

# An Introduction to *GenomeInfoDb*

*Martin Morgan, Hervé Pagès, Marc Carlson, Sonali Arora*

Modified: 26 June, 2020. Compiled: April 5, 2021

## Contents

1	Introduction . . . . .	1
2	Functionality for all existing organisms. . . . .	2
2.1	genomeStyles . . . . .	2
2.2	extractSeqlevels . . . . .	3
2.3	extractSeqlevelsByGroup . . . . .	3
2.4	seqlevelsStyle . . . . .	3
2.5	seqlevelsInGroup . . . . .	3
2.6	orderSeqlevels . . . . .	4
2.7	rankSeqlevels . . . . .	4
2.8	mapSeqlevels . . . . .	5
2.9	renameSeqlevels . . . . .	5
2.10	dropSeqlevels . . . . .	6
2.11	keepSeqlevels . . . . .	7
2.12	keepStandardChromosomes . . . . .	7
3	Seqinfo objects . . . . .	8
4	Examples . . . . .	11
4.1	converting seqlevel styles (eg:UCSC to NCBI) . . . . .	11
4.2	converting styles and removing unwanted seqlevels . . . . .	11
5	Session Information . . . . .	12

## 1 Introduction

---

The *GenomeInfoDb* provides an interface to access seqlevelsStyles (such as UCSC, NCBI, Ensembl) and their supported mappings for organisms. For instance, for Homo sapiens, seqlevelsStyle "UCSC" maps to "chr1", "chr2", ..., "chrX", "chrY". The section below introduces these functions with examples.

## 2 Functionality for all existing organisms

### 2.1 genomeStyles

The `genomeStyles` lists out for each organism, the seqlevelsStyles and their mappings.

```
seqmap <- genomeStyles()
head(seqmap,n=2)

## $Arabidopsis_thaliana
##   circular auto sex NCBI TAIR9 Ensembl
## 1  FALSE  TRUE FALSE  1  Chr1      1
## 2  FALSE  TRUE FALSE  2  Chr2      2
## 3  FALSE  TRUE FALSE  3  Chr3      3
## 4  FALSE  TRUE FALSE  4  Chr4      4
## 5  FALSE  TRUE FALSE  5  Chr5      5
## 6   TRUE FALSE FALSE MT  ChrM      Mt
## 7   TRUE FALSE TRUE  Pltd ChrC      Pt
##
## $Caenorhabditis_elegans
##   circular auto sex NCBI UCSC Ensembl
## 1  FALSE  TRUE FALSE  I  chrI      I
## 2  FALSE  TRUE FALSE II  chrII     II
## 3  FALSE  TRUE FALSE III chrIII    III
## 4  FALSE  TRUE FALSE IV  chrIV     IV
## 5  FALSE  TRUE FALSE  V  chrV      V
## 6  FALSE FALSE TRUE  X  chrX      X
## 7   TRUE  TRUE FALSE MT  chrM     MtDNA
```

Organism's supported by GenomeInfoDb can be found by :

```
names(genomeStyles())

## [1] "Arabidopsis_thaliana"      "Caenorhabditis_elegans"
## [3] "Canis_familiaris"         "Cyanidioschyzon_merolae"
## [5] "Drosophila_melanogaster"   "Homo_sapiens"
## [7] "Mus_musculus"             "Oryza_sativa"
## [9] "Populus_trichocarpa"      "Rattus_norvegicus"
## [11] "Saccharomyces_cerevisiae" "Zea_mays"
```

If one knows the organism one is interested in, then we can directly access the information for the given organism along. Each function accepts an argument called `species` which as "genus species", the default is "Homo sapiens". In the following example we list out only the first five entries returned by the code snippet.

```
head(genomeStyles("Homo_sapiens"),5)

##   circular auto sex NCBI UCSC dbSNP Ensembl
## 1  FALSE TRUE FALSE  1 chr1  ch1      1
## 2  FALSE TRUE FALSE  2 chr2  ch2      2
## 3  FALSE TRUE FALSE  3 chr3  ch3      3
## 4  FALSE TRUE FALSE  4 chr4  ch4      4
## 5  FALSE TRUE FALSE  5 chr5  ch5      5
```

We can also check if a given style is supported by GenomeInfoDb for a given species. For example, if we want to know if "UCSC" mapping is supported for "Homo sapiens" we can ask :

```
"UCSC" %in% names(genomeStyles("Homo_sapiens"))  
## [1] TRUE
```

