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

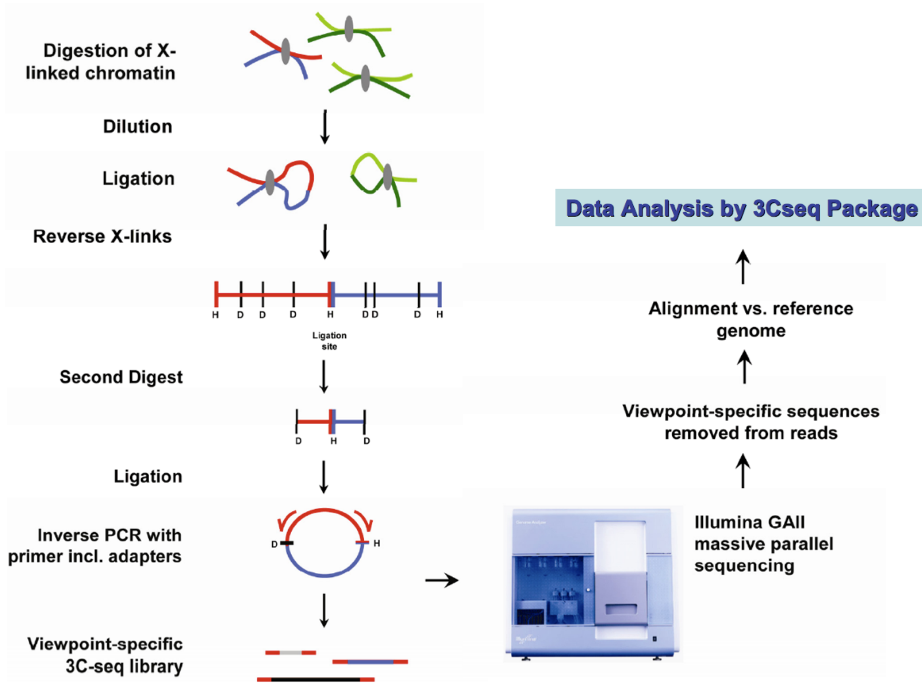

Figure 1: 3C-seq procedures

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 a

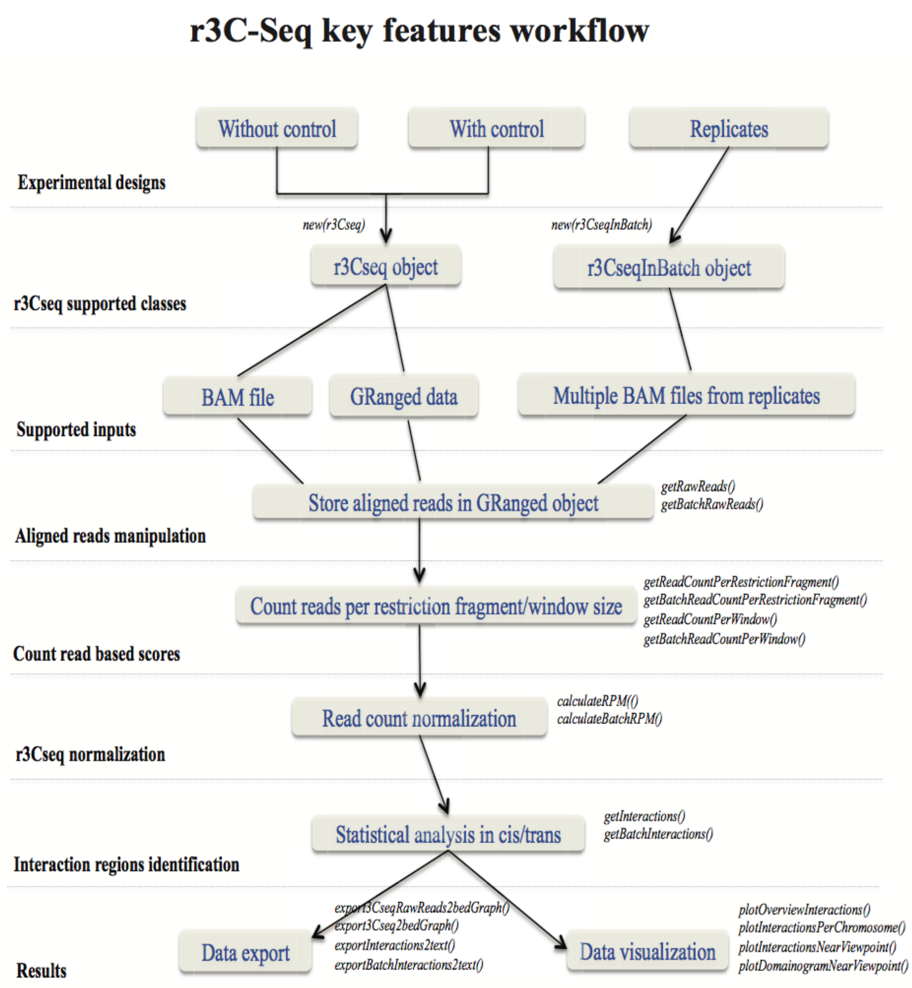

Figure 2: r3Cseq key features workflow

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 Thongjuea et al. (2013). 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 Thongjuea et al. (2013). *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 Stadhouders et al. (2012) 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.Hsapiens.UCSC.hg19.masked, 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 Stadhouders et al. (2012).

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":
RangedData with 0 rows and 0 value columns across 0 spaces

Slot "contrReadCount":
RangedData with 0 rows and 0 value columns across 0 spaces

Slot "expRPM":
RangedData with 0 rows and 0 value columns across 0 spaces
```

```
Slot "contrRPM":
RangedData with 0 rows and 0 value columns across 0 spaces

Slot "expInteractionRegions":
RangedData with 0 rows and 0 value columns across 0 spaces

Slot "contrInteractionRegions":
RangedData with 0 rows and 0 value columns across 0 spaces

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
```

```
RangedData with 19502 rows and 5 value columns across 21 spaces
         space             ranges   |    nReads
      <factor>          <IRanges>   | <integer>
1         chr1    3005881-3006265   |         5
2         chr1    3044137-3055946   |         2
3         chr1    3072236-3078784   |        19
4         chr1    3154269-3155625   |         2
5         chr1    3155630-3155656   |         2
6         chr1    3265404-3268167   |         1
7         chr1    3272080-3273212   |         1
8         chr1    3376639-3378367   |       293
9         chr1    3536048-3536283   |       105
...        ...                ... ...       ...
19494     chrX 165883099-165888229  |         6
19495     chrX 165950246-165952458  |         1
19496     chrX 165980487-165985135  |         1
19497     chrY        73040-77927   |         1
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
               <numeric>          <numeric>
1      0.125366775410638  -0.43137077246295
2     0.0340049639221147 -0.446339868032016
3      0.838988062792645 -0.361514993140646
4     0.0340049639221147 -0.446339868032016
5     0.0340049639221147 -0.446339868032016
6     0.0126734717809562 -0.451329566555038
7     0.0126734717809562 -0.451329566555038
8       41.2620896500628   1.00566240216732
9       9.57050590241137 0.0675990798392258
...                  ...                ...
19494  0.162529159602502 -0.426381073939929
```

