An Introduction to *GenomeInfoDb*

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


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 .................................................... 4
  2.9 renameSeqlevels ............................................... 5
  2.10 dropSeqlevels .................................................. 6
  2.11 keepSeqlevels .................................................. 6
  2.12 keepStandardChromosomes .................................... 7

3 Seqinfo objects .......................................................... 8

4 Examples ...................................................................... 10
  4.1 converting seqlevel styles (eg:UCSC to NCBI) .................... 10
  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.

```r
glimpse(genomeStyles())
```

Oragnism's supported by GenomeInfoDb can be found by:

```r
names(genomeStyles())
```

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.

```r
head(genomeStyles("Homo_sapiens"),5)
```
An Introduction to *GenomInfoDb*

We can also check if a given style is supported by GenomInfoDb 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`

```
exttractSeqlevels(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")
```
An Introduction to *GenomeInfoDb*

```r
## [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:

```r
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 seq-
names. In the following example, we show how you can find the order for a given seqnames
character vector.

```r
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 seq-
names. In the following example, we show how you can find the rank for a given seqnames
character vector.

```r
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.
An Introduction to GenomInfoDb

```r
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.

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

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.

```r
newnames <- paste0("chr",1:35)
head(newnames)
##  [1] "chr1"  "chr2"  "chr3"  "chr4"  "chr5"  "chr6"
```

```r
gr <- renameSeqlevels(gr,newnames)
```

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

```r
newnames <- paste0("chr",1:35)
head(newnames)
##  [1] "chr1"  "chr2"  "chr3"  "chr4"  "chr5"  "chr6"
```

```r
gr <- renameSeqlevels(gr,newnames)
```
An Introduction to *GenomeInfoDb*

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"`.  

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

```
## GRanges object with 22 ranges and 0 metadata columns:
## seqlines 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.  

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

```
## GRanges object with 22 ranges and 0 metadata columns:
## seqlines ranges strand
## <Rle> <IRanges> <Rle>
## [1] chr1 1-5 *
## [2] chr2 2-6 *
## -------
## seqinfo: 35 sequences from an unspecified genome; no seqlengths
```
An Introduction to GenomeInfoDb

## seqinfo: 22 sequences from an unspecified genome; no seqlengths

### keepStandardChromosomes

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

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

### 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")
```

### seqinfo: 35 sequences from an unspecified genome; no seqlengths
3 Seqinfo objects

```r
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

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
```

---

An Introduction to **GenomeInfoDb**

## seqinfo: 7 sequences from an unspecified genome; no seqlengths

## seqinfo: 7 sequences from an unspecified genome; no seqlengths

---
An Introduction to *GenomeInfoDb*

```r
## 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 .merge_two_Seqinfo_objects(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
```
An Introduction to GenomInfoDb

## 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")]

## 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)
An Introduction to **GenomeInfoDb**

```r
## [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  TRUE  2L      chr2L   2L
##  2   FALSE  TRUE  2R      chr2R   2R
##  3   FALSE  TRUE  3L      chr3L   3L
##  4   FALSE  TRUE  3R      chr3R   3R
##  5   FALSE  TRUE   4       chr4    4
##  6   FALSE  TRUE   X       chrX    X
##  7   FALSE  TRUE FALSE  chrY    Y
##  8   TRUE   FALSE FALSE  chrM   dmel_mitochondrion_genome
##  9   FALSE  FALSE  2LHet   chr2LHet 2LHet
## 10   FALSE  FALSE  2RHet   chr2RHet 2RHet
## 11   FALSE  FALSE  3LHet   chr3LHet 3LHet
## 12   FALSE  FALSE  3RHet   chr3RHet 3RHet
## 13   FALSE  FALSE  XHet    chrXHet XHet
## 14   FALSE  FALSE  YHet    chrYHet YHet
## 15   FALSE  FALSE   Un     chrU    U
## 16   FALSE  FALSE FALSE  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:USCS 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".

```r
sequence <- seqlevels(x)
```

```r
## 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)
```
An Introduction to GenomeInfoDb

```r
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:

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

- R version 4.3.1 (2023-06-16), x86_64-pc-linux-gnu
- Locale: LC_CTYPE=en_US.UTF-8, LC_NUMERIC=C, LC_TIME=en_GB, LC_COLLATE=C, LC_MONETARY=en_US.UTF-8, LC_MESSAGES=en_US.UTF-8
- Time zone: America/New_York
- TZcode source: system (glibc)
- Running under: Ubuntu 22.04.3 LTS
- Matrix products: default
- BLAS: /home/biocbuild/bbs-3.18-bioc/R/lib/libRblas.so
- LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.10.0
- Base packages: base, datasets, grDevices, graphics, methods, stats, stats4, utils
- Other packages: AnnotationDbi 1.63.2, Biobase 2.61.0, BiocGenerics 0.47.0, BiocStyle 2.29.1, GenomeInfoDb 1.37.4, GenomicFeatures 1.53.2, GenomicRanges 1.53.1, IRanges 2.35.2, S4Vectors 0.39.1, TxDb.Dmelanogaster.UCSC.dm3.ensGene 3.2.2
- Loaded via a namespace (and not attached): BiocFileCache 2.9.1, BiocIO 1.11.0, BiocManager 1.30.22, BiocParallel 1.35.4, Biostrings 2.69.2, DBI 1.1.3, DelayedArray 0.27.10, GenomeInfoDbData 1.2.10, GenomicAlignments 1.37.0, KEGGREST 1.41.0, Matrix 1.6-1, MatrixGenerics 1.13.1, R6 2.5.1, RCurl 1.98-1.12, RSQLite 2.3.1, Rsamtools 2.17.0, S4Arrays 1.1.6, SparseArray 1.1.2, SummarizedExperiment 1.31.1, XML 3.99-0.14, XVector 0.41.1, abind 1.4-5, biomaRt 2.57.1, bit 4.0.4, bit64 4.0.5, bitspec 1.0.7, blob 1.2.4, bookdown 0.35, bslib 0.5.1, cachem 1.0.8, cli 3.6.1, codeTools 0.2-19, compiler 4.3.1, crayon 1.5.2, curl 5.0.2, dbplyr 2.3.3, digest 0.6.33, dplyr 1.1.3, evaluate 0.21, fansi 1.0.4, fastmap 1.1.1, filelock 1.0.2, generics 0.1.3, glue 0.6.2, grid 4.3.1, highr 0.10, hms 1.1.3, htmltools 0.5.6, httr 1.4.7, jquerylib 0.1.4, jsonlite 1.8.7, knitr 1.43, lattice 0.21-8, lifecycle 1.0.3, magrittr 2.0.3, matrixStats 1.0.0, memoise 2.0.1, parallel 4.3.1, pillar 1.9.0, pkgconfig 2.0.3, png 0.1-8, prettyunits 1.1.1, progress 1.2.2, rappdirs 0.3.3, restfulr 0.0.15, rjson 0.2.21, rlang 1.1.1, rmardown 2.24, rtracklayer 1.61.1, sass 0.4.7, stringi 1.7.12, stringr 1.5.0, tibble 3.2.1, tidyselect 1.2.0, tools 4.3.1, utf8 1.2.3, vctrs 0.6.3, xfun 0.40, xml2 1.3.5, yam1 2.3.7, zlibbioc 1.47.0