## 2.2 extractSeqlevels

We can also extract the desired seqlevelsStyle from a given organism using the `extractSeqlevels`

```
extractSeqlevels(species="Arabidopsis_thaliana", style="NCBI")  
## [1] "1" "2" "3" "4" "5" "MT" "Pltd"
```

## 2.3 extractSeqlevelsByGroup

We can also extract the desired seqlevelsStyle from a given organism based on a group ( Group - 'auto' denotes autosomes, 'circular' denotes circular chromosomes and 'sex' denotes sex chromosomes; the default is all chromosomes are returned).

```
extractSeqlevelsByGroup(species="Arabidopsis_thaliana", style="NCBI",  
                        group="auto")  
## [1] "1" "2" "3" "4" "5"
```

## 2.4 seqlevelsStyle

We can find the seqname Style for a given character vector by using the `seqlevelsStyle`

```
seqlevelsStyle(paste0("chr", c(1:30)))  
## [1] "UCSC"  
seqlevelsStyle(c("2L", "2R", "X", "Xhet"))  
## [1] "NCBI"
```

## 2.5 seqlevelsInGroup

We can also subset a given character vector containing seqnames using the `seqlevelsInGroup`. We currently support 3 groups: 'auto' for autosomes, 'sex' for allosomes/sex chromosomes and circular for 'circular' chromosomes. The user can also provide the style and species they are working with. In the following examples, we extract the sex, auto and circular chromosomes for Homo sapiens :

```
newchr <- paste0("chr", c(1:22, "X", "Y", "M", "1_gl000192_random", "4_ctg9_hap1"))
seqlevelsInGroup(newchr, group="sex")

## [1] "chrX" "chrY"

seqlevelsInGroup(newchr, group="auto")

## [1] "chr1" "chr2" "chr3" "chr4" "chr5" "chr6" "chr7" "chr8" "chr9"
## [10] "chr10" "chr11" "chr12" "chr13" "chr14" "chr15" "chr16" "chr17" "chr18"
## [19] "chr19" "chr20" "chr21" "chr22"

seqlevelsInGroup(newchr, group="circular")

## [1] "chrM"

seqlevelsInGroup(newchr, group="sex", "Homo_sapiens", "UCSC")

## [1] "chrX" "chrY"
```

if we have a vector containing seqnames and we want to verify the species and style for them , we can use:

```
seqnames <- c("chr1", "chr9", "chr2", "chr3", "chr10")
all(seqnames %in% extractSeqlevels("Homo_sapiens", "UCSC"))

## [1] TRUE
```

## 2.6 orderSeqlevels

The `orderSeqlevels` can return the order of a given character vector which contains seqnames. In the following example, we show how you can find the order for a given seqnames character vector.

```
seqnames <- c("chr1", "chr9", "chr2", "chr3", "chr10")
orderSeqlevels(seqnames)

## [1] 1 3 4 2 5

seqnames[orderSeqlevels(seqnames)]

## [1] "chr1" "chr2" "chr3" "chr9" "chr10"
```

## 2.7 rankSeqlevels

The `rankSeqlevels` can return the rank of a given character vector which contains seqnames. In the following example, we show how you can find the rank for a given seqnames character vector.

```
seqnames <- c("chr1", "chr9", "chr2", "chr3", "chr10")
rankSeqlevels(seqnames)

## [1] 1 4 2 3 5
```

## 2.8 mapSeqlevels

Returns a matrix with 1 column per supplied sequence name and 1 row per sequence renaming map compatible with the specified style. If `best.only` is `TRUE` (the default), only the "best" renaming maps (i.e. the rows with less NAs) are returned.

```
mapSeqlevels(c("chrII", "chrIII", "chrM"), "NCBI")

## chrII chrIII chrM
##  "II"  "III"  "MT"
```

We also have several seqlevel utility functions. Let us construct a basic `GRanges` and show how these functions can be used. .

```
gr <- GRanges(paste0("ch", 1:35), IRanges(1:35, width=5))
gr

## GRanges object with 35 ranges and 0 metadata columns:
##      seqnames      ranges strand
##      <Rle> <IRanges> <Rle>
## [1]      ch1         1-5      *
## [2]      ch2         2-6      *
## [3]      ch3         3-7      *
## [4]      ch4         4-8      *
## [5]      ch5         5-9      *
## ...      ...         ...      ...
## [31]     ch31        31-35      *
## [32]     ch32        32-36      *
## [33]     ch33        33-37      *
## [34]     ch34        34-38      *
## [35]     ch35        35-39      *
## -----
## seqinfo: 35 sequences from an unspecified genome; no seqlengths
```

