r3Cseq: an R package for the discovery of long-range genomic interactions with chromosome conformation capture and next-generation sequencing data

Supat Thongjuea *

January 26, 2015

Contents

*The Weatherall Institute of Molecular Medicine, University of Oxford, UK

Abstract

The coupling of chromosome conformation capture (3C)-based and next-generation sequencing (NGS) enable high-throughput detection of long-range genomic interactions via the generation of novel ligation products between DNA sequences that are closely juxtaposed in vivo. These interactions may involve promoter regions, enhacers and other regulatory and structural elements of chromosomes, and can reveal key details in the regulation of gene expression. 3C-seq is a a variant of the method for the detection of interactions between one chosen genomic element (viewpoint) and the rest of the genome. We present an R/Bioconductor package called *r3Cseq*, designed to perform 3C-seq data analysis in a number of different experimental designs, with or without a control experiment. The package can also be used to perform data analysis for the experiment with replicates. The package provides functions to perform 3C-seq data normalization, statistical analysis for cis/trans interactions and visualization to facilitate the identification of genomic regions that physically interact with the given viewpoints of interest. The r3Cseq package greatly facilitates hypothesis generation and the interpretation of experimental results.

1 Introduction

This vignette describes how to use the *r3Cseq* package. *r3Cseq* is a Bioconductor-compliant R package designed to facilitate the identification of interaction regions generated by chromosome conformation capture and next-generation sequencing (3C-seq). The fundamental principles of 3C-seq briefly described in the following Soler et al. (2010) (Figure 1), isolated cells are treated with a cross-linking agent to preserve in vivo nuclear proximity between DNA sequences. The DNA isolated from these cells is then digested using a primary restriction enzyme, typically a 6-base pairs cutting enzyme such as HindIII, EcoRI or BamHI. The digested product is then ligated under dilute conditions to favor intra-molecular over inter-molecular ligation events. This digested and ligated chromatin yields composite sequences representing (distal) genomic regions that are in close physical proximity in the cell nucleus. The digested and ligated chromatin is then de-crosslinked and subjected to a second restriction digest using either Nla III or Dpn II (a 4-cutter) as a secondary restriction enzyme to decrease the fragment sizes. The resulting digested DNA is then ligated again under diluted conditions, creating small circular fragments. These fragments are inverse PCR-amplified using primers specific for a genomic region of interest (eg. promoter, enhancer, or any other element potentially involved in long-range interactions), termed the "viewpoint". The amplified fragments are then sequenced using massively parallel high-throughput sequencing. Because the 3C-seq procedure hybrid DNA molecules being a combination of viewpoint-specific primers followed by sequences dervied from the ligated interaction fragments. As such, these composite sequences are unmappable and need to be trimmed to removed the viewpoint

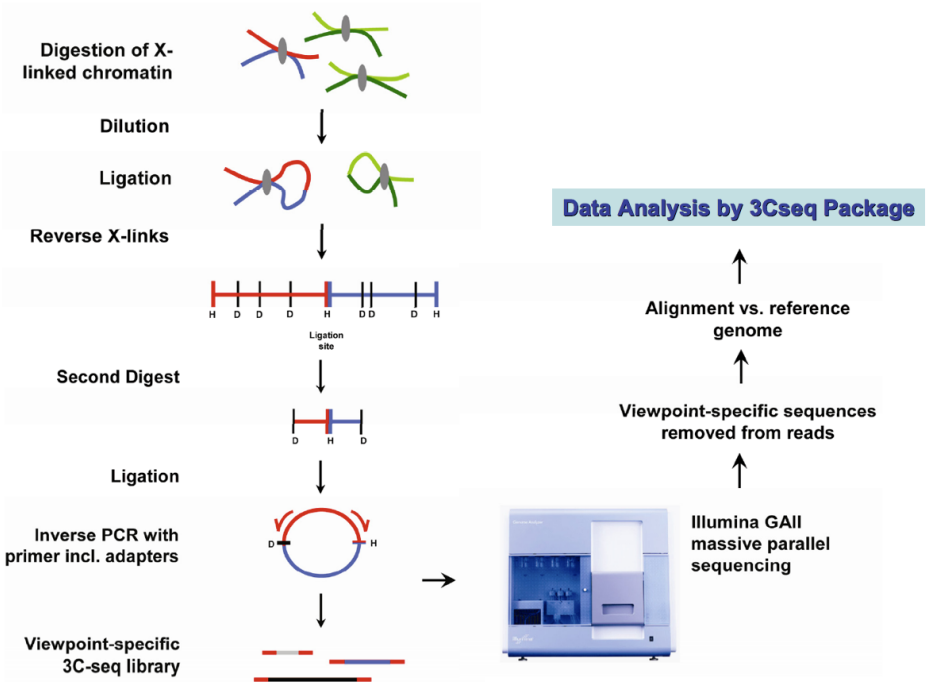

Figure 1: 3C-seq procedures

sequences, thus leaving only the capture sequence fragments for mapping. After trimming, reads are mapped against a reference genome using alignment software such as Bowtie. A mapped read file generated by the mapping software is then transformed to the BAM file and analyzed by using *r3Cseq* package.

