RIPSeeker: a statistical package for identifying protein-associated transcripts from RIP-seq experiments

Yue Li
yueli@cs.toronto.edu
October 29, 2019

1 Introduction

Ribonucleoprotein (RNP) immunoprecipitation (IP) followed by high-throughput sequencing (RIP-seq) has recently been developed to discover genome-wide RNA transcripts that interact with a protein or protein complex. RIP-seq is similar to both RNA-seq and ChIP-seq, but presents unique properties and challenges. Currently, no statistical tool is dedicated to RIP-seq analysis. We developed RIPSeeker, an HMM-based R package for de novo RIP peak predictions. RIPSeeker infers and discriminate RIP peaks from background or control (if available) using two-state HMM with negative binomial emission probability followed by statistical tests for model confidence. To evaluate the performance of RIPSeeker, we used the published RIP-seq data for PRC2 and also generated in-house data for CCNT1. Under an equal footing, RIPSeeker compares competitively with ChIP-seq and transcript-based methods in predicting canonical protein-associated transcripts with high statistical confidence. Importantly, RIPSeeker has the ability to model reads on plus and minus strand separately, which allows identification of strand-specific noncoding RNA (e.g., antisense transcripts). Comparing to the published methods, RIPSeeker is adaptive to a larger dynamical range of peaks suitable for detecting the entire transcripts with various lengths rather than the punctuated binding sites.

While RIPSeeker is specifically tailored for RIP-seq data analysis, it also provides a suite of bioinformatics tools integrated within this self-contained software package comprehensively addressing issues ranging from post-alignments processing to visualization and annotation. In addition, a rule-based approach is provided as an additional function named rulebaseRIPSeek for user to obtain RPKM/FPKM (and fold-change) for the gene/transcripts expressions in RIP (and control) based on automatically retrieved online Ensembl/UCSC annotation given single or paired-end alignments. This vignette provides a guide to the most common application of RIPSeeker package.

2 RIPSeeker overview

Figure ?? in the manuscript (in press) for peer-review presents the workflow of RIPSeeker. The input for RIPSeeker is a list of read alignments in BAM/BED/SAM format. Map-
ping reads to the reference genome can be performed by any RNA-seq aligner such as TopHat [Trapnell et al. 2009]. After post-processing the alignment input by `combineAlignGals`, RIPSeeker first stratifies the genome into bins of automatically selected (selectBinSize) or a fixed user-defined size. Each bin may contain more than one aligned read. Multiple bins may together correspond to a single RNA transcript that binds to the protein of interest. Thus, these bins when treated as individual observations are not independent identically distributed (i.i.d.) and need to be treated as dependent events. Hidden Markov model (HMM) provides a sensible and efficient way to probabilistically model the dependence between sequential events through hidden variables [Rabiner, 1989, Bishop, 2006]. The adaptation of HMM is inspired by HPeak, which was specifically designed for ChIP-seq [Qin et al., 2010].

As an overview, RIPSeeker consists of two major steps: probabilistic inference of RIP regions (mainSeek) and significance test for the inferred RIP regions from HMM (seekRIP). In the first step, we apply a two-state HMM to model the background and RIP distributions (or emission probabilities) of RIP-seq read counts as negative binomial distributions, which has been shown by Anders and Huber [2010] to be a more realistic parametric model than Gaussian and Poisson models. The parameters of HMM are learned from the data by expectation-maximization (EM). The intermediate quantities required in the EM iterations are efficiently computed using forward-backward algorithm. After the optimization, Viterbi algorithm is applied to derive the most probable sequence of hidden states, which encodes whether each region is background (1) or RIP (2) across the genome. The consecutive RIP bins are merged into a single RIP region. In the second step, we compute the statistical significance of each RIP region with or without a control library based on the posterior probabilities derived directly from the HMM.

RIPSeeker is able to detect strand-specific RIP regions by running the same workflow on either plus and minus strand separately, making use of the strand-specific information retained in the original RIP-seq protocol [Cloonan et al., 2008, Zhao et al., 2010]. In addition, RIPSeeker takes advantage of modern computational architecture equipped with multiple processors by treating each chromosome as an independent thread and computing multiple threads in parallel using `mclapply` from `multicore` R package. Thus, the most time consuming step such as HMM inference operates on per-chromosome basis each running on a separate CPU core. The parallel computing is much more computationally and memory efficient than computing the entire genome all at once by treating it as a single concatenated sequence. RIPSeeker has numerous other features including disambiguating multihits (i.e., reads mapped to multiple loci with disambiguateMultihits), automatic annotation of RIP regions (annotateRIP), GO enrichment analysis (annotateRIP), and UCSC visualization (viewRIP).