As you can see, we have "ch" instead of "chr" for chromosome names. We can use `renameSeqlevels` to change the "ch" to "chr"

## 2.9 renameSeqlevels

As the first argument - it takes the object whose seqlevels we need to change, and as the second argument it takes a named vector which has the changes.

```
newnames <- paste0("chr", 1:35)
names(newnames) <- paste0("ch", 1:35)
head(newnames)

##      ch1      ch2      ch3      ch4      ch5      ch6
## "chr1" "chr2" "chr3" "chr4" "chr5" "chr6"

gr <- renameSeqlevels(gr, newnames)
gr

## GRanges object with 35 ranges and 0 metadata columns:
```

```
##      seqnames      ranges strand
##      <Rle> <IRanges> <Rle>
## [1]   chr1        1-5      *
## [2]   chr2        2-6      *
## [3]   chr3        3-7      *
## [4]   chr4        4-8      *
## [5]   chr5        5-9      *
## ...      ...      ...      ...
## [31]  chr31       31-35      *
## [32]  chr32       32-36      *
## [33]  chr33       33-37      *
## [34]  chr34       34-38      *
## [35]  chr35       35-39      *
## -----
## seqinfo: 35 sequences from an unspecified genome; no seqlengths
```

Humans have just 22 primary chromosomes - but here we have some extra seqlevels which we want to remove - there are several ways we can achieve this:

## 2.10 dropSeqlevels

Here the second argument is the seqlevels that you want to drop. Because these seqlevels are in use (i.e. have ranges on them), the ranges on these sequences need to be removed before the seqlevels can be dropped. We call this *pruning*. The `pruning.mode` argument controls how to prune `gr`. Unlike for list-like objects (e.g. `GRangesList`) for which pruning can be done in various ways, pruning a `GRanges` object is straightforward and achieved by specifying `pruning.mode="coarse"`.

```
dropSeqlevels(gr, paste0("chr",23:35), pruning.mode="coarse")

## GRanges object with 22 ranges and 0 metadata columns:
##      seqnames      ranges strand
##      <Rle> <IRanges> <Rle>
## [1]   chr1        1-5      *
## [2]   chr2        2-6      *
## [3]   chr3        3-7      *
## [4]   chr4        4-8      *
## [5]   chr5        5-9      *
## ...      ...      ...      ...
## [18]  chr18       18-22      *
## [19]  chr19       19-23      *
## [20]  chr20       20-24      *
## [21]  chr21       21-25      *
## [22]  chr22       22-26      *
## -----
## seqinfo: 22 sequences from an unspecified genome; no seqlengths
```

## 2.11 keepSeqlevels

Here the second argument is the seqlevels that you want to keep.

```
keepSeqlevels(gr, paste0("chr",1:22), pruning.mode="coarse")

## GRanges object with 22 ranges and 0 metadata columns:
##      seqnames      ranges strand
##      <Rle> <IRanges>  <Rle>
##   [1]   chr1         1-5      *
##   [2]   chr2         2-6      *
##   [3]   chr3         3-7      *
##   [4]   chr4         4-8      *
##   [5]   chr5         5-9      *
##   ...     ...           ...    ...
##  [18]  chr18        18-22      *
##  [19]  chr19        19-23      *
##  [20]  chr20        20-24      *
##  [21]  chr21        21-25      *
##  [22]  chr22        22-26      *
##  -----
## seqinfo: 22 sequences from an unspecified genome; no seqlengths
```

## 2.12 keepStandardChromosomes

This function internally uses the pre-defined tables inside GenomeInfoDb to find the correct seqlevels according to the sequence style of the object.

```
keepStandardChromosomes(gr, pruning.mode="coarse")

## GRanges object with 35 ranges and 0 metadata columns:
##      seqnames      ranges strand
##      <Rle> <IRanges>  <Rle>
##   [1]   chr1         1-5      *
##   [2]   chr2         2-6      *
##   [3]   chr3         3-7      *
##   [4]   chr4         4-8      *
##   [5]   chr5         5-9      *
##   ...     ...           ...    ...
##  [31]  chr31        31-35      *
##  [32]  chr32        32-36      *
##  [33]  chr33        33-37      *
##  [34]  chr34        34-38      *
##  [35]  chr35        35-39      *
##  -----
## seqinfo: 35 sequences from an unspecified genome; no seqlengths
```

