Managing and analyzing multiple NGS samples with Bioconductor bamViews objects: application to RNA-seq

VJ Carey

April 21, 2026

Contents

1 Introduction

We consider a lightweight approach to Bioconductor-based management and interrogation of multiple samples to which NGS methods have been applied.

The basic data store is the binary SAM (BAM) format (Li et al., 2009). This format is widely used in the 1000 genomes project, and transformations between SAM/BAM and output formats of various popular alignment programs are well-established. Bioconductor's *Rsamtools* package allows direct use of important SAM data interrogation facilities from R.

2 Basic design

A collection of NGS samples is represented through the associated set of BAM files and BAI index files. These can be stored in the inst/bam folder of an R package to facilitate documented programmatic access through R file navigation facilities, or the BAM/BAI files can be accessed through arbitrary path or URL references.

The bamViews class is defined to allow reliable fine-grained access to the NGS data along with relevant metadata. A bamViews instance contains access path information for a set of related BAM/BAI files, along with sample metadata and an optional specification of genomic ranges of interest.

A key design aspect of the bamViews class is preservation of semantics of the `X[G, S]` idiom familiar from *ExpressionSet* objects for management of multiple microarrays. With `ExpressionSet` instances, `G` is a predicate specifying selection of microarray probes of interest. With *bamViews* instances, `G` is a predicate specifying selection of genomic features of interest. At present, for *bamViews*, selection using `G` involves ranges of genomic coordinates.

3 Illustration

Data from four samples from a yeast RNA-seq experiment (two wild type, two 'RLP' mutants) are organized in the *leeBamViews* package. The data are collected to allow regeneration of aspects of Figure 8 of Lee et al. (2008). We obtained all reads between bases 800000 and 900000 of yeast chromosome XIII.

We have not yet addressed durable serialization of manager objects, so the *bamViews* instance is created on the fly.

```
> suppressPackageStartupMessages({
+ library(leeBamViews)  # bam files stored in package
+ library(S4Vectors)
+ })
> bpaths = dir(system.file("bam", package="leeBamViews"), full=TRUE, patt="bam$")
> #
```

```
> # extract genotype and lane information from filenames
> #
> gt = gsub(".*/", "", bpaths)
> gt = gsub("_.*", "", gt)
> lane = gsub(".*(.)$", "\\1", gt)
> geno = gsub(".$", "", gt)
> #
> # format the sample-level information appropriately
> #
> pd = DataFrame(geno=geno, lane=lane, row.names=paste(geno,lane,sep="."))
> prd = new("DFrame")  # protocol data could go here
> #
> # create the views object, adding some arbitrary experiment-level information
> #
> bs1 = BamViews(bamPaths=bpaths, bamSamples=pd,
+         bamExperiment=list(annotation="org.Sc.sgd.db"))
> bs1

BamViews dim: 0 ranges x 8 samples
names: isowt.5 isowt.6 ... xrn.1 xrn.2
detail: use bamPaths(), bamSamples(), bamRanges(), ...

> #
> # get some sample-level data
> #
> bamSamples(bs1)$geno

[1] "isowt" "isowt" "rlp"   "rlp"   "ssr"   "ssr"   "xrn"   "xrn"
```

We would like to operate on specific regions of chr XIII for all samples. Note that the aligner in use (bowtie) employed "Scchr13" to refer to this chromosome. We add a *GRanges* instance to the view to identify the region of interest.

```
> START=c(861250, 863000)
> END=c(862750, 864000)
> exc = GRanges(seqnames="Scchr13", IRanges(start=START, end=END), strand="+")
> bamRanges(bs1) = exc
> bs1

BamViews dim: 2 ranges x 8 samples
names: isowt.5 isowt.6 ... xrn.1 xrn.2
detail: use bamPaths(), bamSamples(), bamRanges(), ...
```

