# Whole genome scaffolding using SaaRclust

***David Porubsky***

**2020-03-31**

**Package**

SaaRclust 0.99

## Contents

# 1 Introduction

Strand-seq is a single-cell sequencing technique able to preserve contiguity of individual parental homologues in single-cell (Falconer et al. 2012). Each parental homolog undergoes independent random segregation during cell division, leading to a unique strand state profile in Strand-seq data. Strand-seq distinguishes three possible template strand states for each chromosome of a diploid genome. The Watson-Watson (WW) strand state is characteristic of two Watson (reads aligned to minus strand) templates inherited from both parental homologues. The Crick-Crick (CC) strand state is characteristic of two Crick (reads aligned to plus strand) templates inherited from both parental homologues. Lastly, the Watson-Crick (WC) strand state is characteristic of a Watson and Crick template being inherited from either parental homologue (Sanders et al. 2017). Such Strand-seq signal can be used to assign contigs or long sequencing reads to a chromosome of origin. This feature has been shown to be valuable for scaffolding early build genome assemblies as well finding chimeric or misoriented contigs (Hills et al. 2013). We do so using SaaRclust, an R based package that implements a novel latent variable model and a corresponding Expectation Maximization (EM) algorithm in order to reliably cluster contigs or long sequencing reads by chromosome. SaaRclust was previously introduced for this in silico separation of long sequencing reads by chromosome and direction (Ghareghani et al. 2018). Here we have extended its functionalities to be able to scaffold contig stage assemblies and to detect and correct assembly error such as chimeric or misoriented contigs. SaaRclust employs an Expectation-Maximization (EM) soft clustering algorithm to handle the uncertainty arising from the sparse Strand-seq data. The main idea underlying our clustering algorithm is that contigs originating from the same chromosome share the same directionality pattern of aligned Strand-seq reads across multiple single cells, that differs from contigs originating from a different chromosome. The EM algorithm is based on iterating between assigning strand states for each Strand-seq library and chromosome and assigning chromosomes to each contig, which are both hidden information at the beginning. EM converges to a local optimum solution of the maximum likelihood problem, e.g., maximizing the likelihood of observed data (number of directional aligned Strand-seq reads to long reads), given the model parameters (strand states), and we have shown SaaRclust to be able to assign even individual long sequencing reads to chromosomes of origin.

# 2 Minimal parameters

Here are the minimal parameters required to successfully run genome scaffolding using SaaRclust.

**bamfolder:** A folder name where minimap file(s) is stored.
**outputfolder:** A folder containing BAM files with Strand-seq reads aligned to a denovo assembly.
**pairedEndReads:** Make sure to set to TRUE if paired-end reads are being used.
**assembly.fasta:** A denovo assembly FASTA file, if one want to export scaffolded denovo assembly in FASTA format.

For more details on available parameters please run.

```
library(SaaRclust)
?scaffoldDenovoAssembly
```

# 3 Quick Start

To demonstrate the functionality of SaaRclust we have have aligned Strand-seq from a human individual to GRCh38. Whole chromosomal scaffolds have been cut in to 5Mb long pieces and run through SaaRclust in order to assign them into clusters that in ideal scenario should reconstruct original whole-chromosome scaffolds. Here one can run SaaRclust using pre-computed binned counts of Strand-seq reads from 83 single-cell libraries aligned to the GRCh38. To speed up this process we sent bin size to 5 Mbp and included sequences/scaffolds of 10 Mbp and longer. Because Strand-seq data have been aligned to GRCh38 with chromosomes 1-22, X we have set 'desired.num.clusters' parameter to 23. We omit chromosome Y because of mapping ambiguity in PAR (pseudo-autosomal regions) regions.

Run the code below in order to test SaaRclust on example data that represent Strand-seq mappings to GRCh38.

```
bamfolder <- system.file("extdata", package = "SaaRclust")
scaffoldDenovoAssembly(bamfolder = bamfolder,
                       outputfolder = bamfolder,
                       store.data.obj = TRUE,
                       reuse.data.obj = TRUE,
                       pairedEndReads = TRUE,
                       bin.size = 5000000,
                       step.size = 5000000,
                       bin.method = 'dynamic',
                       prob.th = 0.25,
                       ord.method = 'greedy',
                       min.contig.size = 10000000,
                       concat.fasta = FALSE,
                       num.clusters = 100,
                       desired.num.clusters = 23,
                       min.region.to.order = 1000000,
                       remove.always.WC = TRUE,
```

```
                                mask.regions = FALSE)
```

