Basic4CSim: Simulation of 4C-seq data

Carolin Walter

March 11, 2019

Contents

1 Introduction

Chromosome conformation capture combined with high-throughput sequencing (4C-seq) is a next-generation sequencing (NGS) based method to identify threedimensional contacts between a chosen viewpoint, and the remaining genome [van de Werken *et al.* A, 2012]. 4C-seq data is fragment-based and suffers from biases like blind or non-blind fragments (cp.[van de Werken *et al.* B, 2012]); simulation of this data type is therefore not trivial.

Basic4CSim can create virtual simulation fragment libraries, allows to create a characteristic background distribution and different noise patterns for simulated 4C-seq data, and offers the functionality to add different peak forms to simulated 4C-seq samples. The virtual fragment library is adapted from *Basic4Cseq* (cp. [Walter *et al.*, 2014]). Simulated 4C-seq sample sequences can then be exported as fastq files.

1.1 Loading the package

After installation, the package can be loaded into R by typing

```
> library(Basic4CSim)
```

into the R console.

1.2 Provided functionality

Basic4Cseq requires the R-packages *GenomicAlignments*, *Biostrings*, *caTools*, and *GenomicRanges*. The package *BSgenome.Hsapiens.UCSC.hg19* or any respective corresponding BSgenome package (e.g. for Mus musculus) is suggested for the creation of the virtual simulation library.

This package provides the following basic functions for the simulation of 4C-seq data:

- `createVirtualFragmentLibrarySimulation`: creates virtual fragment libraries for BSgenome packages, with read sequences per fragment end
- `addBackground`: adds power-law distribution based background noise to the simulated sample
- `adaptBackground`: reduces read count for blind fragments, and adjusts the background noise rate to specific desired viewpoint fragment coverage rates
- `addPeaks`: adds peak regions with specified mean and standard deviation ("width") to the sample
- `makeRandomPeaks`: prepares a set of random peaks within predefined intervals for position, mean, and standard deviation
- `simulateSample`: wrapper function for the simulation of a basic 4C-seq sample with background, low noise, and peak regions
- `printFastq`: export function for simulated 4C-seq data to fastq format

In addition to the examples presented in this vignette, more detailed information on the function parameters and additional examples are presented in the corresponding manual pages.

2 Simulation of 4C-seq data

Basic4CSim is based on virtual fragment libraries similar to those used in *Basic4Cseq*. A virtual simulation fragment library specifies the position and properties of 4C-seq fragments, i.e. genomic intervals defined by the chosen first restriction enzyme sequence, with added sequence information for the raw fragments.

For each simulated sample, one empty fragment count table is generated from the virtual simulation fragment library, which specifies the simulated signal or read count per 4C-seq fragment end. Initially, each fragment end has 0 allocated reads; 4C-seq background and signal are subsequently added as needed. As soon as the fragment count table is finished, corresponding fastq files with the respective fragment sequences can be exported.

2.1 Creation of a fragment library

Basic4CSim's data simulations start with the creation of a fragment library. As with *Basic4Cseq*'s fragment libraries, the virtual simulation fragment libraries can be stored and reused for simulations with the same reference genome and restriction enzyme combinations. Creating a library for a 4bp+4bp experiment setting for a full human or mouse reference genome usually takes some hours, hence we demonstrate the function on a short example DNA string object:

```
> testGenome = DNAString("TCATGAAAGATCGGGGCATGTT")
> fragmentData = createVirtualFragmentLibrarySimulation(
     chosenGenome = testGenome, firstCutter = "catg",
     secondCutter = "gatc", readLength = 3,  chromosomeName =
```

```
    "test", libraryName = "")
> head(fragmentData)

  chromosomeName fragmentStart fragmentEnd fragmentCentre isNonBlind fragmentLength
2           test             6          16             11       TRUE             11
  leftFragEndLength rightFragEndLength leftFragEndValid rightFragEndValid leftSequence
2                 3                 4             TRUE              TRUE          AAA
  rightSequence
2           GGG
```