A common operation will be to extract coverage information. We use a transforming method, `readGAlignments`, from the *GenomicAlignments* package to extract reads and metadata for each region and each sample.

```
> library(GenomicAlignments)
> covex =
+   RleList(lapply(bamPaths(bs1), function(x)
+                  coverage(readGAlignments(x))[["Scchr13"]]))
> names(covex) = gsub(".bam$", "", basename(bamPaths(bs1)))
> head(covex, 3)

RleList of length 3
$isowt5_13e
integer-Rle of length 924429 with 21819 runs
  Lengths: 799974      2      2      1      6 ...     10      7      1  24399
  Values :      0      1      2      3      4 ...      4      3      2      0

$isowt6_13e
integer-Rle of length 924429 with 21799 runs
  Lengths: 799976      2      3     14     13 ...     17      1      4  24394
  Values :      0      1      2      3      4 ...      3      2      1      0

$rlp5_13e
integer-Rle of length 924429 with 23037 runs
  Lengths: 799974      2      6     25      3 ...      4      2     30  24397
  Values :      0      1      2      3      4 ...      3      2      1      0
```

Let's visualize what we have so far. We use *GenomeGraphs* and add some supporting software. Get a copy from an archive if you want to run this code.

```
> library(GenomeGraphs)
> cov2baseTrack = function(rle, start, end,
+    dp = DisplayPars(type="l", lwd=0.5, color="black"),
+    countTx=function(x)log10(x+1)) {
+  require(GenomeGraphs)
+  if (!is(rle, "Rle")) stop("requires instance of Rle")
+  dat = runValue(rle)
+  loc = cumsum(runLength(rle))
+  ok = which(loc >= start & loc <= end)
+  makeBaseTrack(base = loc[ok], value=countTx(dat[ok]),
+     dp=dp)
+ }
> trs = lapply(covex, function(x) cov2baseTrack(x, START[1], END[1],
```

```
+    countTx = function(x)pmin(x, 80)))
> ac = as.character
> names(trs) = paste(ac(bamSamples(bs1)$geno), ac(bamSamples(bs1)$lane), sep="")
> library(biomaRt)
> mart = useMart("ensembl", "scerevisiae_gene_ensembl")
> gr = makeGeneRegion(START, END, chromosome="XIII",
+   strand="+", biomart=mart, dp=DisplayPars(plotId=TRUE,
+   idRotation=0, idColor="black"))
> trs[[length(trs)+1]] = gr
> trs[[length(trs)+1]] = makeGenomeAxis()

> print( gdPlot( trs, minBase=START[1], maxBase=END[1]) )
```

We can encapsulate this to something like:

```
> plotStrains = function(bs, query, start, end, snames, mart, chr, strand="+") {
+  filtbs = bs[query, ]
+  cov = lapply(filtbs, coverage)
+  covtrs = lapply(cov, function(x) cov2baseTrack(x[[1]], start, end,
+    countTx = function(x) pmin(x,80)))
+  names(covtrs) = snames
+  gr = makeGeneRegion(start, end, chromosome=chr,
+        strand=strand, biomart=mart, dp=DisplayPars(plotId=TRUE,
+        idRotation=0, idColor="black"))
+  grm = makeGeneRegion(start, end, chromosome=chr,
+        strand="-", biomart=mart, dp=DisplayPars(plotId=TRUE,
+        idRotation=0, idColor="black"))
+  covtrs[[length(covtrs)+1]] = gr
+  covtrs[[length(covtrs)+1]] = makeGenomeAxis()
+  covtrs[[length(covtrs)+1]] = grm
+  gdPlot( covtrs, minBase=start, maxBase=end )
+ }
```

4 Comparative counts in a set of regions of interest

4.1 Counts in a regular partition

The supplementary information for the Lee paper includes data on unnannotated transcribed regions reported in other studies. We consider the study of David et al., confining attention to chromosome XIII. If you wanted to study their intervals you could use code like:

```
> data(leeUnn)
> names(leeUnn)
> leeUnn[1:4,1:8]
> table(leeUnn$study)
> l13 = leeUnn[ leeUnn$chr == 13, ]
> l13d = na.omit(l13[ l13$study == "David", ])
> d13r = GRanges(seqnames="Scchr13", IRanges( l13d$start, l13d$end ),
+   strand=ifelse(l13d$strand==1, "+", ifelse(l13d$strand=="0", "*", "-")))
> elementMetadata(d13r)$name = paste("dav13x", 1:length(d13r), sep=".")
> bamRanges(bs1) = d13r
> d13tab = tabulateReads( bs1 )
```