# 4 Selected parameter descriptions

**min.contig.size:** This parameter should be set at least 2x the N50 read length used for the assembly in order to filter out singleton reads. Removing small contigs/reads will also speed-up the clustering process.
**min.region.to.order:** We recommned to set this parameter if one want to increase the ordering accuracy of contings within clusters (chromosomes). Small contigs tend to introduce a lot of noise in ordering process so we recommend to set this parameter to 500000 bp.
**num.clusters:** By default this parameter is to 100 in order to be able to capture small human chromosomes such as chromosome 21 or 22. We recommend to set this number even higher in case you expect large heterozygous inversions in your sample. This way one can ensure proper assignment of such inversions to correct cluster/chromosome.
**desired.num.clusters:** If an user has a prior knowledge on the expected number of chromosomes/clusters for a given sample it can be set by this parameter. If this parameter is not set, SaaRclust will still report a perdicted number of clusters/chromosomes based on shared strand inheritance.
**alpha:** Defines an expected level of noise in Strand-seq data. Increase this parameter in case Strand-seq data are of lower quality and contains a lot of background reads (see (Sanders et al. 2017) for more details on Strand-seq library quality).
**ord.method:** Tries to find traversal through the contigs that constitues a single cluster. This can be defined as 'Traveling salesman problem' and is implemented by using 'TSP' package and 'contiBAIT' (O'Neill et al. 2017). In order to achieve a reliable contig ordering we recommend to restrict ordering to contigs of size 500Kb and longer by setting a parameter 'min.region.to.order'.
**bin.method:** Defines the strategy used to split the genome assembly into chunks of a user defined size (see bin.size and step.size parameters). This paramter takes two values: either 'fixed' or 'dynamic'. In case of fixed binning the genome is divided into chunks of the same size. Dynamic binning checks the mappability using supplied Strand-data. In this case the 'bin.size' and 'step.size' parameters represents the number of mappable positions within each bin the instead of number of base-pairs. It means that each bin is dynamically extended in order to accomodate the defined number of mappable positions.

# 5 SaaRclust results

Saarclust results are organinzed in three subfolders: 'clustered_assembly', 'data' and 'plots'.
**clustered assembly:** This folder contains a table of all clustered contigs and their assignment to specific clusters stored as RData file. Also predicted assembly errors are reported here in a .tsv table. Lastly, this folder contains all exported FASTA files that contains contig sequences per cluster/chromosome.
**data:** This folder contains all intermediate results that are created during the clustering process. This files are created only if the parameter 'store.data.obj' is set to TRUE.
**plots:** Contains some useful plots that allow a user to evaluate how well genome scaffolding perfomed.

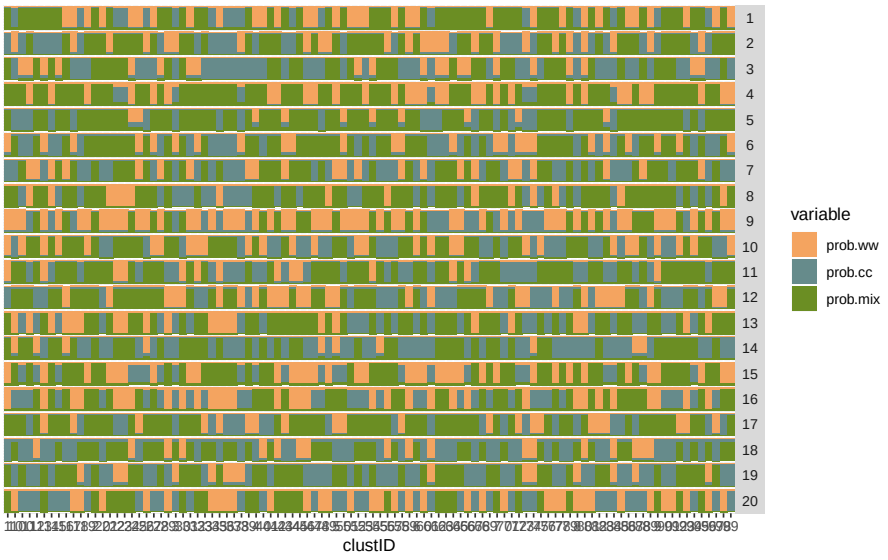
# 6 Data quality exploration

SaaRclust can produce a number of useful plots that can be helpful to assess how well the genome scaffoding performed.

```
## Load the SaaRclust package
library(SaaRclust)
```

Plot distribution of strand state probabilities per cluster (columns) and per single cell (rows). Given the random segregation of template strands into doughter cells (Falconer et al. 2012), one would expect approximatelly 50% of WC states and about 25% of WW and CC strand states per cluster (columns). However, in case of genomic regions where short-reads has difficulity to map we often observe a cluster where majority of single-cell has WC strand state. This cluster can often be removed by setting a parameter 'remove.always.WC' to TRUE.