r3Cseq package is built on, and extends the functionality of Bioconductor package such as *GenomicRanges*, *BSgenome*, *Rsamtools*, and *rtracklayer*. The package provides classes and methods to facilitate single-end reads, which are generated by the next-generation sequencing. The package can perform data analysis on both single input file (single lane from one experiment) and two input files from an experiment and a control. The package also provides a class and functions to perform 3C-seq data analysis from replicates (see working with replicates section). The key features workflow of *r3Cseq* depicts in the following figure. (Figure 2)

r3Cseq analysis workflow starts from the class initialization. There are two classes found in this package. One is `r3Cseq` class that is designed to support a single experiment in both with and without a control experiment. Another is `r3CseqInBatch` class that is designed to support analysis with replicates. To initialize the class both `r3Cseq` and `r3CseqInBatch`, a user gives the input parameters for example the input file name, genome assembly version, primary restriction fragment name and so on (see more details in the manual page of `r3Cseq` and `r3CseqInBatch`). The class is then will be created and it is ready to perform 3C-seq analysis. The `getRawReads` and `getRawReadsInBatch` functions can be next used to read in the BAM files. It may take

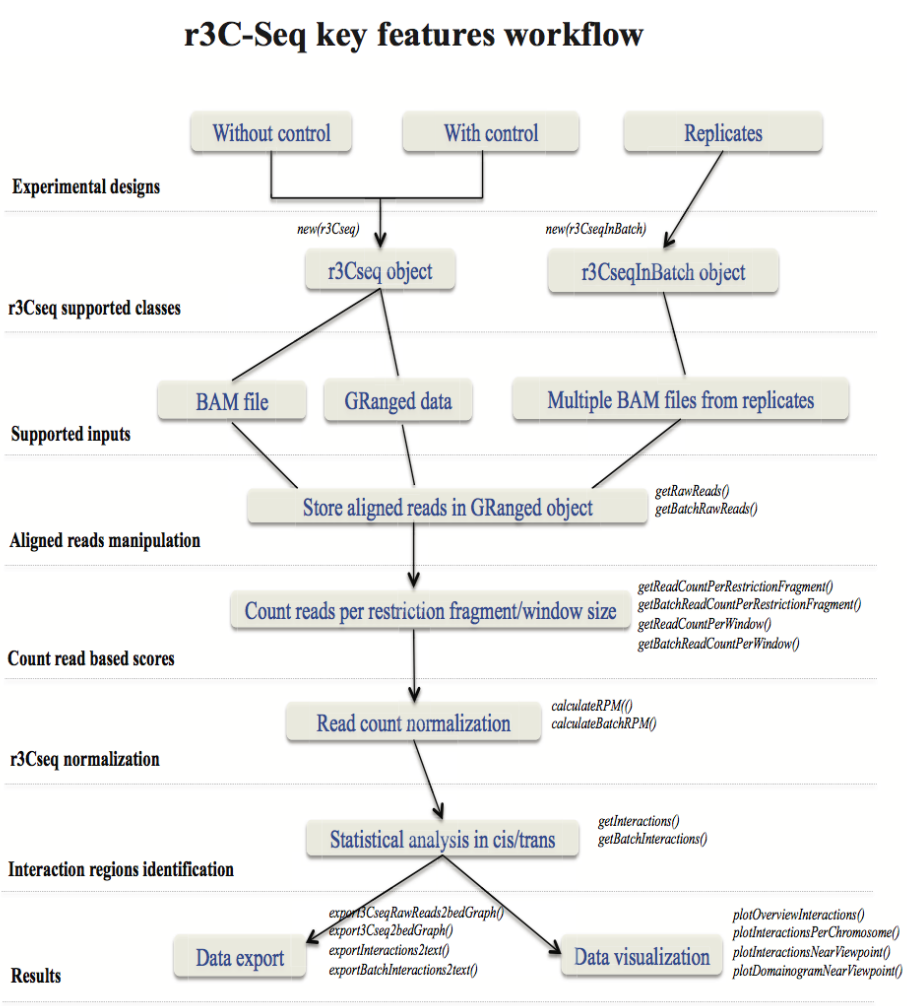

Figure 2: r3Cseq key features workflow

a few minutes for data processing depending on the size of the input BAM files and the speed of CPU and the size of the RAM of a computer that performs analysis. To run the `getRawReadsInBatch` function for replicates, a user might have a powerful computer server. `getRawReads` function reads in aligned reads from input BAM files and transforms aligned reads to the `GRanged` objects that can be stored in the r3Cseq object, whereas `getRawReadsInBatch` processes the data in batch and stores the aligned reads `GRanges` in the R files (.rdata). To count number of reads preparing for downstream analysis, `r3Cseq` provides two ways to count number of reads per region; 1) count number of reads per resitrction fragments, using the function `getReadCountPerRestrictionFragment` and 2) count the number of reads per non-overlapping window size, using function `getReadCountPerWindow`, whereas `getBatchReadCountPerRestrictionFragment` and `getBatchReadCountPerWindow` can do the same for replicates. *r3Cseq* provides `calculateRPM` and `calculateBatchRPM` functions to calculate reads per million per restriction fragment size (RPM) as normalized interaction frequency values. There are two methods to calculate RPM. which are described in the r3Cseq paper **?**. After data normalization, `getInteractions` and `getBatchInteractions` will be performed to identify candidate interactions. Statistical analysis for both cis and trans interactions is described in the r3Cseq paper **?**. *r3Cseq* provides functions `export3CseqRawReads2BedGraph`, `export3Cseq2bedGraph`, `exportInteractions2text`, and `exportBatchInteractions2text` to export raw reads and all identified interactions to the bedGraph file format, which is simply uploaded to the UCSC genome browser. The package also provides functionalities for plotting to show data analysis result of interaction regions. Plotting functions consist of `plotOverviewInteractions`, `plotInteractionPerChromosome`, `plotInteractionNearViewpoint`, and `plotDomainogramNearViewpoint`. These functions will be demonstrated in the visualization of 3C-seq data section.

