Package 'HiContacts'

March 10, 2023

Title HiContacts: R interface to cool files

Version 1.1.0

Date 2022-08-16

Description HiContacts: R interface to (m)cool files and other Hi-C processed file formats. HiContacts provides a collection of tools to analyse and visualize Hi-C datasets. It can import data from pairs or (m)cool files.

License MIT + file LICENSE

URL https://github.com/js2264/HiContacts

BugReports https://github.com/js2264/HiContacts/issues

Depends R (>= 4.2)

Imports HiContactsData, InteractionSet, GenomicInteractions, GenomicRanges, IRanges, GenomeInfoDb, S4Vectors, BiocGenerics, methods, rhdf5, Matrix, vroom, tibble, tidyr, dplyr, glue, stringr, reticulate, ggplot2, ggrastr, scales

Suggests cowplot, testthat (>= 3.0.0), BiocStyle, knitr, rmarkdown

biocViews HiC, DNA3DStructure, DataImport

Encoding UTF-8

VignetteBuilder knitr

LazyData false

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.1

git_url https://git.bioconductor.org/packages/HiContacts

git_branch master

git_last_commit 042f21c

git_last_commit_date 2022-11-01

Date/Publication 2023-03-10

Author Jacques Serizay [aut, cre] (<https://orcid.org/0000-0002-4295-0624>)

Maintainer Jacques Serizay <jacquesserizay@gmail.com>

R topics documented:

`bwrColors` *Matrix palettes*

Description

Matrix palettes

Usage

```
bwrColors()

afmhotrColors()

bbrColors()
```

Value

ggplot

Examples

```
bwrColors()
afmhotrColors()
bbrColors()
```

centros_yeast *Example datasets provided in* HiContacts *&* HiContactsData

Description

Example datasets provided in HiContacts & HiContactsData

Usage

```
data(centros_yeast)

contacts_yeast()

contacts_yeast_eco1()

full_contacts_yeast()
```

Format

An object of class "GRanges".

An object of class "Contacts".

Source

HiContacts

Examples

```
data(centros_yeast)
centros_yeast
contacts_yeast()
contacts_yeast_eco1()
full_contacts_yeast()
```

check_cool_file *Checks functions*

Description

Useful functions to validate the nature/structure of (m)cool files or Contacts objects.

Usage

```
check_cool_file(path)

check_resolution(contacts, resolution)

check_cool_format(path, resolution)

is_mcool(path)

is_cool(path)

is_same_seqinfo(...)

is_same_resolution(...)

is_same_bins(...)

is_same_regions(...)

is_comparable(...)

is_square(pair)

are_contacts(...)

is_symmetrical(contacts)
```

Arguments

path	Path of a (m)cool file
contacts	A Contacts object
resolution	Resolution
...	Contacts object
pair	Pairs object with length of 1

Value

Logical

Examples

```
library(HiContacts)
contacts_yeast <- contacts_yeast()
is_symmetrical(contacts_yeast)
```

cisTransRatio *cisTransRatio*

Description

Quickly computes a cis-trans ratio of interactions.

Usage

```
cisTransRatio(x)
```

Arguments

x	A Contacts object over the full genome

Value

a tibble, listing for each chr. the % of cis/trans interactions

Examples

```
library(HiContacts)
full_contacts_yeast <- full_contacts_yeast()
cisTransRatio(full_contacts_yeast)
```