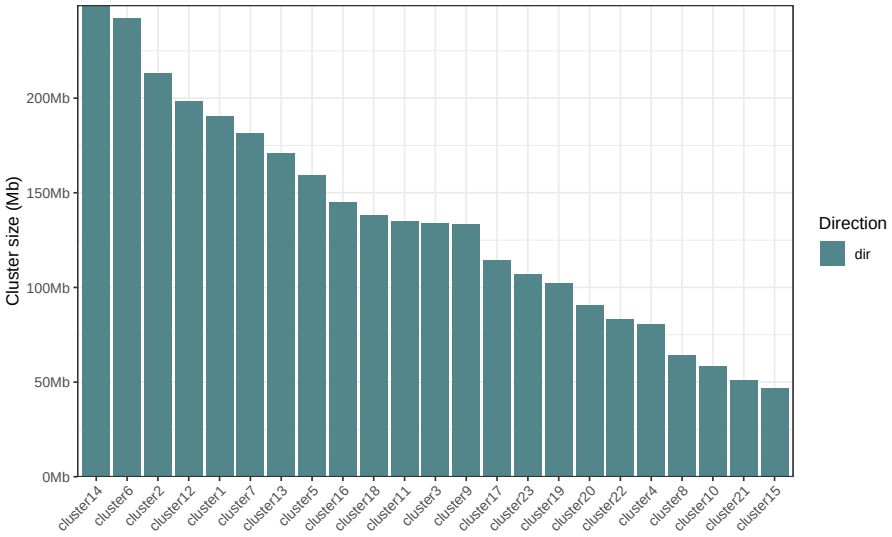
```
## Get example files
example.data <- system.file("extdata/data",
                            "softClust_100K_5e+06bp_dynamic.RData",
                            package = "SaaRclust")
EM.obj <- get(load(example.data))
## Plot theta parameter for 20 single cells
plotThetaEstimates(theta.param = EM.obj$theta.param[1:20])
```




Plot size of each cluster and the directionality of contigs within each cluster (dir - direct, revcomp - reverse complement). This plot useful in order to check to sum of contig sizes assigned to each cluster. In case there is a single cluster with lenght substantially bigger than the rest of the cluster suggest that this cluster is composed of more than two homologs. In this case it is advised to rerun SaaRclust after setting 'num.clusters' parameter to a higher number.

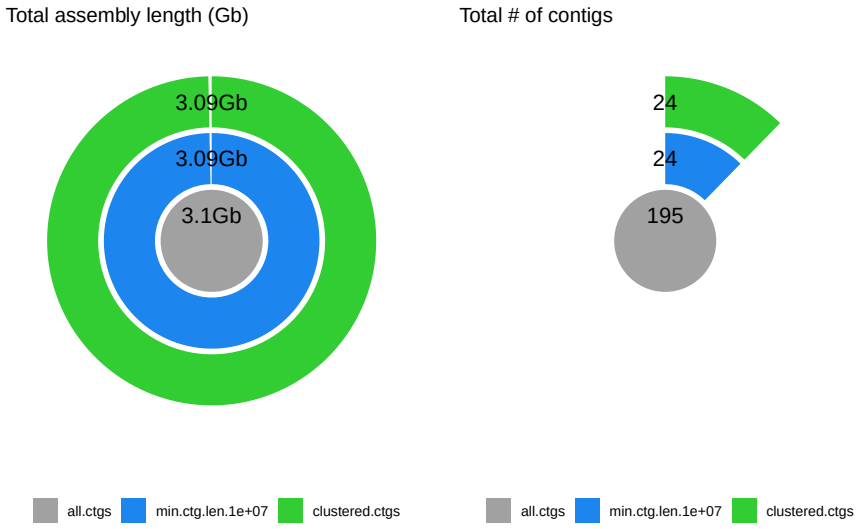
```
## Get example files
example.data <- system.file("extdata/clustered_assembly",
                            "ordered&oriented_5e+06bp_chunks.RData",
```

```
                              package = "SaaRclust")
ordered.contigs.gr <- get(load(example.data))
## Plot size and orientation of each cluster
plotClusteredContigSizes(clustered.gr = ordered.contigs.gr)
```