Here is a list of some of its most important functions.

1. `getRawReads`: a function to read in BAM files.

2. `getBatchRawReads`: a function to read in multiple BAM files for replicates.

3. `getReadCountPerRestrictionFragment` : a function to count the number of reads per restriction fragment. A user has to specify the name of restriction enzyme. The package will then automatically generate the genome-wide restriction fragments and counts how many 3C-seq reads are mapped into that particular restriction fragments.

4. `getBatchReadCountPerRestrictionFragment` : Similar to `getReadCountPerRestrictionFragment` using it for replicates

5. `getReadCountPerWindow` : a function to count the number of reads per defined non-overalapping window size. A user has to specify the window size of interest.

The package will then automatically generate the genome-wide windows and counts how many 3C-seq reads are mapped into that particular windows.

6. `getBatchReadCountPerWindow` : similar to `getReadCountPerWindow` using for replicates

7. `calculateRPM` : a function to calcuate reads per million (RPM) per each restriction fragment

8. `calculateBatchRPM` : similar to `calculateRPM` using for replicates

9. `getInteractions` : a function to perform statistical analysis to identify candidate interactions

10. `getBatchInteractions` : similar to `getInteractions` using for replicates

11. *Visualiztion* : the package contains functions for visualizing the interaction regions with the powerful plotting facilities. These functions are `plotOverviewInteractions`, `plotInteractionsNearViewpoint`,`plotInteractionsPerChromosome`, and `plotDomainogramNearViewpoint`.

12. *Data export* : the package contains functions to export the data into tab-delimited text format, which can be easily uploaded to the UCSC genome browser for further visualization and exploration. Currently it supports the bedGraph format. These functions are `export3CseqRawReads2bedGraph`, `export3Cseq2bedGraph`, `exportInteractions2text`, `exportBatchInteractions2text`. The package can also generate a summary report in PDF format by `generate3CseqReport` function.

2 Preparation input files for r3Cseq

The required input file for *r3Cseq* package is the BAM file, obtained as an output from the mapping software. The represented identifier for a reference genome shown in each input BAM file is important to run *r3Cseq* properly. The represented identifier for each chromosome must be in "chr[1..19XYM]" format for the mouse reference genome and "chr[1..22XYM]" format for the human reference genome. Therefore, before using *r3Cseq* package, a user has to check the identifier for the reference genome. If the identifier for each chromosome found in the mapped file is not in a proper format for example 'mm9_ref_chr01.fa', the Unix command like 'sed' might be used to replace 'mm9_ref_chr01.fa' to 'chr1'.