but our object `bs1` is too restricted in its coverage. Instead, we illustrate with a small set of subintervals of the basic interval in use:

```
> myrn = GRanges(seqnames="Scchr13",
+   IRanges(start=seq(861250, 862750, 100), width=100), strand="+")
> elementMetadata(myrn)$name = paste("til", 1:length(myrn), sep=".")
> bamRanges(bs1) = myrn
> tabulateReads(bs1, "+")
```

	til.1	til.2	til.3	til.4	til.5	til.6	til.7	til.8	til.9	til.10
start	861250	861350	861450	861550	861650	861750	861850	861950	862050	862150
end	861349	861449	861549	861649	861749	861849	861949	862049	862149	862249
isowt.5	1	1	3	6	2	7	299	605	408	380
isowt.6	2	6	9	12	7	4	306	666	458	382
rlp.5	1	5	65	53	36	11	158	247	186	145
rlp.6	3	2	47	48	37	16	123	238	163	159
ssr.1	2	6	35	27	21	8	423	700	541	496
ssr.2	2	6	43	37	26	13	443	839	616	509
xrn.1	7	8	75	78	24	5	180	446	357	288
xrn.2	4	9	96	110	31	8	225	611	465	356

	til.11	til.12	til.13	til.14	til.15	til.16
start	862250	862350	862450	862550	862650	862750
end	862349	862449	862549	862649	862749	862849
isowt.5	482	554	895	631	643	702
isowt.6	446	517	870	689	691	701
rlp.5	174	180	316	251	239	277
rlp.6	190	215	336	270	269	281
ssr.1	573	596	966	737	669	771
ssr.2	576	606	987	775	742	811
xrn.1	349	484	678	549	396	342
xrn.2	430	578	837	643	453	420

4.2 Counts in annotated intervals: genes

We can use Bioconductor annotation resources to acquire boundaries of yeast genes on our subregion of chromosome 13.

In the following chunk we generate annotated ranges of genes on the Watson strand.

```
> library(org.Sc.sgd.db)
> library(IRanges)
> c13g = get("13", revmap(org.Sc.sgdCHR))  # all genes on chr13
> c13loc = unlist(mget(c13g, org.Sc.sgdCHRLOC))  # their 'start' addresses
> c13locend = unlist(mget(c13g, org.Sc.sgdCHRLOCEND))
> c13locp = c13loc[c13loc>0]     # confine attention to + strand
> c13locendp = c13locend[c13locend>0]
> ok = !is.na(c13locp) & !is.na(c13locendp)
> c13pr = GRanges(seqnames="Scchr13", IRanges(c13locp[ok], c13locendp[ok]),
+     strand="+")   # store and clean up names
> elementMetadata(c13pr)$name = gsub(".13$", "", names(c13locp[ok]))
> c13pr

GRanges object with 324 ranges and 1 metadata column:
        seqnames        ranges strand |        name
           <Rle>     <IRanges>  <Rle> | <character>
    [1]  Scchr13 267174-267800      + |     YML001W
    [2]  Scchr13 264541-266754      + |     YML002W
    [3]  Scchr13 263483-264355      + |     YML003W
    [4]  Scchr13 260221-261609      + |     YML005W
    [5]  Scchr13 253848-255800      + |     YML007W
    ...      ...           ...    ... .         ...
  [320]  Scchr13 923015-923494      + |     ARS1335
  [321]  Scchr13 667324-667346      + |        ETC5
  [322]  Scchr13 158887-159277      + |   ARS1307.5
  [323]  Scchr13   23564-26578      + |   YNCM0001W
  [324]  Scchr13 480924-481187      + |   YMR106W-A
  -------
  seqinfo: 1 sequence from an unspecified genome; no seqlengths

> c13pro = c13pr[ order(ranges(c13pr)), ]
```

That's the complete set of genes on the Watson strand of chromosome XIII. In the *leeBamViews* package, we do not have access to all these, but only those lying in a 100kb interval.