Plot summary statistics of successfully clustered contigs. In this plot one can see the total number of contigs (all.ctgs) present in the assembly, number of contigs after size selection set by 'min.contig.size' parameter (min.ctg.len) and number of successfully clustered contigs at the end of the analysis (clustered.ctgs). Remember that this number also depend on parameters such as 'prob.th' and 'min.region.to.order'.

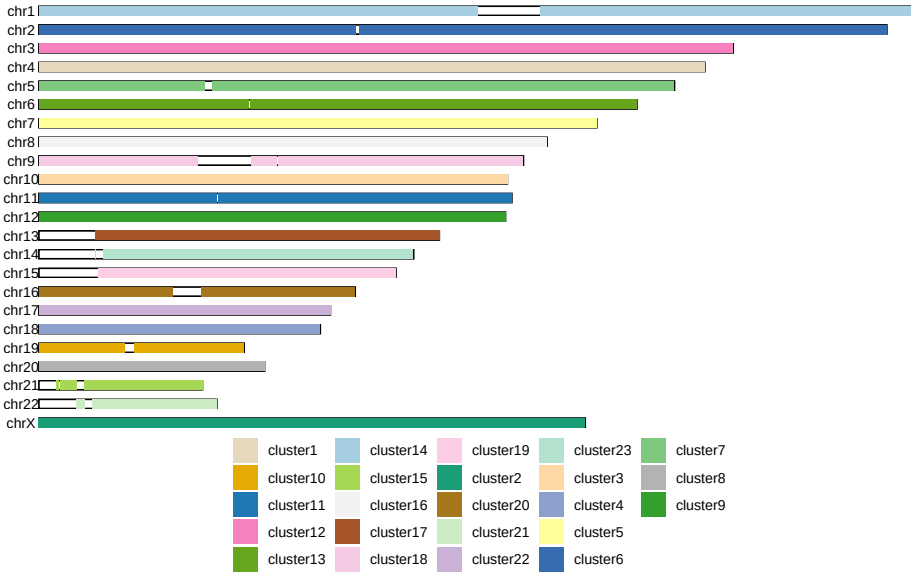
```
## Get example files
example.data <- system.file("extdata/clustered_assembly",
                            "ctgStat_minCtgSize_1e+07.RData",
                            package = "SaaRclust")
ctg.stat <- get(load(example.data))
## Plot statistics of clustered contigs
plotCTGstat(ctg.stat = ctg.stat)
```




Plot clustering accuracy of GRCh38 divided into 5Mb long genomics chunks. In an ideal scenario one will observe a single cluster (unique color) per chromosome. Clustering accuracy can be checked only for denovo assembled genome for which there is a high quality reference genome by mapping clustered contigs back to this reference.

```
## Load reference genome
library(BSgenome.Hsapiens.UCSC.hg38)
```

```
## Get example file
example.bed <- system.file("extdata/",
                           "SaaRclust_GRCh38_rescaffolded.bed",
                           package = "SaaRclust")
## Plot clustering accuracy
plotClusteredContigs(bedfile = example.bed,
                     min.mapq = 10,
                     chromosomes = paste0('chr', c(1:22,'X')),
                     bsgenome = BSgenome.Hsapiens.UCSC.hg38,
                     info.delim = '_',
                     info.fields = c('ctg', 'ord', 'id'),
                     col.by = 'id',
                     report = 'clustering')
```




# 7 General Recommendations

To speed up the clustering process we recommend to use fixed binning strategy instead of dynamic binning by setting a 'bin.method' parameter to 'fixed'.