3 Getting started

The 3C-seq data generated by **?** will be used for the demonstration. The current version of r3Cseq supports mouse, human, and rat genomes. Therefore, the package requires one

of the followings *BSgenome* packages to be installed;*BSgenome.Mmusculus.UCSC.mm9.masked*, *BSgenome.Mmusculus.UCSC.mm10.masked*, *BSgenome.Hsapiens.UCSC.hg18.masked*, *BSgenome.Hsapie and BSgenome.Rnorvegicus.UCSC.rn5.masked*.

Loading the *r3Cseq* package into R.

```
> library(r3Cseq)
```

There are 2 data sets found in the package.

```
> data(Myb_prom_FL)
> data(Myb_prom_FB)
```

1. `Myb_prom_FL`, the 3C-seq data contains the aligned reads of the Myb promoter interactions signal in fetal liver. It was stored in the `GRanges` object processed by the *Rsamtools* package.

2. `Myb_prom_FB`, the 3C-seq data contains the aligned reads of Myb promoter interactions signal in fetal brain.

We will perform *r3Cseq* to discover interaction regions, which possibly interact with the promoter region of Myb gene in both fetal liver and brain **?**.

3.1 r3Cseq object initialization

In this section, we will analyze 3C-seq data, which were derived from fetal liver (expressing high levels of Myb) and fetal brain (expression low level of Myb). The latter will be used as a negative control. More examples of r3Cseq data analysis can be found at the official r3Cseq website `http://r3cseq.genereg.net`. We firstly initialized the r3Cseq object.

```
> my3Cseq.obj<-new("r3Cseq",organismName='mm9',isControlInvolved=TRUE,
+ viewpoint_chromosome='chr10',viewpoint_primer_forward='TCTTTGTTTGATGGCATCTGTT',
+ viewpoint_primer_reverse='AAAGGGGAGGAGAAGGAGGT',expLabel="Myb_prom_FL",
+ contrLabel="MYb_prom_FB",restrictionEnzyme='HindIII')
```

Definition of input parameters is described in the `r3Cseq` help page. We next add raw reads from Myb_prom_FL and Myb_prom_FB to the existing `my3Cseq.obj`.

```
> expRawData(my3Cseq.obj)<-exp.GRanges
> contrRawData(my3Cseq.obj)<-contr.GRanges
```

Type `my3Cseq.obj` to see the r3Cseq object:

```
> my3Cseq.obj
```

```
An object of class "r3Cseq"
Slot "alignedReadsBamExpFile":
[1] ""

Slot "alignedReadsBamContrFile":
[1] ""

Slot "expLabel":
[1] "Myb_prom_FL"

Slot "contrLabel":
[1] "MYb_prom_FB"

Slot "expLibrarySize":
integer(0)

Slot "contrLibrarySize":
integer(0)

Slot "expReadLength":
integer(0)

Slot "contrReadLength":
integer(0)

Slot "expRawData":
GRanges object with 2478476 ranges and 0 metadata columns:
            seqnames          ranges strand
               <Rle>       <IRanges>  <Rle>
        [1]     chr1 3005887-3005930      +
        [2]     chr1 3005887-3005930      +
        [3]     chr1 3005887-3005930      +
        [4]     chr1 3005887-3005930      +
        [5]     chr1 3005887-3005930      +
        ...      ...             ...    ...
  [2478472]     chrM     12341-12384      +
  [2478473]     chrM     12341-12384      +
  [2478474]     chrM     12341-12384      +
  [2478475]     chrM     12341-12384      +
  [2478476]     chrM     12341-12384      +
  -------
  seqinfo: 22 sequences from an unspecified genome; no seqlengths
```

```
Slot "contrRawData":
GRanges object with 2838018 ranges and 0 metadata columns:
            seqnames          ranges strand
               <Rle>       <IRanges>  <Rle>
        [1]     chr1 3046711-3046754      -
        [2]     chr1 3046711-3046754      -
        [3]     chr1 3046711-3046754      -
        [4]     chr1 3046711-3046754      -
        [5]     chr1 3402788-3402831      +
        ...      ...             ...    ...
  [2838014]     chrM     11987-12030      +
  [2838015]     chrM     11987-12030      +
  [2838016]     chrM     11987-12030      +
  [2838017]     chrM     11987-12030      +
  [2838018]     chrM     11990-12033      +
  -------
  seqinfo: 22 sequences from an unspecified genome; no seqlengths

Slot "organismName":
[1] "mm9"

Slot "restrictionEnzyme":
[1] "HindIII"

Slot "viewpoint_chromosome":
[1] "chr10"

Slot "viewpoint_primer_forward":
[1] "TCTTTGTTTGATGGCATCTGTT"

Slot "viewpoint_primer_reverse":
[1] "AAAGGGGAGGAGAAGGAGGT"

Slot "expReadCount":
GRanges object with 0 ranges and 0 metadata columns:
   seqnames    ranges strand
      <Rle> <IRanges>  <Rle>
  -------
  seqinfo: no sequences

Slot "contrReadCount":
```

```
GRanges object with 0 ranges and 0 metadata columns:
   seqnames    ranges strand
      <Rle> <IRanges>  <Rle>
  -------
  seqinfo: no sequences

Slot "expRPM":
GRanges object with 0 ranges and 0 metadata columns:
   seqnames    ranges strand
      <Rle> <IRanges>  <Rle>
  -------
  seqinfo: no sequences

Slot "contrRPM":
GRanges object with 0 ranges and 0 metadata columns:
   seqnames    ranges strand
      <Rle> <IRanges>  <Rle>
  -------
  seqinfo: no sequences

Slot "expInteractionRegions":
GRanges object with 0 ranges and 0 metadata columns:
   seqnames    ranges strand
      <Rle> <IRanges>  <Rle>
  -------
  seqinfo: no sequences

Slot "contrInteractionRegions":
GRanges object with 0 ranges and 0 metadata columns:
   seqnames    ranges strand
      <Rle> <IRanges>  <Rle>
  -------
  seqinfo: no sequences

Slot "isControlInvolved":
[1] TRUE
```