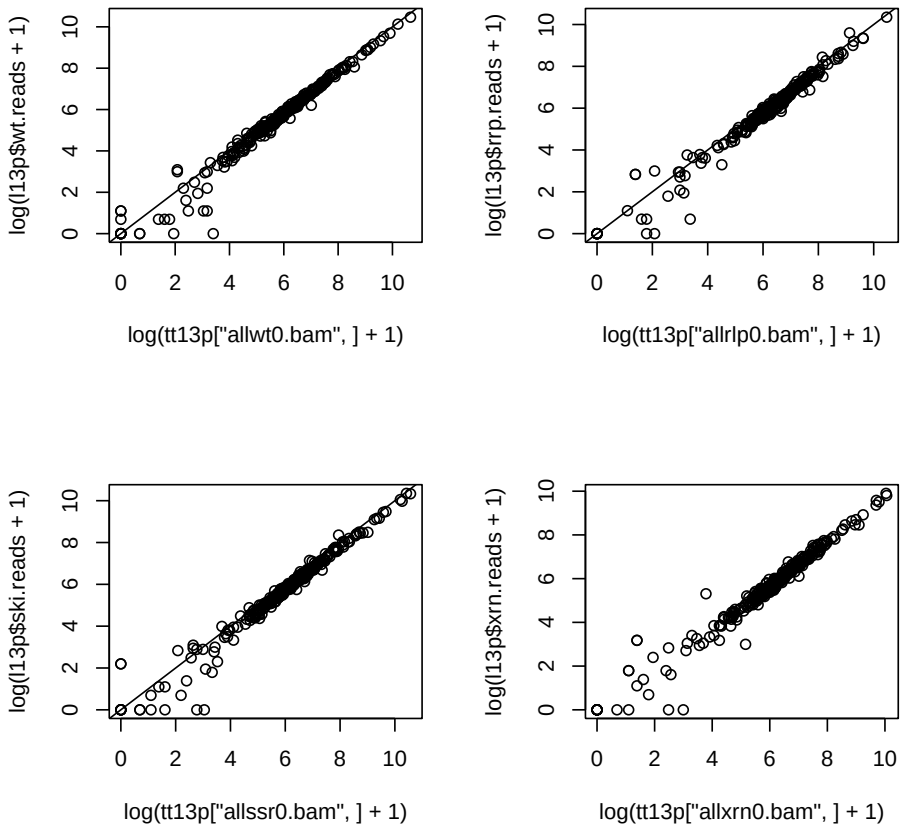
```
> lim = GRanges(seqnames="Scchr13", IRanges(800000,900000), strand="+")
> c13prol = c13pro[ which(overlapsAny(c13pro , lim) ), ]
```

Now that we have a set of annotation-based genomic regions, we can tabulate read counts lying in those regions and obtain an annotated matrix.

```
> bamRanges(bs1) = c13prol
> annotab = tabulateReads(bs1, strandmarker="+")
```

5 Larger scale sanity check

The following plot compares read counts published with the Lee et al. (2008) paper to those computed by the methods sketched here, for all regions noted on the plus strand of chromosome XIII. Exact correspondence is not expected because of different approaches to read filtering.



6 Statistical analyses of differential expression

6.1 Using edgeR

Statistical analysis of read counts via negative binomial distributions with moderated dispersion is developed in Robinson and Smyth (2008). The *edgeR* differential expression statistics are computed using regional read counts, and total library size plays a role. We compute total read counts directly (the operation can be somewhat slow for very large BAM files):

```
> totcnts = totalReadCounts(bs1)
```

In the following demonstration, we will regard multiple lanes from the same genotype as replicates. This is probably inappropriate for this method; the original authors tested for lane effects and ultimately combined counts across lanes within strain.