## 7.1 Recommended settings to scaffold human genome

In general we have seen a very good performance of these parameters on multiple human genomes.

```
scaffoldDenovoAssembly(bamfolder = <bamfolder>,
                       outputfolder = SaaRclust_results,
                       store.data.obj = TRUE,
                       reuse.data.obj = TRUE,
                       pairedEndReads = TRUE,
                       bin.size = 200000,
                       step.size = 200000,
                       bin.method = 'dynamic',
                       prob.th = 0.25,
                       min.contig.size = 100000,
                       concat.fasta = TRUE,
                       num.clusters = 100,
                       desired.num.clusters = 24 - female; 25 - male,
                       remove.always.WC = TRUE,
                       mask.regions = FALSE)
```

## 7.2 Recommended settings to scaffold human genome with expected large SVs

In case one shall expect a large scale rearrangements in any given assembly we recommend to increase the number of initial clusters by setting 'num.clusters' parameter to a higher number. At the same time we do not recommend to set this number to a higher number than 200.

```
scaffoldDenovoAssembly(bamfolder = bamfolder,
                       outputfolder = bamfolder,
                       store.data.obj = TRUE,
                       reuse.data.obj = TRUE,
                       pairedEndReads = TRUE,
                       bin.size = 200000,
                       step.size = 200000,
                       bin.method = 'dynamic',
                       prob.th = 0.25,
                       min.contig.size = 100000,
                       concat.fasta = TRUE,
                       num.clusters = 150,
                       desired.num.clusters = 25,
                       remove.always.WC = TRUE,
                       mask.regions = FALSE)
```

## 7.3 Recommended settings to scaffold unknown (non-human) genome

When it comes to genome assemblies where there is no prior information about number chromosomes we have seen very good performance in reporting a reasonble number clusters that in many cases reflect the real number of chromosome in a given sample. To assure that the number of clusters reported by SaaRclust is reasonable we advise user to check the plot that visualize the size and contig orientation of each cluster.

```
scaffoldDenovoAssembly(bamfolder = bamfolder,
                       outputfolder = bamfolder,
                       store.data.obj = TRUE,
                       reuse.data.obj = TRUE,
                       pairedEndReads = TRUE,
                       bin.size = 200000,
                       step.size = 200000,
                       bin.method = 'dynamic',
                       prob.th = 0.25,
                       min.contig.size = 100000,
                       concat.fasta = TRUE,
                       num.clusters = 100,
                       remove.always.WC = TRUE,
                       mask.regions = FALSE)
```