3.2 Getting reads per restriction fragments/user defined window size

To get number of reads per restriction fragement, function `getReadCountPerRestrictionFragment` will be performed.

```
> getReadCountPerRestrictionFragment(my3Cseq.obj)
```

```
[1] "Fragmenting genome by....HindIII...."
[1] "Count processing in the experiment is done."
[1] "Count processing in the control is done."
```

The package provides the function `getReadCountPerWindow` to count number of reads per non-overlapping window size defined by a user.

3.3 Normalization

We next perform normalization.

```
> calculateRPM(my3Cseq.obj)
```

```
[1] "Normal RPM calculation is done."
```

3.4 Getting interaction regions

After normalization, the `getInteractions` function will be performed.

```
> getInteractions(my3Cseq.obj,fdr=0.05)
```

```
[1] "Calculation is done. Use function 'expInteractionRegions' or 'contrInteractionRegi
```

In order to see the result of interaction regions, Two functions `expInteractionRegions` and `contrInteractionRegions` need to be used to access the slot of r3Cseq object. To get the result of interaction regions for the experiment, `expInteractionRegions` will be performed.

```
> fetal.liver.interactions<-expInteractionRegions(my3Cseq.obj)
> fetal.liver.interactions
```

```
GRanges object with 19502 ranges and 5 metadata columns:
          seqnames          ranges strand |    nReads
             <Rle>       <IRanges>  <Rle> | <integer>
      [1]    chr10 3009255-3014345      * |         3
      [2]    chr10 3067271-3068193      * |       122
      [3]    chr10 3127452-3132355      * |       101
      [4]    chr10 3132360-3134033      * |        70
      [5]    chr10 3141247-3145770      * |       113
      ...      ...             ...    ... .       ...
  [19498]     chrY 1229367-1231265      * |         2
  [19499]     chrY 1279671-1287427      * |         2
  [19500]     chrY 1293808-1295589      * |       137
  [19501]     chrY 1686797-1687184      * |         2
```

```
  [19502]     chrY 2641919-2642337      * |         1
              RPMs          z   p.value   q.value
         <numeric>  <numeric> <numeric> <numeric>
     [1]  0.0605738 -0.1731100  1.000000         1
     [2] 11.8504107  0.1063353  0.915316         1
     [3]  9.0555762  0.0407624  0.967485         1
     [4]  5.3727006 -0.0368364  1.000000         1
     [5] 10.6253420  0.0673961  0.946266         1
     ...        ...        ...       ...       ...
  [19498]  0.0340050  -0.486505  1.000000         1
  [19499]  0.0340050  -0.486505  1.000000         1
  [19500] 13.9779585   0.249857  0.802698         1
  [19501]  0.0340050  -0.486505  1.000000         1
  [19502]  0.0126735  -0.491960  1.000000         1
  -------
  seqinfo: 21 sequences from an unspecified genome; no seqlengths
```