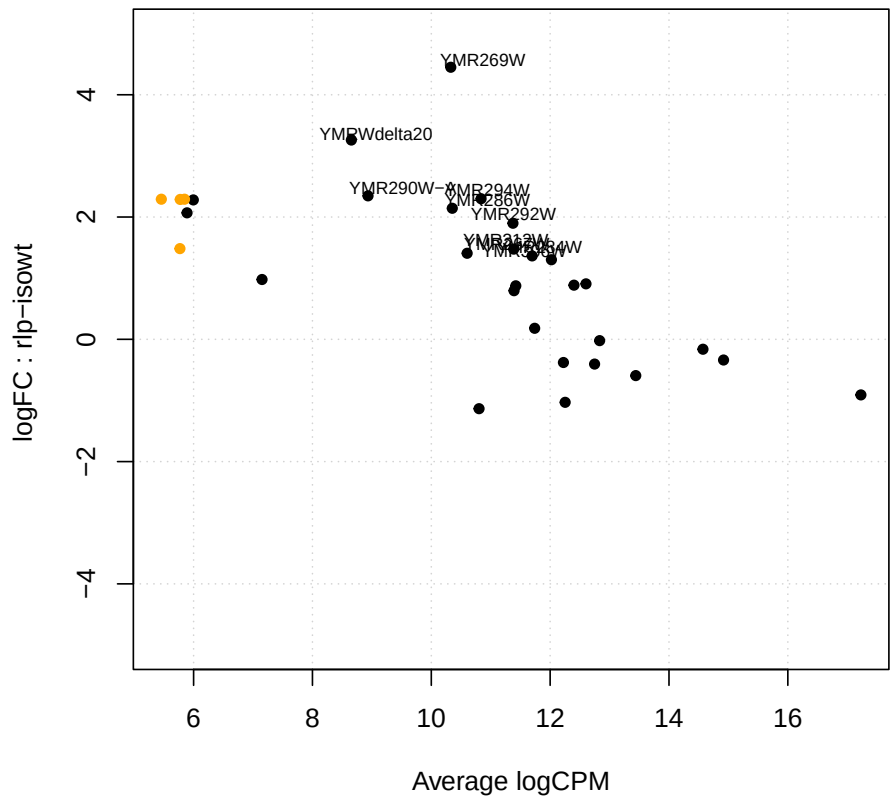
```
> library(edgeR)
> #
> # construct an edgeR container for read counts, including
> #   genotype and region (gene) metadata
> #
> dgel1 = DGEList( counts=t(annotab)[,-c(1,2)],
+    group=factor(bamSamples(bs1)$geno),
+    lib.size=totcnts, genes=colnames(annotab))
> #
> # compute a dispersion factor for the negative binomial model
> #
> cd = estimateCommonDisp(dgel1)
> #
> # test for differential expression between two groups
> # for each region
> #
> et12 = exactTest(cd)
> #
> # display statistics for the comparison
> #
> tt12 = topTags(et12)
> tt12

Comparison of groups:  rlp-isowt
                  genes    logFC    logCPM       PValue          FDR
YMR269W         YMR269W 4.428960 10.868737 7.684011e-10 2.382043e-08
YMRWdelta20 YMRWdelta20 3.218156  9.068611 2.747347e-05 4.258388e-04
YMR294W         YMR294W 2.293633 10.942933 3.332007e-04 3.443074e-03
```

```
YMR286W         YMR286W 2.136147 10.944960 9.566131e-04 7.413751e-03
YMR290W-A     YMR290W-A 2.324579  9.523028 1.233201e-03 7.645844e-03
YMR292W         YMR292W 1.895158 11.383960 2.157942e-03 1.114937e-02
YMR312W         YMR312W 1.472817 11.254803 1.643625e-02 7.278909e-02
YMR284W         YMR284W 1.362186 11.820508 2.444618e-02 9.201598e-02
YMR267W         YMR267W 1.403464 11.272892 2.671432e-02 9.201598e-02
YMR306W         YMR306W 1.300375 11.565562 3.064195e-02 9.499006e-02
```

An analog of the "MA-plot" familiar from microarray studies is available for this analysis. The 'concentration' is the log proportion of reads present in each gene, and the "log fold change" is the model-based estimate of relative abundance. In the following display we label the top 10 genes (those with smallest FDR).

```
> plotSmear(cd, cex=.8, ylim=c(-5,5))
> text(tt12$table$logCPM, tt12$table$logFC+.15, as.character(
+   tt12$table$genes), cex=.65)
```