```
19495 0.0126734717809562 -0.451329566555038
19496 0.0126734717809562 -0.451329566555038
19497 0.0126734717809562 -0.451329566555038
19498 0.0340049639221147 -0.446339868032016
19499 0.0340049639221147 -0.446339868032016
19500   13.9779584889841  0.227269432575922
19501 0.0340049639221147 -0.446339868032016
19502 0.0126734717809562 -0.451329566555038
                p.value   q.value
              <numeric> <numeric>
1                     1         1
2                     1         1
3                     1         1
4                     1         1
5                     1         1
6                     1         1
7                     1         1
8      0.314577994970149         1
9      0.946104787907482         1
...                 ...       ...
19494                 1         1
19495                 1         1
19496                 1         1
19497                 1         1
19498                 1         1
19499                 1         1
19500 0.820214240494089         1
19501                 1         1
19502                 1         1
```

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

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

RangedData with 9684 rows and 5 value columns across 21 spaces
        space          ranges   |    nReads
     <factor>       <IRanges>   | <integer>
1        chr1 3044137-3055946   |         4
2        chr1 3398749-3404191   |         1
3        chr1 3524538-3526603   |       133
4        chr1 3692880-3695243   |         6
```

```
5         chr1 3879715-3883178   |          2
6         chr1 3913040-3923201   |          1
7         chr1 4096094-4097561   |          1
8         chr1 4130630-4150663   |        691
9         chr1 4323088-4329958   |          1
...        ...               ... ...        ...
9676      chrY 1232902-1237361   |          1
9677      chrY 1279671-1287427   |          4
9678      chrY 1293808-1295589   |         82
9679      chrY 1466934-1470207   |        359
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
               <numeric>          <numeric>
1      0.101138955526621 -0.463882590117245
2       0.01653813879084 -0.471677549029147
3       9.83413074398331 -0.128699356905453
4      0.171765140855819 -0.458685950842644
5     0.0408980449857918 -0.469079229391846
6       0.01653813879084 -0.471677549029147
7       0.01653813879084 -0.471677549029147
8       84.6276411817108   1.32116300070835
9       0.01653813879084 -0.471677549029147
...                  ...                ...
9676    0.01653813879084 -0.471677549029147
9677   0.101138955526621 -0.463882590117245
9678    5.22850186649733 -0.261213658407789
9679    35.9782823873425  0.458520881124509
9680    0.01653813879084 -0.471677549029147
9681    0.01653813879084 -0.471677549029147
9682    0.01653813879084 -0.471677549029147
9683    0.01653813879084 -0.471677549029147
9684    0.01653813879084 -0.471677549029147
               p.value   q.value
             <numeric> <numeric>
1                    1         1
2                    1         1
3                    1         1
4                    1         1
```

```
5                       1          1
6                       1          1
7                       1          1
8     0.186447020174595          1
9                       1          1
...                   ...        ...
9676                    1          1
9677                    1          1
9678                    1          1
9679  0.646578264184368          1
9680                    1          1
9681                    1          1
9682                    1          1
9683                    1          1
9684                    1          1
```

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

RangedData with 1 row and 0 value columns across 1 space
     space            ranges |
  <factor>         <IRanges> |
1    chr10 20880662-20880775 |
```

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)
```

