gUtils Documentation

Release 1.0.0

Marcin Imielinski and Jeremiah Wala

August 04, 2015

Contents

Contents:

CHAPTER 1

Introduction

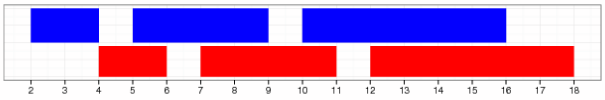
CHAPTER 2

Range Operations

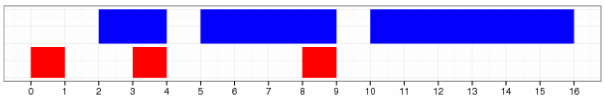
This section will describe additional GRanges operations provided by gUtils.

```
## make some example data sets
ref19 <- readRDS(system.file("extdata","refGene.hg19.gr.rds", package="gUtils"))
gr  <- GRanges(1, IRanges(c(2,5,10), c(4,9,16)), seqinfo=Seqinfo("1", 20))
gr2 <- c(gr, GRanges(1, IRanges(c(1,9), c(6,14)), seqinfo=Seqinfo("1", 20)))
dt <- data.table(seqnames=1, start=c(2,5,10), end=c(3,8,15))
```

```
shift(gr, 2)
```




```
flank(gr, width=2)
```




```
gr.start(gr, width=3)
```

```
gr.end(gr, width=3)
```

```
gr.mid(gr) + 2
```

```
grbind
```

```
## add metadata to one field
mcols(gr)$score = 3
## try to concatenate
c(gr,gr2)  ## ERROR
## with grbind
grbind(gr, gr2) ## SUCCESS. Adds NA for missing fields
## GenomicRanges::c does this already for GRangesList
```

```
streduce(gr2)
```

```
gr.sample(gr2, 2, len=2, replace=TRUE)
```

```
## output GRanges
GRanges object with 3 ranges and 1 metadata column:
   seqnames    ranges strand |  query.id
      <Rle> <IRanges>  <Rle> | <integer>
[1]        1   [ 8,  9]      * |         2
[2]        1   [ 5,  6]      * |         2
[3]        1   [11, 12]      * |         3
```

```
gr.rand(w=c(2,5,3), seqinfo(gr))
```

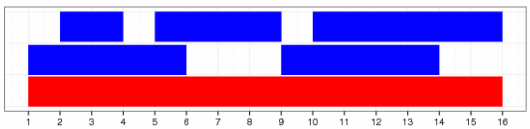
```
gr.simplify
```

```
gr.tile(GRanges(1, IRanges(1,9)), w=3) + 1
```

```
gr.refactor
```

```
gr.tile.map
```

```
gr.round
```



CHAPTER 3

Data manipulation

R provides a number of data structures for storing genomic data, each with its advantages and drawbacks.

The most useful structures for this purpose are:

GRanges Store ranges along with metadata, sequences and the coordaintes of the reference genome.

GRangesList Store groups of ranges, with additional metadata belonging to the group.

data.table Fast and efficient general-purpose container similar to data.frame, but with significant performance improvements.

In gUtils functions, we often manipulate the data to move between these data structures where one is more useful than another. A key example is in `gr.findoverlaps`, which converts the input `GRanges` into `data.table` objects to take advantage of the blazing fast `foverlaps` util. For the most part, these conversions should be invisible to the user.

However, often there are data structures conversions that may be useful to the end user. This includes unlisting GRangesList objects into GRanges, making data.table objects from GRanges, and binding together multiple GRanges or GRangesList objects, among others. This section will describe and demonstrate the functionality `gUtils` provides for manipulating these data structures.

```
ref19 <- readRDS(system.file("extdata","refGene.hg19.gr.rds", package="gUtils"))
gr <- GRanges(1, IRanges(c(2,5,10), c(4,9,16)), seqinfo=Seqinfo("1", 20))
dt <- data.table(seqnames=1, start=c(2,5,10), end=c(3,8,15))
```

grbind

grlbind

dtgr

grdt

si2gr

gr2gatk

gr.flatten

gr.flatmap

grl.split

grl.stripnames

grl.unlist

grl.span

grl.pivot

rrbind

CHAPTER 4

Additional utilities

gr.string and grl.string

parse.grl and parse.gr

gr.gatk

gr.chr and gr.nochr

gr.fix

gr.tostring

affine.map

gr.findoverlaps

gr.match

gr.in

gr.duplicated

gr.val

gr.dist

alpha

grl.in

chunk

import.ucsc

CHAPTER 5

BAM/SAM operations

This section will describe utility functions for manipipulating BAM files, using functions based on the Rsamtools package.

read.bam

bam.cov.gr

bam.cov.tile.st

bam.pair.cov

counts2rpkm

get.pairs.grl

count.clips

varbase

varcount

mafcount

splice.cigar

gc_content

bamtag

CHAPTER 6

Indices and tables

- genindex
- modindex
- search