7 Summary

- The BAM format provides reasonably compact and comprehensive information about a alignments of short reads obtained in a sequencing experiment. samtools utilities permit efficient random access to read collections of interest.
- *Rsamtools* brings samtools functionality into R, principally through the `scanBam` method, which is richly parameterized so that many details of access to and filtering of reads from BAM files can be controlled in R.
- *Rsamtools* defines the *bamViews* container for management of collections of BAM files. Read data are managed external to R; data on aligned reads can be imported efficiently, and "streaming read" models for scanning large collections of reads can be used. Many embarrassingly parallel operations can be accomplished concurrently using *multicore* or similar packages.
- The *leeBamViews* package provides small excerpts from BAM files generated after bowtie alignment of FASTQ records available through the NCBI short read archives. These excerpts can be analyzed using code shown in this vignette.
- After the count data have been generated, various approaches to inference on differential expression are available. We consider the moderated negative binomial models of *edgeR* above; more general variance modeling is available in the developmental *DESeq* package.

References

Albert Lee, Kasper Daniel Hansen, James Bullard, Sandrine Dudoit, Gavin Sherlock, and Michael Snyder. Novel low abundance and transient rnas in yeast revealed by tiling microarrays and ultra high–throughput sequencing are not conserved across closely related yeast species. *PLoS Genet*, 4(12):e1000299, Dec 2008. doi: 10.1371/journal.pgen.1000299.t002.

Heng Li, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, Richard Durbin, and 1000 Genome Project Data Processing Subgroup. The sequence alignment/map format and samtools. *Bioinformatics*, 25(16): 2078–9, Aug 2009. doi: 10.1093/bioinformatics/btp352.

Mark D Robinson and Gordon K Smyth. Small-sample estimation of negative binomial dispersion, with applications to sage data. *Biostatistics (Oxford, England)*, 9(2):321–32, Apr 2008. doi: 10.1093/biostatistics/kxm030. URL `http://biostatistics.oxfordjournals.org/cgi/content/full/9/2/321`.

8 Session data

```
> sessionInfo()

R version 4.6.0 RC (2026-04-17 r89917)
Platform: x86_64-pc-linux-gnu
Running under: Ubuntu 24.04.4 LTS

Matrix products: default
BLAS:   /home/biocbuild/bbs-3.24-bioc/R/lib/libRblas.so
LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.12.0  LAPACK version 3.12.0

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_GB              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
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

time zone: America/New_York
tzcode source: system (glibc)

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

other attached packages:
 [1] edgeR_4.9.8                 limma_3.67.1
 [3] org.Sc.sgd.db_3.22.0        AnnotationDbi_1.73.1
 [5] GenomicAlignments_1.47.0    SummarizedExperiment_1.41.1
 [7] MatrixGenerics_1.23.0       matrixStats_1.5.0
 [9] leeBamViews_1.47.0          BSgenome_1.79.1
[11] rtracklayer_1.71.3          BiocIO_1.21.0
[13] Rsamtools_2.27.2            Biostrings_2.79.5
[15] XVector_0.51.0              GenomicRanges_1.63.2
[17] IRanges_2.45.0              S4Vectors_0.49.2
[19] Seqinfo_1.1.0               Biobase_2.71.0
[21] BiocGenerics_0.57.1         generics_0.1.4

loaded via a namespace (and not attached):
 [1] SparseArray_1.11.13 bitops_1.0-9        RSQLite_2.4.6
 [4] lattice_0.22-9      grid_4.6.0          fastmap_1.2.0
```

```
 [7] blob_1.3.0          Matrix_1.7-5        cigarillo_1.1.0
[10] restfulr_0.0.16     DBI_1.3.0           httr_1.4.8
[13] XML_3.99-0.23       codetools_0.2-20    abind_1.4-8
[16] cli_3.6.6           rlang_1.2.0         crayon_1.5.3
[19] bit64_4.6.0-1       cachem_1.1.0        DelayedArray_0.37.1
[22] yaml_2.3.12         otel_0.2.0          S4Arrays_1.11.1
[25] tools_4.6.0         parallel_4.6.0      BiocParallel_1.45.0
[28] memoise_2.0.1       locfit_1.5-9.12     curl_7.0.0
[31] vctrs_0.7.3         R6_2.6.1            png_0.1-9
[34] KEGGREST_1.51.1     bit_4.6.0           pkgconfig_2.0.3
[37] statmod_1.5.1       rjson_0.2.23        compiler_4.6.0
[40] RCurl_1.98-1.18
```