Although the ultimate goal is to predict RIP regions, each step or component in the workflow has been implemented as a stand-alone function. Please referred to ?? in the manuscript (may not be available during peer-review) for the underlying algorithms and the help manual for their usage:

```r
> library(RIPSeeker)
> ?RIPSeeker
```
3 RIP-seq data

3.1 PRC2 in mouse embryonic stem cell

Only a few RIP-seq datasets are available. In this tutorial, we will make use of part of the dataset from Zhao et al. [2010] (GEO accession: GSE17064). The data was generated to identify transcripts physically associated with Ezh2 (a PRC2 unique subunit) in mouse embryonic stem cell (mESC). A negative control was also generated from mutant mESC depleted of Ezh2 (Ezh2 -/-) (MT). Here we will use the RIP data consisting of two technical replicates (code ID: SRR039210-1, SRR039212) and the MT control (code ID: SRR039214). Notably, the library construction and strand-specific sequencing generated sequences from the opposite strand of the PRC2-bound RNA [Zhao et al., 2010], consequently, each read will be treated as if it were reverse complemented. After the quality control (QC) and alignments (?? and ?? in Supplementary Data for the manuscript), the technical replicates were merged, resulting in 1,022,474 and 208,445 reads mapped to unique loci of the mouse reference genome (mm9 build) for RIP and control, respectively. To make the demonstration and the package size more manageable, only the alignments to chromosome X (chrX) were extracted to generate the test data stored in the package. The full data (containing all of the processed alignments for the test data and another biological replicate SRR039213) are available as a Bioconductor data package RIPSeekerData. User can try the same command below on the full dataset.

```r
> BiocManager::install("RIPSeekerData")
> library(RIPSeekerData)
> extdata.dir <- system.file("extdata", package="RIPSeekerData")
> bamFiles <- list.files(extdata.dir, "\.bam$", recursive=TRUE, full.names=TRUE)
> bamFiles <- grep("PRC2/", bamFiles, value=TRUE)
```

3.2 CCNT1 in HEK293

The data for CCNT1 and GFP control were generated in-house in two experiments. The pilot experimental data contain 5,853 and 4,556 uniquely mapped read after the stringent QC for the CCNT1 and GFP control, respectively. Same as in the PRC2 data, the reads came from the second strand of the cDNA synthesis opposite to the original RNA strand. The non-strand-specific library from the second screen has deeper coverage with 26,859 and 45,024 uniquely aligned reads under QC for CCNT1 and GFP, respectively. CCNT1 is known to only associate with RN7SK, which can be used as an empirical measurement of sensitivity and specificity of RIPSeeker. The data are in RIPSeekerData package.