2.2 Creating a set of random peaks

Basic4CSim's default peaks follow the shape of a normal distribution ("bell-shaped"), with a certain position, maximum height and standard deviation. Peak regions can be set to specific values, or created randomly within certain preset constraints.

```
> vpStart = 69999869
> makeRandomPeaks(peakNumber=6, vpStart)

  ID     mean     max     sd
1  1 70055262 2624.07 788.52
2  2 70082099 1823.63 768.74
3  3 69926623 2661.48 889.69
4  4 69949298 1926.08 907.64
5  5 70098393 2377.07 655.09
6  6 69981095 2485.28 878.51
```

2.3 Background noise

While there is a high concentration of signal at or close to the 4C-seq experiment's viewpoint ("near-cis"), a certain level of background noise is also expected for the remaining ("cis") part of the viewpoint chromosome. This noise can be added with `addBackground`. Since the amount of fragments which are actually covered is typically far lower in cis than in near-cis, different coverage rates can be set to reflect this. The simulated background noise fragments' general signal strength follows a power-law distribution, as a similar distribution can usually be observed for real-world data ([Thongjuea *et al.*, 2013]). A simulated example near-cis background is visualized in Figure 1.

```
> simLibFile <- system.file("extdata", "simLib_short.csv",
    package="Basic4CSim")
> # load virtual simulation fragment library
> simLib = read.csv(simLibFile, sep = "\t", header = TRUE)
> vpStart = 69999869
> # create empty simulation fragment count table
> simTable = data.frame(simLib[,1:3], "readsL" = 0, "readsR" = 0,
    simLib[,c(5,9,10)])
> colnames(simTable) = c("chromosomeName", "start", "end",
    "readsL", "readsR", "isNonBlind", "leftFragEndValid", "rightFragEndValid")
> # add actual background
> set.seed(42)
> simTable = addBackground(simTable, vpStart)
> head(simTable)
```

```
  chromosomeName    start      end readsL readsR isNonBlind leftFragEndValid
1             10 69900252 69900422      5    157      FALSE             TRUE
2             10 69900427 69900454      5      1      FALSE             TRUE
3             10 69900459 69900777     13     44       TRUE             TRUE
4             10 69900782 69900783      0      5      FALSE            FALSE
5             10 69900788 69900862     47      3      FALSE             TRUE
6             10 69900867 69900940      0     82       TRUE            FALSE
  rightFragEndValid
1              TRUE
2              TRUE
3              TRUE
4             FALSE
5              TRUE
6             FALSE
```

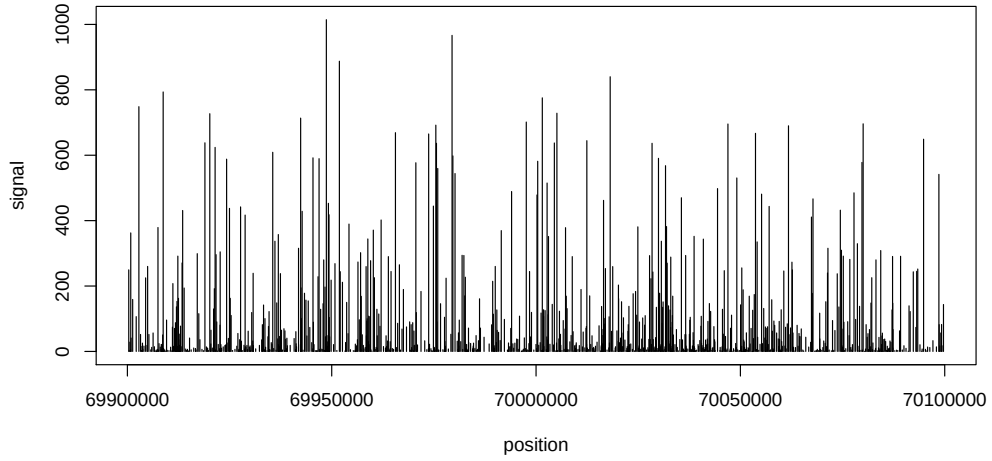

Figure 1: Simulated raw background noise in near-cis, with a chosen fragment coverage rate of 0.8. The coverage rate and signal strength can be customized to account for cis/near-cis regions and different sample characteristics.