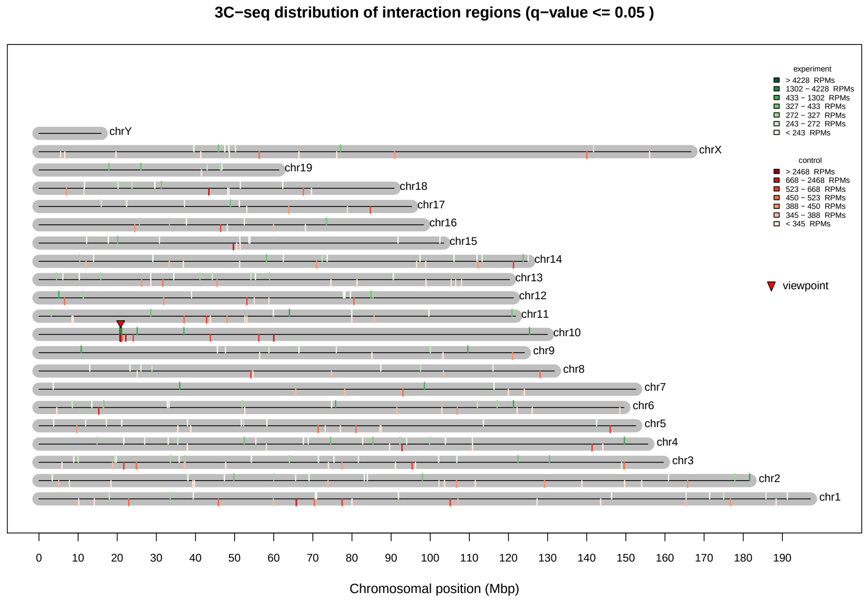

Figure 3: Distribution of interaction regions across genome

4.2 Plot of interactions in cis

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

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

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)
```

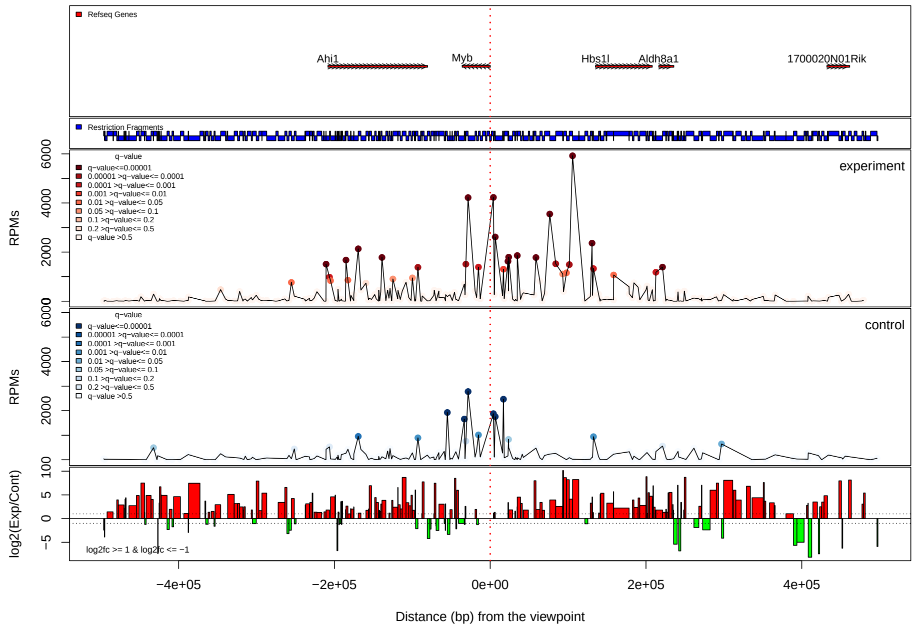

Figure 4: Zoom in interaction regions near the viewpoint

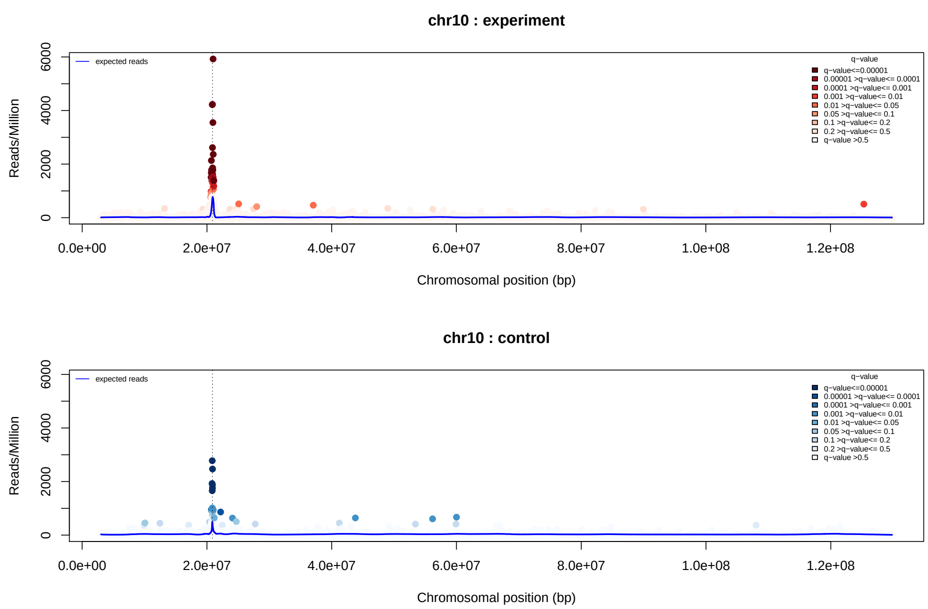

Figure 5: Distribution of interaction regions across chromosome 10

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
141      chr10       Myb        43217  20539.955
118      chr10     Hbs1l        34310  16031.869
19       chr10      Ahi1        29738  11912.850
23       chr10   Aldh8a1         9422   3633.383
111       chr4    Gm5506        14317   3632.061
112       chr4    Gm5801        12007   3042.817
```