# 8 Session Info

```
devtools::session_info()
## - Session info -------------------------------------------------------------
##  setting  value
##  version  R version 3.5.3 (2019-03-11)
##  os       Ubuntu 18.04.2 LTS
##  system   x86_64, linux-gnu
##  ui       X11
##  language (EN)
##  collate  en_US.UTF-8
##  ctype    en_US.UTF-8
##  tz       America/Los_Angeles
##  date     2020-03-31
##
## - Packages -----------------------------------------------------------------
##  package                     * version   date       lib source
##  assertthat                    0.2.1     2019-03-21 [1] CRAN (R 3.5.3)
##  backports                     1.1.4     2019-04-10 [1] CRAN (R 3.5.3)
##  bamsignals                    1.14.0    2018-10-30 [1] Bioconductor
##  BH                          * 1.69.0-1  2019-01-07 [1] CRAN (R 3.5.3)
##  Biobase                       2.42.0    2018-10-30 [1] Bioconductor
##  BiocGenerics                * 0.28.0    2018-10-30 [1] Bioconductor
##  BiocManager                   1.30.4    2018-11-13 [1] CRAN (R 3.5.3)
##  BiocParallel                  1.16.6    2019-02-10 [1] Bioconductor
##  BiocStyle                   * 2.10.0    2018-10-30 [1] Bioconductor
##  Biostrings                  * 2.50.2    2019-01-03 [1] Bioconductor
##  bitops                        1.0-6     2013-08-17 [1] CRAN (R 3.5.3)
##  bookdown                      0.12      2019-07-11 [1] CRAN (R 3.5.3)
##  BSgenome                    * 1.50.0    2018-10-30 [1] Bioconductor
##  BSgenome.Hsapiens.UCSC.hg38 * 1.4.1     2019-04-10 [1] Bioconductor
##  callr                         3.3.2     2019-09-22 [1] CRAN (R 3.5.3)
##  cli                           1.1.0     2019-03-19 [1] CRAN (R 3.5.3)
##  cluster                       2.0.7-1   2018-04-09 [4] CRAN (R 3.5.0)
##  codetools                     0.2-16    2018-12-24 [4] CRAN (R 3.5.2)
##  colorspace                    1.4-1     2019-03-18 [1] CRAN (R 3.5.3)
##  cowplot                     * 1.0.0     2019-07-11 [1] CRAN (R 3.5.3)
##  crayon                        1.3.4     2017-09-16 [1] CRAN (R 3.5.3)
##  data.table                    1.12.8    2019-12-09 [1] CRAN (R 3.5.3)
##  DelayedArray                  0.8.0     2018-10-30 [1] Bioconductor
##  desc                          1.2.0     2018-05-01 [1] CRAN (R 3.5.3)
##  devtools                      2.1.0     2019-07-06 [1] CRAN (R 3.5.3)
##  digest                        0.6.21    2019-09-20 [1] CRAN (R 3.5.3)
##  dplyr                         0.8.3     2019-07-04 [1] CRAN (R 3.5.3)
##  ellipsis                      0.3.0     2019-09-20 [1] CRAN (R 3.5.3)
##  evaluate                      0.14      2019-05-28 [1] CRAN (R 3.5.3)
##  foreach                       1.4.4     2017-12-12 [1] CRAN (R 3.5.3)
##  fs                            1.3.1     2019-05-06 [1] CRAN (R 3.5.3)
##  GenomeInfoDb                * 1.18.2    2019-02-12 [1] Bioconductor
##  GenomeInfoDbData              1.2.0     2019-04-10 [1] Bioconductor
```

```
##  GenomicAlignments            1.18.1    2019-01-04 [1] Bioconductor
##  GenomicRanges              * 1.34.0    2018-10-30 [1] Bioconductor
##  ggplot2                    * 3.2.1     2019-08-10 [1] CRAN (R 3.5.3)
##  glue                         1.3.1     2019-03-12 [1] CRAN (R 3.5.3)
##  gtable                       0.3.0     2019-03-25 [1] CRAN (R 3.5.3)
##  htmltools                    0.4.0     2019-10-04 [1] CRAN (R 3.5.3)
##  igraph                       1.2.4.1   2019-04-22 [1] CRAN (R 3.5.3)
##  IRanges                    * 2.16.0    2018-10-30 [1] Bioconductor
##  iterators                    1.0.10    2018-07-13 [1] CRAN (R 3.5.3)
##  knitr                        1.25      2019-09-18 [1] CRAN (R 3.5.3)
##  labeling                     0.3       2014-08-23 [1] CRAN (R 3.5.3)
##  lattice                      0.20-38   2018-11-04 [4] CRAN (R 3.5.1)
##  lazyeval                     0.2.2     2019-03-15 [1] CRAN (R 3.5.3)
##  lifecycle                    0.1.0     2019-08-01 [1] CRAN (R 3.5.3)
##  lpSolve                      5.6.13.3  2019-08-19 [1] CRAN (R 3.5.3)
##  magrittr                     1.5       2014-11-22 [1] CRAN (R 3.5.3)
##  Matrix                       1.2-16    2019-03-08 [4] CRAN (R 3.5.2)
##  matrixStats                  0.54.0    2018-07-23 [1] CRAN (R 3.5.3)
##  memoise                      1.1.0     2017-04-21 [1] CRAN (R 3.5.3)
##  munsell                      0.5.0     2018-06-12 [1] CRAN (R 3.5.3)
##  pillar                       1.4.2     2019-06-29 [1] CRAN (R 3.5.3)
##  pkgbuild                     1.0.3     2019-03-20 [1] CRAN (R 3.5.3)
##  pkgconfig                    2.0.3     2019-09-22 [1] CRAN (R 3.5.3)
##  pkgload                      1.0.2     2018-10-29 [1] CRAN (R 3.5.3)
##  plyr                         1.8.4     2016-06-08 [1] CRAN (R 3.5.3)
##  prettyunits                  1.0.2     2015-07-13 [1] CRAN (R 3.5.3)
##  processx                     3.4.1     2019-07-18 [1] CRAN (R 3.5.3)
##  ps                           1.3.0     2018-12-21 [1] CRAN (R 3.5.3)
##  purrr                        0.3.2     2019-03-15 [1] CRAN (R 3.5.3)
##  R6                           2.4.0     2019-02-14 [1] CRAN (R 3.5.3)
##  RColorBrewer                 1.1-2     2014-12-07 [1] CRAN (R 3.5.3)
##  Rcpp                         1.0.2     2019-07-25 [1] CRAN (R 3.5.3)
##  RCurl                        1.95-4.12 2019-03-04 [1] CRAN (R 3.5.3)
##  remotes                      2.1.0     2019-06-24 [1] CRAN (R 3.5.3)
##  reshape2                     1.4.3     2017-12-11 [1] CRAN (R 3.5.3)
##  rlang                        0.4.0     2019-06-25 [1] CRAN (R 3.5.3)
##  rmarkdown                    1.16      2019-10-01 [1] CRAN (R 3.5.3)
##  rprojroot                    1.3-2     2018-01-03 [1] CRAN (R 3.5.3)
##  Rsamtools                    1.34.1    2019-01-31 [1] Bioconductor
##  rtracklayer                * 1.42.2    2019-03-01 [1] Bioconductor
##  S4Vectors                  * 0.20.1    2018-11-09 [1] Bioconductor
##  SaaRclust                  * 0.99      2020-04-01 [1] local
##  scales                       1.0.0     2018-08-09 [1] CRAN (R 3.5.3)
##  sessioninfo                  1.1.1     2018-11-05 [1] CRAN (R 3.5.3)
##  stringi                      1.4.3     2019-03-12 [1] CRAN (R 3.5.3)
##  stringr                      1.4.0     2019-02-10 [1] CRAN (R 3.5.3)
##  SummarizedExperiment         1.12.0    2018-10-30 [1] Bioconductor
##  testthat                     2.1.1     2019-04-23 [1] CRAN (R 3.5.3)
##  tibble                       2.1.3     2019-06-06 [1] CRAN (R 3.5.3)
##  tidyr                        1.0.0     2019-09-11 [1] CRAN (R 3.5.3)
##  tidyselect                   0.2.5     2018-10-11 [1] CRAN (R 3.5.3)
```

```
##  TSP                           1.1-7     2019-05-22 [1] CRAN (R 3.5.3)
##  usethis                       1.5.1     2019-07-04 [1] CRAN (R 3.5.3)
##  vctrs                         0.2.0     2019-07-05 [1] CRAN (R 3.5.3)
##  withr                         2.1.2     2018-03-15 [1] CRAN (R 3.5.3)
##  xfun                          0.10      2019-10-01 [1] CRAN (R 3.5.3)
##  XML                           3.98-1.20 2019-06-06 [1] CRAN (R 3.5.3)
##  XVector                     * 0.22.0    2018-10-30 [1] Bioconductor
##  yaml                          2.2.0     2018-07-25 [1] CRAN (R 3.5.3)
##  zeallot                       0.1.0     2018-01-28 [1] CRAN (R 3.5.3)
##  zlibbioc                      1.28.0    2018-10-30 [1] Bioconductor
##
## [1] /home/porubsky/R/x86_64-pc-linux-gnu-library/3.5
## [2] /usr/local/lib/R/site-library
## [3] /usr/lib/R/site-library
## [4] /usr/lib/R/library
```