3.3 User-supplied data

For the following example, user may replace the extdata.dir with the directory containing their own alignment data and change the cNAME to point to the specific control (if applicable).

```r
> # Retrieve system files
> # OR change it to the extdata.dir from the code chunk above
> # to get RIP predictions on the full alignment data
> extdata.dir <- system.file("extdata", package="RIPSeeker")
> bamFiles <- list.files(extdata.dir, ".bam", +
    recursive=TRUE, full.names=TRUE)
> bamFiles <- grep("PRC2", bamFiles, value=TRUE)
> # RIP alignment for Ezh2 in mESC
> ripGal <- combineAlignGals(grep(pattern="SRR039214", bamFiles, value=TRUE, invert=TRUE),
+ reverseComplement=TRUE, genomeBuild="mm9")
> # Control RIP alignments for mutant Ezh2 -/- mESC
> ctlGal <- combineAlignGals(grep(pattern="SRR039214", bamFiles, value=TRUE, invert=FALSE),
+ reverseComplement=TRUE, genomeBuild="mm9")
> ripGal

GAlignments object with 31054 alignments and 1 metadata column:

<table>
<thead>
<tr>
<th>seqnames</th>
<th>strand</th>
<th>cigar</th>
<th>qwidth</th>
<th>start</th>
<th>end</th>
</tr>
</thead>
<tbody>
<tr>
<td>chrX</td>
<td>+</td>
<td>36M</td>
<td>36</td>
<td>3000326</td>
<td>3000361</td>
</tr>
<tr>
<td>chrX</td>
<td>-</td>
<td>36M</td>
<td>36</td>
<td>3001293</td>
<td>3001328</td>
</tr>
<tr>
<td>chrX</td>
<td>-</td>
<td>36M</td>
<td>36</td>
<td>3002790</td>
<td>3002825</td>
</tr>
<tr>
<td>chrX</td>
<td>-</td>
<td>36M</td>
<td>36</td>
<td>3016328</td>
<td>3016363</td>
</tr>
<tr>
<td>chrX</td>
<td>+</td>
<td>36M</td>
<td>36</td>
<td>3021161</td>
<td>3021196</td>
</tr>
</tbody>
</table>

... ...

SRR039211.1427139 chrX - 36M 36 166443604 166443639
SRR039211.447965 chrX - 36M 36 166445241 166445276
SRR039211.1352670 chrX - 36M 36 166446255 166446290
SRR039211.918096 chrX - 36M 36 166446315 166446350
SRR039211.67451 chrX - 36M 36 166446621 166446656

width njunc | uniqueHit
<integer> | <logical>
36 0 | FALSE
36 0 | FALSE
36 0 | FALSE
36 0 | TRUE
... ...
36 0 | FALSE
36 0 | FALSE
36 0 | FALSE
36 0 | TRUE

-------
seqinfo: 22 sequences from mm9 genome

> ctlGal

GAlignments object with 8276 alignments and 1 metadata column:

<table>
<thead>
<tr>
<th>seqnames</th>
<th>strand</th>
<th>cigar</th>
<th>qwidth</th>
<th>start</th>
<th>end</th>
</tr>
</thead>
<tbody>
<tr>
<td>chrX</td>
<td>+</td>
<td>36M</td>
<td>36</td>
<td>3028539</td>
<td>3028574</td>
</tr>
<tr>
<td>chrX</td>
<td>-</td>
<td>35M</td>
<td>35</td>
<td>3039791</td>
<td>3039825</td>
</tr>
<tr>
<td>chrX</td>
<td>-</td>
<td>36M</td>
<td>36</td>
<td>3084567</td>
<td>3084602</td>
</tr>
<tr>
<td>chrX</td>
<td>+</td>
<td>36M</td>
<td>36</td>
<td>3139109</td>
<td>3139144</td>
</tr>
</tbody>
</table>
SRR039214.5753635  chrX  +  36M  36  3139131  3139166
... ... ... ... ... ...
SRR039214.9045244  chrX  -  32M  32  166445788  166445819
SRR039214.2473565  chrX  -  21M  21  166445960  166445980
SRR039214.576581  chrX  -  36M  36  166446074  166446109
SRR039214.3756853  chrX  -  36M  36  166446781  166446816
SRR039214.2347055  chrX  +  36M  36  166650104  166650139

Note from the commands above that the RIP alignment files corresponding to the technical replicates are combined into a single `GAlignments` object [Aboyoun et al.]. The `uniqueHit` column from the output above indicate whether the read mapped to a single locus or multiple loci. The latter is commonly referred as the “multihits”. The ambiguous alignments arise from the repetitive elements or gene duplication events in the mammalian genomes. By default, the unique and multihits are flagged with binary TRUE and FALSE value in column `uniqueHit`, respectively. In the later step, each multihit will be assigned by the function `disambiguateMultihits` to a unique locus based on the posterior probabilities of the loci of being at the background or read enriched states of the two-state HMM (?? in the manuscript).

### 4 RIP-seq analysis

#### 4.1 High-level visualization of alignments at chromosomal level

After processing the alignment files, we could first visualize the alignments by calling `plotStrandedCoverage` to get a rough idea about the alignment quality and where the reads aggregate within each chromosome. `plotStrandedCoverage` first bins each chromosome by nonoverlapping windows of fixed size (1 kb in the example below) and plots the number of alignments fall within these bins across the whole chromosome. The function is implemented based on the vignette `GenomicRanges Use Cases` from the `GenomicRanges` package [Aboyoun et al.].