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 3.5.1 (2018-07-02)
Platform: x86_64-apple-darwin15.6.0 (64-bit)
Running under: macOS Sierra 10.12.6

Matrix products: default
BLAS: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRlapack.dylib

locale:
[1] C/en_GB.UTF-8/C/C/C/en_GB.UTF-8

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

other attached packages:
 [1] RSQLite_2.1.1
 [2] BSgenome.Mmusculus.UCSC.mm9.masked_1.3.99
 [3] BSgenome.Mmusculus.UCSC.mm9_1.4.0
 [4] BSgenome_1.48.0
 [5] r3Cseq_1.26.0
 [6] qvalue_2.12.0
 [7] VGAM_1.0-6
 [8] rtracklayer_1.40.6
 [9] Rsamtools_1.32.3
[10] Biostrings_2.48.0
[11] XVector_0.20.0
[12] GenomicRanges_1.32.6
[13] GenomeInfoDb_1.16.0
[14] IRanges_2.14.11
[15] S4Vectors_0.18.3
[16] BiocGenerics_0.26.0

loaded via a namespace (and not attached):
 [1] Biobase_2.40.0            bit64_0.9-7
 [3] gsubfn_0.7                assertthat_0.2.0
 [5] blob_1.1.1                GenomeInfoDbData_1.1.0
 [7] pillar_1.3.0              lattice_0.20-35
 [9] glue_1.3.0                chron_2.3-52
```

```
[11] digest_0.6.16                RColorBrewer_1.1-2
[13] colorspace_1.3-2             Matrix_1.2-14
[15] plyr_1.8.4                   XML_3.98-1.16
[17] pkgconfig_2.0.2              zlibbioc_1.26.0
[19] purrr_0.2.5                  scales_1.0.0
[21] BiocParallel_1.14.2          tibble_1.4.2
[23] ggplot2_3.0.0                sqldf_0.4-11
[25] SummarizedExperiment_1.10.1 lazyeval_0.2.1
[27] proto_1.0.0                  magrittr_1.5
[29] crayon_1.3.4                 memoise_1.1.0
[31] tools_3.5.1                  data.table_1.11.4
[33] matrixStats_0.54.0           stringr_1.3.1
[35] munsell_0.5.0                DelayedArray_0.6.5
[37] bindrcpp_0.2.2               compiler_3.5.1
[39] rlang_0.2.2                  grid_3.5.1
[41] RCurl_1.95-4.11              tcltk_3.5.1
[43] bitops_1.0-6                 gtable_0.2.0
[45] DBI_1.0.0                    reshape2_1.4.3
[47] R6_2.2.2                     GenomicAlignments_1.16.0
[49] dplyr_0.7.6                  bit_1.1-14
[51] bindr_0.1.1                  stringi_1.2.4
[53] Rcpp_0.12.18                 tidyselect_0.2.4
```

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.

Stadhouders, R., Thongjuea, S., Andrieu-Soler, C., Palstra, R., Bryne, J., De Boer, E., Kockx, C., van der Sloot, A., van den Hout, M., van IJcken, W., Lenhard, B., Grosveld, F., and Soler, E. (2012). Dynamic long-range chromatin interactions control Myb proto-oncogene transcription during erythriod development. *EMBO*, 31:986–999.

Thongjuea, S., Stadhouders, R., Grosveld, F., Soler, E., and Lenhard, B. (2013). r3Cseq-an R package for the discovery of long-range genomic interactions with chromosome conformation capture and next-generation sequencing data. *Nucleic Acids Research*, 41.