Report any issues here:

# References

Falconer, Ester, Mark Hills, Ulrike Naumann, Steven S S Poon, Elizabeth A Chavez, Ashley D Sanders, Yongjun Zhao, Martin Hirst, and Peter M Lansdorp. 2012. "DNA Template Strand Sequencing of Single-Cells Maps Genomic Rearrangements at High Resolution." *Nat. Methods* 9 (11): 1107–12.

Ghareghani, Maryam, David Porubsky, Ashley D Sanders, Sascha Meiers, Evan E Eichler, Jan O Korbel, and Tobias Marschall. 2018. "Strand-Seq Enables Reliable Separation of Long Reads by Chromosome via Expectation Maximization." *Bioinformatics* 34 (13): i115–i123.

Hills, Mark, Kieran O'Neill, Ester Falconer, Ryan Brinkman, and Peter M Lansdorp. 2013. "BAIT: Organizing Genomes and Mapping Rearrangements in Single Cells." *Genome Med.* 5 (9): 82.

O'Neill, Kieran, Mark Hills, Mike Gottlieb, Matthew Borkowski, Aly Karsan, and Peter M Lansdorp. 2017. "Assembling Draft Genomes Using contiBAIT." *Bioinformatics* 33 (17): 2737–9.

Sanders, Ashley D, Ester Falconer, Mark Hills, Diana C J Spierings, and Peter M Lansdorp. 2017. "Single-Cell Template Strand Sequencing by Strand-Seq Enables the Characterization of Individual Homologs." *Nat. Protoc.* 12 (6): 1151–76.