To get the result of interaction regions for the control, `contrInteractionRegions` will be performed.

```
> fetal.brain.interactions<-contrInteractionRegions(my3Cseq.obj)
> fetal.brain.interactions

GRanges object with 9684 ranges and 5 metadata columns:
         seqnames          ranges strand |    nReads
            <Rle>       <IRanges>  <Rle> | <integer>
     [1]    chr10 3019822-3022796      * |       425
     [2]    chr10 3063618-3067025      * |       525
     [3]    chr10 3087994-3096004      * |       309
     [4]    chr10 3138895-3141226      * |       750
     [5]    chr10 3148290-3157485      * |         4
     ...      ...             ...    ... .       ...
  [9680]     chrY 1686797-1687184      * |         1
  [9681]     chrY 1690876-1691262      * |         1
  [9682]     chrY 2371312-2391988      * |         1
  [9683]     chrY 2884699-2887174      * |         1
  [9684]     chrY 2890136-2899310      * |         1
              RPMs          z   p.value   q.value
         <numeric>  <numeric> <numeric> <numeric>
     [1] 44.851838  0.2283635  0.819364         1
     [2] 59.109041  0.3818504  0.702572         1
     [3] 29.577233  0.0709858  0.943409         1
     [4] 94.187285  0.7229188  0.469730         1
```

```
    [5]  0.101139 -0.3636379  1.000000         1
    ...       ...        ...       ...       ...
[9680] 0.0165381  -0.474278         1         1
[9681] 0.0165381  -0.474278         1         1
[9682] 0.0165381  -0.474278         1         1
[9683] 0.0165381  -0.474278         1         1
[9684] 0.0165381  -0.474278         1         1
-------
seqinfo: 21 sequences from an unspecified genome; no seqlengths
```

3.5 Getting the viewpoint information

To see the viewpoint information, `getViewpoint` function can be used. `getViewpoint` will return the RangedData object of the viewpoint information.

```
> viewpoint<-getViewpoint(my3Cseq.obj)
> viewpoint
```

```
GRanges object with 1 range and 0 metadata columns:
      seqnames            ranges strand
         <Rle>         <IRanges>  <Rle>
  [1]    chr10 20880662-20880775      *
  -------
  seqinfo: 1 sequence from an unspecified genome; no seqlengths
```

4 Visualization of 3C-seq data

r3Cseq package provides visualization functions. These functions are `plotOverviewInteractions`, `plotInteractionsNearViewpoint`, `plotInteractionsPerChromosome`, and `PlotDomainogramNearViewpoint`.

4.1 The overview plot of interactions

`plotOverviewInteractions` function shows the overview of interaction regions distributed across genome.

```
> plotOverviewInteractions(my3Cseq.obj)
```

4.2 Plot of interactions in cis

`plotInteractionsNearViewpoint` function shows the zoom in of interaction regions located close to the viewpoint.

```
> plotInteractionsNearViewpoint(my3Cseq.obj)
```