One can also specify the optional species argument to be more precise.

```
plantgr <- GRanges(c(1:5,"MT","Pltd"), IRanges(1:7,width=5))
keepStandardChromosomes(plantgr, species="Arabidopsis thaliana",
```

```

                                pruning.mode="coarse")

## GRanges object with 7 ranges and 0 metadata columns:
##      seqnames      ranges strand
##      <Rle> <IRanges> <Rle>
## [1]      1      1-5      *
## [2]      2      2-6      *
## [3]      3      3-7      *
## [4]      4      4-8      *
## [5]      5      5-9      *
## [6]     MT     6-10      *
## [7]    Pltd     7-11      *
## -----
##      seqinfo: 7 sequences from an unspecified genome; no seqlengths

```

### 3 Seqinfo objects

```

## Note that all the arguments (except 'genome') must have the
## same length. 'genome' can be of length 1, whatever the lengths
## of the other arguments are.
x <- Seqinfo(seqnames=c("chr1", "chr2", "chr3", "chrM"),
             seqlengths=c(100, 200, NA, 15),
             isCircular=c(NA, FALSE, FALSE, TRUE),
             genome="toy")

length(x)
## [1] 4

seqnames(x)
## [1] "chr1" "chr2" "chr3" "chrM"

names(x)
## [1] "chr1" "chr2" "chr3" "chrM"

seqlevels(x)
## [1] "chr1" "chr2" "chr3" "chrM"

seqlengths(x)
## chr1 chr2 chr3 chrM
## 100 200 NA 15

isCircular(x)
## chr1 chr2 chr3 chrM
## NA FALSE FALSE TRUE

genome(x)
## chr1 chr2 chr3 chrM
## "toy" "toy" "toy" "toy"

x[c("chrY", "chr3", "chr1")] # subset by names

```



```
## Seqinfo object with 3 sequences from 2 genomes (NA, toy):
##   seqnames seqlengths isCircular genome
##   chrY      NA      NA    <NA>
##   chr3      NA     FALSE    toy
##   chr1     100      NA     toy

## Rename, drop, add and/or reorder the sequence levels:
xx <- x
seqlevels(xx) <- sub("chr", "ch", seqlevels(xx)) # rename
xx

## Seqinfo object with 4 sequences (1 circular) from toy genome:
##   seqnames seqlengths isCircular genome
##   ch1      100      NA     toy
##   ch2     200     FALSE    toy
##   ch3      NA     FALSE    toy
##   chM      15      TRUE    toy

seqlevels(xx) <- rev(seqlevels(xx)) # reorder
xx

## Seqinfo object with 4 sequences (1 circular) from toy genome:
##   seqnames seqlengths isCircular genome
##   chM      15      TRUE    toy
##   ch3      NA     FALSE    toy
##   ch2     200     FALSE    toy
##   ch1     100      NA     toy

seqlevels(xx) <- c("ch1", "ch2", "chY") # drop/add/reorder
xx

## Seqinfo object with 3 sequences from 2 genomes (toy, NA):
##   seqnames seqlengths isCircular genome
##   ch1      100      NA     toy
##   ch2     200     FALSE    toy
##   chY      NA      NA    <NA>

seqlevels(xx) <- c(chY="Y", ch1="1", "22") # rename/reorder/drop/add
xx

## Seqinfo object with 3 sequences from 2 genomes (NA, toy):
##   seqnames seqlengths isCircular genome
##   Y      NA      NA    <NA>
##   1     100      NA     toy
##   22     NA      NA    <NA>

y <- Seqinfo(seqnames=c("chr3", "chr4", "chrM"),
              seqlengths=c(300, NA, 15))
y

## Seqinfo object with 3 sequences from an unspecified genome:
##   seqnames seqlengths isCircular genome
##   chr3     300      NA    <NA>
##   chr4     NA      NA    <NA>
##   chrM     15      NA    <NA>
```

```

merge(x, y) # rows for chr3 and chrM are merged

## Warning in .Seqinfo.mergexy(x, y): Each of the 2 combined objects has sequence
levels not in the other:
## - in 'x': chr1, chr2
## - in 'y': chr4
## Make sure to always combine/compare objects based on the same reference
## genome (use suppressWarnings() to suppress this warning).

## Seqinfo object with 5 sequences (1 circular) from 2 genomes (toy, NA):
##   seqnames seqlengths isCircular genome
##   chr1      100      NA      toy
##   chr2      200     FALSE     toy
##   chr3      300     FALSE     toy
##   chrM       15      TRUE     toy
##   chr4       NA      NA     <NA>

suppressWarnings(merge(x, y))

## Seqinfo object with 5 sequences (1 circular) from 2 genomes (toy, NA):
##   seqnames seqlengths isCircular genome
##   chr1      100      NA      toy
##   chr2      200     FALSE     toy
##   chr3      300     FALSE     toy
##   chrM       15      TRUE     toy
##   chr4       NA      NA     <NA>

## Note that, strictly speaking, merging 2 Seqinfo objects is not
## a commutative operation, i.e., in general 'z1 <- merge(x, y)'
## is not identical to 'z2 <- merge(y, x)'. However 'z1' and 'z2'
## are guaranteed to contain the same information (i.e. the same
## rows, but typically not in the same order):
suppressWarnings(merge(y, x))

## Seqinfo object with 5 sequences (1 circular) from 2 genomes (toy, NA):
##   seqnames seqlengths isCircular genome
##   chr3      300     FALSE     toy
##   chr4       NA      NA     <NA>
##   chrM       15      TRUE     toy
##   chr1      100      NA      toy
##   chr2      200     FALSE     toy

## This contradicts what 'x' says about circularity of chr3 and chrM:
isCircular(y)[c("chr3", "chrM")] <- c(TRUE, FALSE)
y

## Seqinfo object with 3 sequences (1 circular) from an unspecified genome:
##   seqnames seqlengths isCircular genome
##   chr3      300      TRUE     <NA>
##   chr4       NA      NA     <NA>
##   chrM       15     FALSE     <NA>

if (interactive()) {
  merge(x, y) # raises an error
}

