12), 1572-1573 (2010).

4. Contact

- Lisui Bao, PhD
  Ben May Department for Cancer Research, The University of Chicago Chicago, IL 60637, USA
  Email: baolisui@ouc.edu.cn

- Lixing Yang, PhD
  Ben May Department for Cancer Research, The University of Chicago Chicago, IL 60637, USA
  Email: lixingyang@uchicago.edu