2.4 Adding peaks

After simulating background noise, intervals with increased signal strength ("peaks") can be added to the fragment table with `addPeaks`. Peaks from `makeRandomPeaks` can be exported and used to simulate 4C-seq replicate data with different properties. In addition, smaller, randomly created noise peaks can be introduced as well (`addRandomPeaks`). These noise peaks vary for each sample, unless a fixed seed is used in R. These raw peaks do not contain fragment-specific noise or adjustments for biases, e.g. penalties for read counts on fragments without a secondary restriction enzyme site ("blind fragments"), which are expected to yield less reads than non-blind fragments. An example for a near-cis region with simulated background and additional raw peaks is shown in Figure 2.

```
> simTableFile <- system.file("extdata", "simTable_bg.csv",
     package="Basic4CSim")
```

```
> simTable = read.csv(simTableFile, sep = "\t", header = TRUE)
> vpStart = 69999869
> set.seed(42)
> randomPeaks = makeRandomPeaks(peakNumber=6, vpStart)
> simTable = addPeaks(simTable, randomPeaks, vpStart)
> head(simTable)

  chromosomeName    start      end readsL readsR isNonBlind leftFragEndValid
1             10 69900252 69900422      5    157      FALSE             TRUE
2             10 69900427 69900454      5      1      FALSE             TRUE
3             10 69900459 69900777     13     44       TRUE             TRUE
4             10 69900782 69900783      0      5      FALSE            FALSE
5             10 69900788 69900862     47      3      FALSE             TRUE
6             10 69900867 69900940      0     82       TRUE            FALSE
  rightFragEndValid
1              TRUE
2              TRUE
3              TRUE
4             FALSE
5              TRUE
6             FALSE

> simTableFile <- system.file("extdata", "simTable_peaks.csv",
    package="Basic4CSim")
> simTable = read.csv(simTableFile, sep = "\t", header = TRUE)
> vpStart = 69999869
> set.seed(42)
> simTable = addRandomPeaks(simTable, vpStart)
> head(simTable)

  chromosomeName    start      end readsL readsR isNonBlind leftFragEndValid
1             10 69900252 69900422      5    157      FALSE             TRUE
2             10 69900427 69900454      5      1      FALSE             TRUE
3             10 69900459 69900777     13     44       TRUE             TRUE
4             10 69900782 69900783      0      5      FALSE            FALSE
5             10 69900788 69900862     47      3      FALSE             TRUE
6             10 69900867 69900940      0     82       TRUE            FALSE
  rightFragEndValid
1              TRUE
2              TRUE
3              TRUE
4             FALSE
5              TRUE
6             FALSE
```

2.5 Adapting the near-cis region

With a certain amount of background noise and higher 4C-seq signal in certain intervals, the simulated near-cis region still lacks noise per fragment, and a modification that induces increasing loss of signal with increasing distance from the viewpoint. Furthermore, blind fragments without a secondary restriction enzyme site are expected to yield less reads than non-blind fragments. The function `adaptBackground`