```r
> ripGR <- as(ripGal, "GRanges")
> ripGRList <- split(ripGR, seqnames(ripGR))
> # get only the nonempty chromosome
> ripGRList <- ripGRList[elementNROWS(ripGRList) != 0]
```
Here we first converted the GAlignments object ripGal and ctlGal to GenomicRangesList to have each list item represented by a chromosome. We then `lapply` the function to all nonempty chromosomes, which in our case is `chrX` for both RIP (left) and control (right). RIP represents the pooled alignments of the two technical replicates. Read count on + and - strand are displayed as blue and red bars on the positive and negative y-axis, respectively. Read counts are drawn to scale within a chromosome. Unlike ChIP-seq on double-stranded DNA, no symmetry is observed for the peak for the strand-specific sequencing data. However, considerable noise is observed within the Ezh2 -/- mutant library (right panel), which ideally should not have any aligned reads. This may also imply considerable noise within the wild type library and thus a high false discovery rate if the RIP regions were simply determined based on read counts. Please refer to Figure ?? and ?? in Supplementary Data for similar RIP-seq results on all chromosomes, which motivated the development of RIPSeeker.
4.2 ripSeek: the all-in-one function

The front-end main function ripSeek is in many cases the only function that users need to run to get the RIP predictions and most of the relevant information out of the alignment data. It requires a few important parameter settings:

```r
> # specify control name
> cNAME <- "SRR039214"
> # output file directory
> outDir <- file.path(getwd(), "RIPSeeker_vigenette_example_PRC2")
> # Parameters setting
> binSize <- NULL # set to NULL to automatically determine bin size
> minBinSize <- 10000 # min bin size in automatic bin size selection
> maxBinSize <- 10100 # max bin size in automatic bin size selection
> multicore <- TRUE # use multicore
> strandType <- "-" # set strand type to minus strand
> biomart <- "ENSEMBL_MART_ENSEMBL" # use archive to get ensembl 65
> biomart_dataset <- "mmusculus_gene_ensembl" # mouse dataset id name
> host <- "dec2011.archive.ensembl.org" # use ensembl 65 for annotation for mm9
> goAnno <- "org.Mm.eg.db" # GO annotation database
>
> # specify control name
> seekOut.PRC2 <- ripSeek(bamPath = bamFiles, cNAME = cNAME,
> reverseComplement = TRUE, genomeBuild = "mm9",
> strandType = strandType,
> uniqueHit = TRUE, assignMultihits = TRUE,
> rerunWithDisambiguatedMultihits = TRUE,
> binSize = binSize, minBinSize = minBinSize,
> maxBinSize = maxBinSize,
> biomart = biomart, host = host,
> biomart_dataset = biomart_dataset, goAnno = goAnno,
> multicore = multicore, outDir = outDir)
```

- `bamPath` saves the paths to the bam files;
- `cNAME` is a character string that distinguishes the control from the RIP among the bam files;
- `reverseComplement=TRUE` indicates that the reads were sequenced from the opposite strand of the original RNA molecule such that the original strand signs of the alignments are switched (i.e. + to - , - to +);
- `genomeBuild="mm9"` specifies the fact that the previous mouse genome (mm9/NCBIM37) build was used as reference for the alignments;
- `strandType` is set to "-" to search for transcripts only on the minus strand of chromosomes such as Xist, which is known to physically interact with PRC2 Zhao et al. [2010].