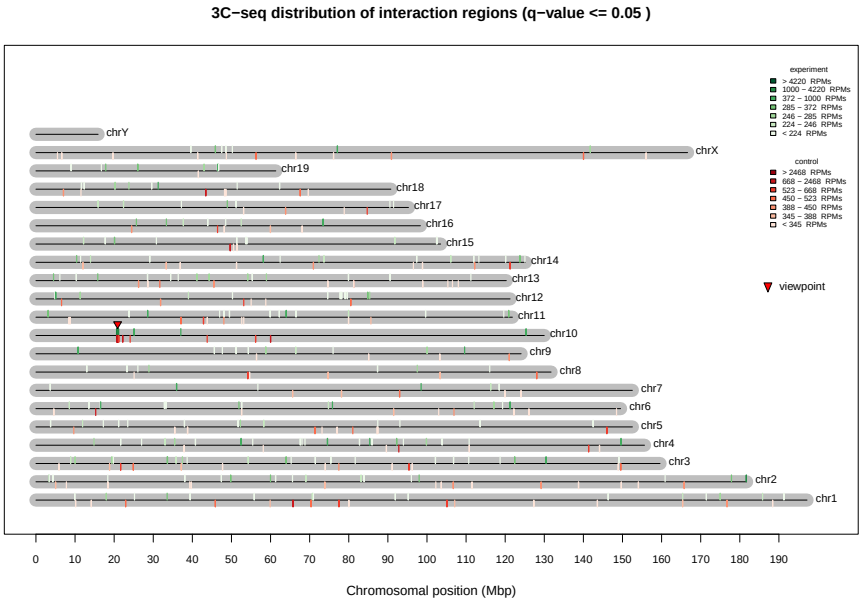

Figure 3: Distribution of interaction regions across genome

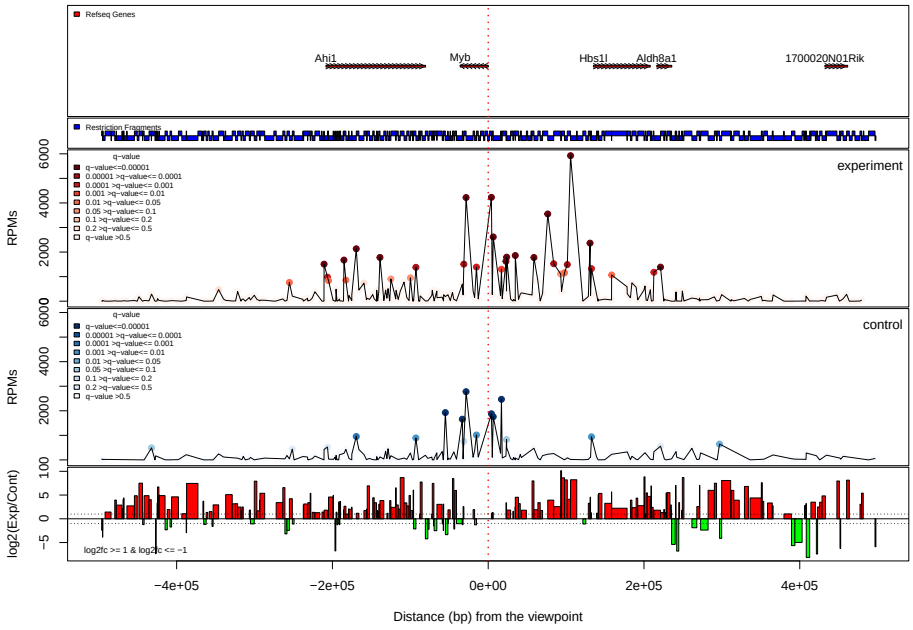

Figure 4: Zoom in interaction regions near the viewpoint

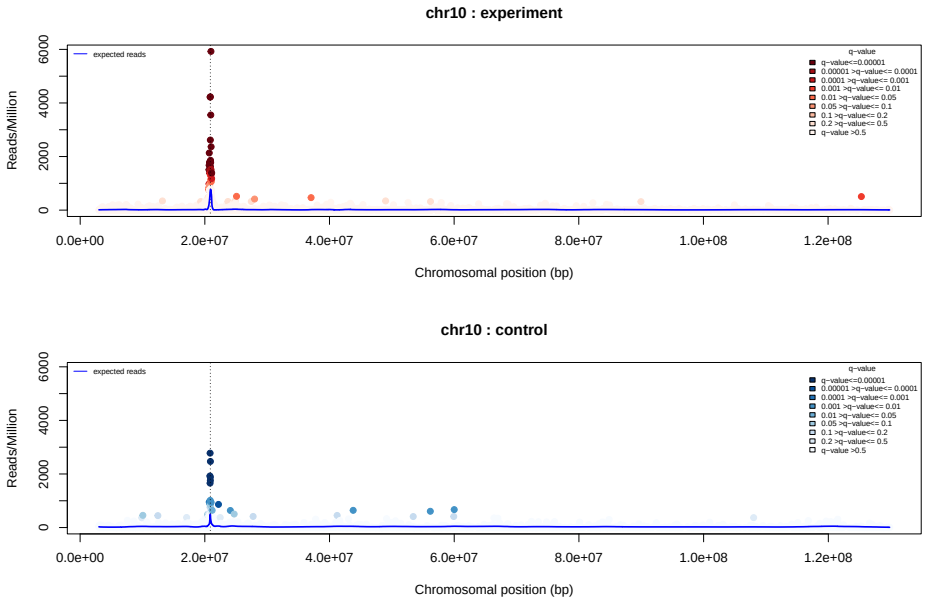

Figure 5: Distribution of interaction regions across chromosome 10

4.3 Plot of interactions in each selected chromosome

`plotInteractionsPerChromosome` function shows the interaction regions found in the chromosome10.

```
> plotInteractionsPerChromosome(my3Cseq.obj,"chr10")
```

4.4 Domainogram of interactions

`plotDomainogramNearViewpoint` function shows the domainogram of interactions found in cis. This function may takes several minutes to produce domainograms. We therefore, skip this command to produce plots for the vignette. You can see the example of the plots and find more details at `http://r3Cseq.genereg.net`.

```
> #plotDomainogramNearViewpoint(my3Cseq.obj)
```

4.5 Associate interaction signals to the Refseq genes

`getExpInteractionsInRefseq` and `getContrInteractionsInRefseq` functions can be used to detect the list of genes that contain significant interaction signals in their proximity.

```
> detected_genes<-getExpInteractionsInRefseq(my3Cseq.obj)
> head(detected_genes)
```

```
    chromosome gene_name total_nReads total_RPMs
173      chr10       Myb        43217  20539.955
143      chr10     Hbs1l        34310  16031.869
29       chr10      Ahi1        29738  11912.850
135       chr4    Gm5506        15792   3939.331
33       chr10   Aldh8a1         9422   3633.383
137       chr4    Gm5801        13482   3350.087
```