```

## 4 Examples

### 4.1 converting seqlevel styles (eg:UCSC to NCBI)

A quick example using *Drosophila Melanogaster*. The txdb object contains seqlevels in UCSC style, we want to convert them to NCBI

```
txdb <- TxDb.Dmelanogaster.UCSC.dm3.ensGene
seqlevels(txdb)

## [1] "chr2L"      "chr2R"      "chr3L"      "chr3R"      "chr4"      "chrX"
## [7] "chrU"       "chrM"       "chr2LHet"   "chr2RHet"   "chr3LHet"   "chr3RHet"
## [13] "chrXHet"    "chrYHet"    "chrUextra"
```

```
genomeStyles("Drosophila melanogaster")

##   circular  sex  auto  NCBI      UCSC      Ensembl
## 1   FALSE FALSE  TRUE   2L     chr2L      2L
## 2   FALSE FALSE  TRUE   2R     chr2R      2R
## 3   FALSE FALSE  TRUE   3L     chr3L      3L
## 4   FALSE FALSE  TRUE   3R     chr3R      3R
## 5   FALSE FALSE  TRUE    4     chr4       4
## 6   FALSE  TRUE FALSE    X     chrX       X
## 7   FALSE  TRUE FALSE    Y     chrY       Y
## 8    TRUE FALSE FALSE   MT     chrM dmel_mitochondrion_genome
## 9   FALSE FALSE FALSE  2LHet  chr2LHet    2LHet
##10   FALSE FALSE FALSE  2RHet  chr2RHet    2RHet
##11   FALSE FALSE FALSE  3LHet  chr3LHet    3LHet
##12   FALSE FALSE FALSE  3RHet  chr3RHet    3RHet
##13   FALSE FALSE FALSE  Xhet   chrXHet     XHet
##14   FALSE FALSE FALSE  Yhet   chrYHet     YHet
##15   FALSE FALSE FALSE   Un    chrU        U
##16   FALSE FALSE FALSE <NA> chrUextra  Uextra
```

```
mapSeqlevels(seqlevels(txdb), "NCBI")

##   chr2L      chr2R      chr3L      chr3R      chr4      chrX      chrU
##   "2L"      "2R"      "3L"      "3R"      "4"      "X"      "Un"
##   chrM chr2LHet chr2RHet chr3LHet chr3RHet chrXHet chrYHet
##   "MT"  "2LHet"  "2RHet" "3LHet" "3RHet" "Xhet"  "Yhet"
## chrUextra
##      NA
```

### 4.2 converting styles and removing unwanted seqlevels

Suppose we read in a Bam file or a BED file and the resulting GRanges have a lot of seqlevels which are not required by your analysis or you want to rename the seqlevels from the current style to your own style (eg:UCSC to NCBI), we can use the functionality provided by *GenomeInfoDb* to do that.