- `uniqueHit = TRUE` requires training HMM with only the unique hits.
- `assignMultihits = TRUE` enables `disambiguateMultihits` to assign each multihit to a unique locus based on the posterior probabilities derived from HMM.
• rerunWithDisambiguatedMultihits = TRUE tells RIPSeeker to retrain the HMM using the dataset with disambiguated multihits.

• binSize = NULL enables automatic bin size selection by the routine selectBinSize.

• minBinSize=10000, maxBinSize=12000 For demonstration pupose, we set the lower and upper bounds of the search space for the optimal bin size to 10 kb and 12 kb, respectively. Such choice may suffice if one just wants to quickly look for some known long noncoding RNA (lncRNA) such as Xist (22,843 nt) or Tsix (53,441 nt).

• biomart, host, biomaRt_dataset, goAnno are set to enable automatic online annotation of RIP predictions via annotateRIP, which makes use of the functions from ChIPpeakAnno [Zhu et al., 2010] and biomaRt [Durinck et al., 2005, 2009]. Since the genome we used for the alignments is mm9 not the most recent build mm10, we need to retrieve the archive annotation from Ensembl by setting the host and biomart. If the user uses the most recent assembly as reference genome for the alignments, then biomart should be set to "ensembl" and host may be omitted.

• multicore = TRUE enables parallel computing on each chromosome separately. It will greatly improve the computation time when used on a cluster.

• outDir the output directory to save the results (See Value section in ?ripSeek)

4.3 ripSeek outputs

The main function ripSeek returns a list which comprises four items outlined below. The user may find annotatedRIPGR the most useful. In most cases, the other three items can be considered as intermediate results.

> names(seekOut.PRC2)
> df <- elementMetadata(seekOut.PRC2$annotatedRIPGR$sigGRangesAnnotated)
> # order by eFDR
> df <- df[order(df$eFDR), ]
> # get top 100 predictions
> df.top100 <- head(df, 100)
> head(df.top100)
> # examine known PRC2-associated lncRNA transcripts
> df.top100[grep("Xist", df$external_gene_id), ]

• mainSeekOutputRIPA (inner) list containing three items:
  – nbhGRLList A GRangesList object of the HMM trained parameters for each chromosome on RIP:
  – alignGal, alignGalFiltered GAlignments objects of the RIP alignment outputs from combineAlignGals and disambiguateMultihits, respectively. The former may contain multiple alignments due the same reads whereas the latter contains a one-to-one mapping from read to alignment after disambiguating the multihits;

• mainSeekOutputCTL Same as mainSeekOutputRIP but for the control library;
• **RIPGRList** The peaks in GRangesList. Each list item represents the RIP peaks on a chromosome accompanied with statistical scores including (read) count, logOddScore, pval, pvalAdj, eFDR for the RIP and control (if available).

• **annotatedRIPGR** A list containing two items:
  – sigGRangesAnnotatedGRanges containing peaks and their scores (same as in RIPGRList) accompanied with genomic information retrieved from Ensembl database;
  – enrichedGOA data.frame containing enriched GO terms associated with the RIP peaks.

In addition, the (annotated) peaks and enriched GO annotations are saved in `outDir` as text files in tab-delimited (viewable with Excel) or GFF3 (General Feature Format 3) formats (\?rtracklayer:io for details for the formats).