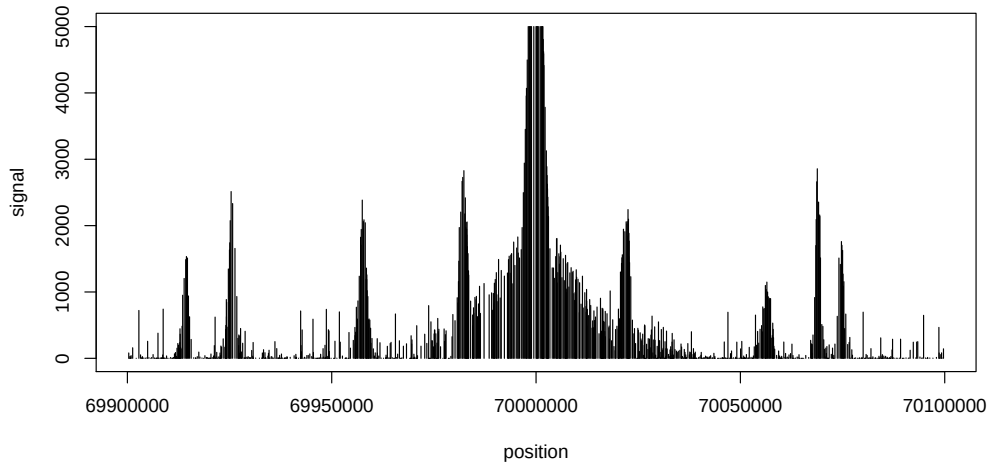

Figure 2: Simulated background, with added peaks, noise, and viewpoint. Peak files can be used for different simulated samples to create simulated replicates. Adapting the profile to account for blind fragments, distance from the viewpoint, and random fragment effects is still necessary, though.

can be used to model and customize these effects for a simulated 4C-seq sample, its effects for the example viewpoint are shown in Figure 3.

```
> simTableFile <- system.file("extdata", "simTable_noise.csv",
    package="Basic4CSim")
> simTable = read.csv(simTableFile, sep = "\t", header = TRUE)
> vpStart = 69999869
> set.seed(42)
> simTable = adaptBackground(simTable, vpStart, useNormDist = TRUE,
    vpRegionDist = 150000)
> head(simTable)
```

```
  chromosomeName    start      end readsL readsR isNonBlind leftFragEndValid
1             10 69900252 69900422      1     31      FALSE             TRUE
2             10 69900427 69900454      1      0      FALSE             TRUE
3             10 69900459 69900777     13     44       TRUE             TRUE
4             10 69900782 69900783      0      1      FALSE            FALSE
5             10 69900788 69900862      9      1      FALSE             TRUE
6             10 69900867 69900940      0     82       TRUE            FALSE
  rightFragEndValid
1              TRUE
2              TRUE
3              TRUE
4             FALSE
5              TRUE
6             FALSE
```

Additionally, further manipulations of the fragment table are possible with basic R table operations. The actual viewpoint fragment is typically vastly overrepresented in a 4C-seq experiment, and removed

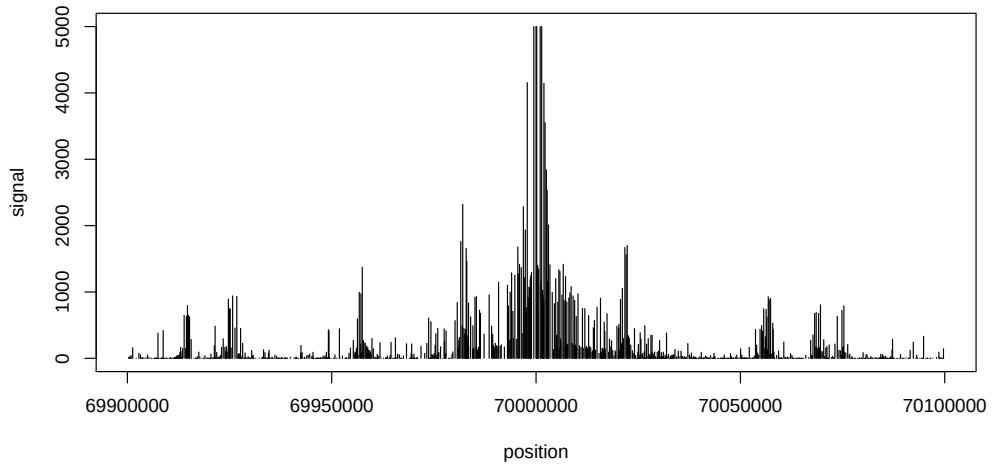

Figure 3: Simulated 4C-seq data, with adapted near-cis region. Peaks with a higher distance from the simulated viewpoint have now a reduced signal strength, noise per fragment is introduced for the peak regions, and blind fragments have an appropriate penalty for their read count.

accordingly, but if this feature is of interest, the fragment can manually be set to a high read count value.