4.6 Export interactions to the bedGraph format

`export3Cseq2bedGraph` function exports all interactions from the `RangedData` to the bedGraph format, which simply upload to the UCSC genome browser.

```
> #export3Cseq2bedGraph(my3Cseq.obj)
```

4.7 Summary report

`generate3CseqReport` function generates the summary report from *r3Cseq* analysis results. The report contains a pdf file for all plots and text files of interaction regions. This function may takes several minutes to produce the report. We therefore, skip this command during the vignette creation.

```
> #generate3CseqReport(my3Cseq.obj)
```

5 Working with replicates

The example of how to work with replicats can be found at `http://r3cseq.genereg.net/`.

6 r3Cseq website

We have developed the website `http://r3cseq.genereg.net`. The website provides more details of *r3Cseq* analysis pipeline. The example data sets and the current version of *r3Cseq* package can be downloaded from the website.

7 Session Info

```
> sessionInfo()
```

```
R version 4.1.0 (2021-05-18)
Platform: x86_64-w64-mingw32/x64 (64-bit)
```

```
Running under: Windows Server x64 (build 17763)

Matrix products: default

locale:
[1] C

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

other attached packages:
 [1] RSQLite_2.2.7
 [2] BSgenome.Mmusculus.UCSC.mm9.masked_1.3.99
 [3] BSgenome.Mmusculus.UCSC.mm9_1.4.0
 [4] BSgenome_1.61.0
 [5] r3Cseq_1.39.0
 [6] qvalue_2.25.0
 [7] VGAM_1.1-5
 [8] rtracklayer_1.53.0
 [9] Rsamtools_2.9.0
[10] Biostrings_2.61.0
[11] XVector_0.33.0
[12] GenomicRanges_1.45.0
[13] GenomeInfoDb_1.29.0
[14] IRanges_2.27.0
[15] S4Vectors_0.31.0
[16] BiocGenerics_0.39.0

loaded via a namespace (and not attached):
 [1] MatrixGenerics_1.5.0        Biobase_2.53.0
 [3] bit64_4.0.5                 gsubfn_0.7
 [5] assertthat_0.2.1            blob_1.2.1
 [7] GenomeInfoDbData_1.2.6      yaml_2.2.1
 [9] pillar_1.6.1                lattice_0.20-44
[11] glue_1.4.2                  chron_2.3-56
[13] RColorBrewer_1.1-2          colorspace_2.0-1
[15] Matrix_1.3-4                plyr_1.8.6
[17] XML_3.99-0.6                pkgconfig_2.0.3
[19] zlibbioc_1.39.0             purrr_0.3.4
[21] scales_1.1.1                BiocParallel_1.27.0
[23] tibble_3.1.2                generics_0.1.0
```

```
[25] ggplot2_3.3.3               sqldf_0.4-11
[27] ellipsis_0.3.2              cachem_1.0.5
[29] SummarizedExperiment_1.23.0 proto_1.0.0
[31] magrittr_2.0.1              crayon_1.4.1
[33] memoise_2.0.0               fansi_0.5.0
[35] tools_4.1.0                 data.table_1.14.0
[37] BiocIO_1.3.0                lifecycle_1.0.0
[39] matrixStats_0.59.0          stringr_1.4.0
[41] munsell_0.5.0               DelayedArray_0.19.0
[43] compiler_4.1.0              rlang_0.4.11
[45] grid_4.1.0                  RCurl_1.98-1.3
[47] rstudioapi_0.13             rjson_0.2.20
[49] tcltk_4.1.0                 bitops_1.0-7
[51] restfulr_0.0.13             gtable_0.3.0
[53] DBI_1.1.1                   reshape2_1.4.4
[55] R6_2.5.0                    GenomicAlignments_1.29.0
[57] dplyr_1.0.6                 fastmap_1.1.0
[59] bit_4.0.4                   utf8_1.2.1
[61] stringi_1.6.2               Rcpp_1.0.6
[63] vctrs_0.3.8                 tidyselect_1.1.1
```

References

Soler, E., Andrieu-Soler, C., de Boer, E., Bryne, J. C., Thongjuea, S., Stadhouders, R., Palstra, R.-J., Stevens, M., Kockx, C., van IJcken, W., Hou, J., Steinhoff, C., Rijkers, E., Lenhard, B., and Grosveld, F. (2010). The genome-wide dynamics of the binding of Ldb1 complexes during erythroid differentiation. *Genes & Development*, 24(3):277–289.