Let us say that we have extracted the seqlevels of the Seqinfo object(say GRanges from a BED file) in a variable called "sequence".

```
sequence <- seqlevels(x)

## sequence is in UCSC format and we want NCBI style
newStyle <- mapSeqlevels(sequence, "NCBI")
newStyle <- newStyle[complete.cases(newStyle)] # removing NA cases.

## rename the seqlevels
x <- renameSeqlevels(x, newStyle)

## keep only the seqlevels you want (say autosomes)
auto <- extractSeqlevelsByGroup(species="Homo sapiens", style="NCBI",
                                group="auto")
x <- keepSeqlevels(x, auto)
```

## 5 Session Information

Here is the output of `sessionInfo` on the system on which this document was compiled:

```
toLatex(sessionInfo())
```

- R version 4.0.4 (2021-02-15), x86\_64-pc-linux-gnu
- Locale: LC\_CTYPE=en\_US.UTF-8, LC\_NUMERIC=C, LC\_TIME=en\_US.UTF-8, LC\_COLLATE=C, LC\_MONETARY=en\_US.UTF-8, LC\_MESSAGES=en\_US.UTF-8, LC\_PAPER=en\_US.UTF-8, LC\_NAME=C, LC\_ADDRESS=C, LC\_TELEPHONE=C, LC\_MEASUREMENT=en\_US.UTF-8, LC\_IDENTIFICATION=C
- Running under: Ubuntu 18.04.5 LTS
- Matrix products: default
- BLAS: /home/biocbuild/bbs-3.12-bioc/R/lib/libRblas.so
- LAPACK: /home/biocbuild/bbs-3.12-bioc/R/lib/libRlapack.so
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, stats4, utils
- Other packages: AnnotationDbi 1.52.0, Biobase 2.50.0, BiocGenerics 0.36.0, GenomeInfoDb 1.26.5, GenomicFeatures 1.42.3, GenomicRanges 1.42.0, IRanges 2.24.1, S4Vectors 0.28.1, TxDb.Dmelanogaster.UCSC.dm3.ensGene 3.2.2
- Loaded via a namespace (and not attached): BiocFileCache 1.14.0, BiocManager 1.30.12, BiocParallel 1.24.1, BiocStyle 2.18.1, Biostrings 2.58.0, DBI 1.1.1, DelayedArray 0.16.3, GenomeInfoDbData 1.2.4, GenomicAlignments 1.26.0, Matrix 1.3-2, MatrixGenerics 1.2.1, R6 2.5.0, RCurl 1.98-1.3, RSQLite 2.2.5, Rcpp 1.0.6, Rsamtools 2.6.0, SummarizedExperiment 1.20.0, XML 3.99-0.6, XVector 0.30.0, askpass 1.1, assertthat 0.2.1, biomaRt 2.46.3, bit 4.0.4, bit64 4.0.5, bitops 1.0-6, blob 1.2.1, cachem 1.0.4, compiler 4.0.4, crayon 1.4.1, curl 4.3, dbplyr 2.1.0, debugme 1.1.0, digest 0.6.27, dplyr 1.0.5, ellipsis 0.3.1, evaluate 0.14, fansi 0.4.2, fastmap 1.1.0, generics 0.1.0, glue 1.4.2, grid 4.0.4, highr 0.8, hms 1.0.0, htmltools 0.5.1.1, httr 1.4.2, knitr 1.31, lattice 0.20-41, lifecycle 1.0.0, magrittr 2.0.1,

matrixStats 0.58.0, memoise 2.0.0, openssl 1.4.3, pillar 1.5.1, pkgconfig 2.0.3,  
prettyunits 1.1.1, progress 1.2.2, purrr 0.3.4, rappdirs 0.3.3, rlang 0.4.10,  
rmarkdown 2.7, rtracklayer 1.50.0, stringi 1.5.3, stringr 1.4.0, tibble 3.1.0,  
tidyselect 1.1.0, tools 4.0.4, utf8 1.2.1, vctrs 0.3.7, xfun 0.22, xml2 1.3.2, yaml 2.2.1,  
zlibbioc 1.36.0