```r
> list.files(outDir)
```

File (1) and (4) can be imported to a dedicated genome browser such as Integrative Genomic Viewer (IGV) Robinson et al. [2011] to visualize and interact with the putative RIP regions accompanied with all of the scores. Files (2) and (3) provide the most detailed information directly viewable in Excel. File (5) stores all of the intermediate results as RData for future reference or retrieval (with load command).

### 4.4 Visualization with UCSC browser

In addition, we could visualize the results using `viewRIP`, which launches online UCSC genome browser with uploaded custom track corresponding to the loci of RIP regions and scores (RIPScore, -log10(p-value), -log10(q-value), or -log10(eFDR)) generated from `ripSeek`. This task is accomplished seamlessly within the R console by making use of the available functions from `rtracklayer` Lawrence et al.

```r
> viewRIP(seekOut.PRC2$RIPGRList$chrX, +        seekOut.PRC2$mainSeekOutputRIP$alignGalFiltered, +        seekOut.PRC2$mainSeekOutputCTL$alignGalFiltered, scoreType="eFDR")
```

### 4.5 Compute RPKM and fold-change of RIP over control for known genes or transcripts

User may want to know the abundance of all of the known coding or noncoding genes or transcripts in their RIP libraries and how their expressions compare with the control. Given a list of single-end or paired-end read alignment files in BAM/SAM/BED format, the function `computeRPKM` computes the read counts and normalized read counts as expression of annotated transcripts in the unit of "reads per kilobase of exon per million mapped reads" (RPKM) or "fragments per kilobase of exon per million mapped reads" (FPKM), respectively. Furthermore, `rulebaseRIPSeek` computes the fold-change difference of gene expression in terms of RPKM/FPKM between the RIP and control libraries. `rulebaseRIPSeek` reports putative RIP genes/transcripts if their RPKM expression and the ratio of RPKM[RIP]/RPKM[control] (on either + or - strand) are above \( t_1 \) and \( t_2 \). By default, \( t_1 = 0.4 \) and \( t_2 = 3.0 \), consistent with the thresholds applied in original method Zhao et al. [2010].

---

1The “fragments” refers to the long fragment sequenced on both ends by the short read pairs.
5 RIP-seq analysis on CCNT1 data

As promised earlier, we will apply ripSeek to the CCNT1 data and annotate the de novo putative RIP regions with the latest online Ensembl annotation based on human genome reference NCBI37/hg19.
> goAnno <- "org.Hs.eg.db"  # GO annotation database
> # run main function ripSeek to predict RIP
> seekOut.CCNT1 <- ripSeek(bamPath = bamFiles, cNAME = cNAME,
+ reverseComplement = TRUE, genomeBuild = "hg19",
+ strandType = strandType,
+ uniqueHit = TRUE, assignMultihits = TRUE,
+ rerunWithDisambiguatedMultihits = TRUE,
+ binSize = binSize, minBinSize = minBinSize,
+ maxBinSize = maxBinSize,
+ biomart = biomart, goAnno = goAnno,
+ biomaRt_dataset = biomaRt_dataset,
+ multicore = multicore, outDir = outDir)
> df <- elementMetadata(seekOut.CCNT1$annotatedRIPGR$sigGRangesAnnotated)
> # order by eFDR
> df <- df[order(df$eFDR), ]
> # get top 100 predictions
> df.top20 <- head(df, 20)
> # examine known PRC2-associated lncRNA transcripts
> df.top20[grep("RN7SK", df$external_gene_id)[1], ]
> list.files(outDir)
> viewRIP(seekOut.CCNT1$RIPGRList$chr6, seekOut.CCNT1$mainSeekOutputRIP$alignGalFiltered, seekOut.CCNT1$mainSeekOutputCTL$alignGalFiltered, scoreType = "eFDR")
> # use biomart
> txDbName <- "biomart"
> biomart <- "ensembl"
> dataset <- "hsapiens_gene_ensembl"
> # compute transcript abundance in RIP
> ripRPKM <- computeRPKM(bamFiles = bamFiles[1],
+ dataset = dataset, moreGeneInfo = TRUE, justRPKM = FALSE,
+ idType = "ensembl_transcript_id", txDbName = txDbName,
+ biomart = biomart, by = "tx")
> # compute transcript abundance in RIP and control as well as foldchange in RIP over control
> rulebase.results <- rulebaseRIPSeek(bamFiles = bamFiles, cNAME = cNAME, myMin = 1,
+ featureGRanges = ripRPKM$featureGRanges,
+ biomart = biomart, dataset = dataset)
> head(ripRPKM$rpkmDF)
> df <- rulebase.results$rpkmDF
> df <- df[order(df$foldchange, decreasing = TRUE), ]
> # top 10 transcripts
> head(df, 10)

6 Session Info

> sessionInfo()

R version 3.6.1 (2019-07-05)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 18.04.3 LTS
Matrix products: default
BLAS: /home/biocbuild/bbs-3.10-bioc/R/lib/libRblas.so
LAPACK: /home/biocbuild/bbs-3.10-bioc/R/lib/libRlapack.so

locale:
[1] LC_CTYPE=en_US.UTF-8 LC_NUMERIC=C
[3] LC_TIME=en_US.UTF-8 LC_COLLATE=C
[5] LC_MONETARY=en_US.UTF-8 LC_MESSAGES=en_US.UTF-8
[7] LC_PAPER=en_US.UTF-8 LC_NAME=C
[9] LC_ADDRESS=C LC_TELEPHONE=C

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

other attached packages:
[1] RIPSeeker_1.26.0 rtracklayer_1.46.0
[3] GenomicAlignments_1.22.0 Rsamtools_2.2.0
[5] Biostrings_2.54.0 XVector_0.26.0
[7] SummarizedExperiment_1.16.0 DelayedArray_0.12.0
[9] BiocParallel_1.20.0 matrixStats_0.55.0
[11] Biobase_2.46.0 GenomicRanges_1.38.0
[13] GenomeInfoDb_1.22.0 IRanges_2.20.0
[15] S4Vectors_0.24.0 BiocGenerics_0.32.0

loaded via a namespace (and not attached):
[1] zlibbioc_1.32.0 lattice_0.20-38 tools_3.6.1
[4] grid_3.6.1 Matrix_1.2-17 GenomeInfoDbData_1.2.2
[7] bitops_1.0-6 RCurl_1.95-4.12 compiler_3.6.1
[10] XML_3.98-1.20

References