A wrapper function, `simulateSample`, encapsulates the above-mentioned functions, simulates a basic 4C-seq sample, and prints the corresponding fastq file.

2.6 Export simulated data

If the simulated sample is finished, a corresponding fastq file can be exported for the simulated fragment table with `printFastq`.

```
> simLibFile <- system.file("extdata", "simLib_short.csv",
     package="Basic4CSim")
> simLib = read.csv(simLibFile, sep = "\t", header = TRUE)
> simTableFile <- system.file("extdata", "simTable_adapted.csv",
     package="Basic4CSim")
> simTable = read.csv(simTableFile, sep = "\t", header = TRUE)
> printFastq(simTable, simLib, "test", "CATG")
```

References

[Thongjuea *et al.*, 2013] Thongjuea, S., Stadhouders, R., Grosveld, F., et al. (2013) r3Cseq: an R/Bioconductor package for the discovery of long-range genomic interactions from chromosome conformation capture and next-generation sequencing data, *Nucleic Acids Research*, 41(13), e132.

[van de Werken *et al.* A, 2012] van de Werken, H., Landan, G., Holwerda, S., et al. (2012) Robust 4C-seq data analysis to screen for regulatory DNA interactions, *Nature Methods*, **9**, 969-971.

[van de Werken *et al.* B, 2012] van de Werken, H., de Vree, P., Splinter, E., et al. (2012) 4C technology: protocols and data analysis, *Methods Enzymology*, **513**, 89-112.

[Walter *et al.*, 2014] Walter, C., Schuetzmann, D., Rosenbauer, F., et al. (2014) Basic4Cseq: an R/Bioconductor package for analyzing 4C-seq data, *Bioinformatics*, 30(22), 3268-3269.

3 Session Information

```
R version 3.5.2 (2018-12-20)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Debian GNU/Linux 8 (jessie)

Matrix products: default
BLAS: /usr/lib/libblas/libblas.so.3.0
LAPACK: /usr/lib/lapack/liblapack.so.3.0

locale:
 [1] LC_CTYPE=de_DE.UTF-8       LC_NUMERIC=C               LC_TIME=de_DE.UTF-8
 [4] LC_COLLATE=C               LC_MONETARY=de_DE.UTF-8    LC_MESSAGES=de_DE.UTF-8
 [7] LC_PAPER=de_DE.UTF-8       LC_NAME=C                  LC_ADDRESS=C
[10] LC_TELEPHONE=C             LC_MEASUREMENT=de_DE.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] stats4    parallel  stats     graphics  grDevices utils     datasets  methods
[9] base

other attached packages:
 [1] Basic4CSim_0.99                     BSgenome.Ecoli.NCBI.20080805_1.3.1000
 [3] BSgenome_1.48.0                     rtracklayer_1.40.3
 [5] GenomicAlignments_1.16.0            Rsamtools_1.32.0
 [7] SummarizedExperiment_1.10.1         DelayedArray_0.6.1
 [9] BiocParallel_1.14.1                 matrixStats_0.53.1
[11] Biobase_2.40.0                      GenomicRanges_1.32.7
[13] GenomeInfoDb_1.16.0                 Biostrings_2.48.0
[15] XVector_0.20.0                      IRanges_2.14.10
[17] S4Vectors_0.18.3                    BiocGenerics_0.26.0

loaded via a namespace (and not attached):
 [1] zlibbioc_1.26.0        lattice_0.20-35        tools_3.5.2
 [4] grid_3.5.2             Matrix_1.2-15          GenomeInfoDbData_1.1.0
 [7] bitops_1.0-6           RCurl_1.95-4.10        compiler_3.5.2
[10] XML_3.98-1.11
```