Contacts-class Contacts *S4 class and methods*

Description

Contacts S4 class and methods

Usage

```
Contacts(
  file,
  resolution = NULL,
  focus = NULL,
  metadata = list(),
  topologicalFeatures = S4Vectors::SimpleList(loops =
   S4Vectors::Pairs(GenomicRanges::GRanges(), GenomicRanges::GRanges()), borders =
   GenomicRanges::GRanges(), compartments = GenomicRanges::GRanges(), viewpoints =
     GenomicRanges::GRanges()),
  pairsFile = NULL
)
```

```
## S4 method for signature 'Contacts'
fileName(object)

## S4 method for signature 'Contacts'
resolutions(x)

## S4 method for signature 'Contacts'
resolution(x)

## S4 method for signature 'Contacts'
focus(x)

## S4 replacement method for signature 'Contacts,character'
focus(x) <- value

## S4 method for signature 'Contacts'
interactions(x)

## S4 replacement method for signature 'Contacts,GInteractions'
interactions(x) <- value

## S4 method for signature 'Contacts,missing'
scores(x)

## S4 replacement method for signature 'Contacts,character,numeric'
scores(x, name) <- value

## S4 method for signature 'Contacts,missing'
topologicalFeatures(x)

## S4 replacement method for signature 'Contacts,character,GRangesOrGInteractions'
topologicalFeatures(x, name) <- value

## S4 method for signature 'Contacts'
pairsFile(x)

## S4 replacement method for signature 'Contacts,character'
pairsFile(x) <- value

## S4 replacement method for signature 'Contacts,list'
metadata(x) <- value

## S4 method for signature 'Contacts'
length(x)

## S4 method for signature 'Contacts,numeric,ANY,ANY'
x[i]
```

```
## S4 method for signature 'Contacts'
seqinfo(x)

## S4 method for signature 'Contacts'
bins(x)

## S4 method for signature 'Contacts'
anchors(x)

## S4 method for signature 'Contacts'
regions(x)

## S4 method for signature 'Contacts'
summary(object)

## S4 method for signature 'Contacts'
show(object)
```

Arguments

file	Path to a (m)cool file
resolution	Resolution to use with mcool file
focus	focus Chr. coordinates for which interaction counts are extracted from the .(m)cool file, provided as a character string (e.g. "II:4000-5000"). If not provided, the entire (m)cool file will be imported.
metadata	list of metadata
topologicalFeatures	topologicalFeatures provided as a named SimpleList
pairsFile	Path to an associated .pairs file
object	A Contacts object.
x	A Contacts object.
value	value
name	name
i	a range or boolean vector.

Value

a new Contacts object.

Slots

fileName Path of (m)cool file

focus Chr. coordinates for which interaction counts are extracted from the .(m)cool file.

resolutions Resolutions available in the .(m)cool file.

resolution Current resolution

`interactions` Genomic Interactions extracted from the .(m)cool object

`scores` Available interaction scores.

`topologicalFeatures` Topological features associated with the dataset (e.g. loops (\<Pairs\>), borders (\<GRanges\>), viewpoints (\<GRanges\>), etc...)

`pairsFile` Path to the .pairs file associated with the .(m)cool file

`metadata` metadata associated with the .(m)cool file.

Examples

```
library(HiContacts)
contacts_yeast <- contacts_yeast()
contacts_yeast
fileName(contacts_yeast)
resolutions(contacts_yeast)
resolution(contacts_yeast)
focus(contacts_yeast)
interactions(contacts_yeast)
scores(contacts_yeast)
tail(scores(contacts_yeast, 1))
tail(scores(contacts_yeast, 'balanced'))
scores(contacts_yeast, 'test') <- runif(length(contacts_yeast))
tail(scores(contacts_yeast, 'test'))
full_contacts_yeast <- full_contacts_yeast()
topologicalFeatures(full_contacts_yeast)
topologicalFeatures(full_contacts_yeast, 1)
topologicalFeatures(full_contacts_yeast, 'centromeres')
data(centros_yeast)
topologicalFeatures(contacts_yeast, 'centromeres') <- centros_yeast
topologicalFeatures(contacts_yeast, 'centromeres')
pairsFile(full_contacts_yeast)
length(contacts_yeast)
contacts_yeast[seq_len(10)]
seqinfo(contacts_yeast)
bins(contacts_yeast)
anchors(contacts_yeast)
regions(contacts_yeast)
summary(contacts_yeast)
show(contacts_yeast)
as(contacts_yeast, 'GInteractions')
as(contacts_yeast, 'ContactMatrix')
as(contacts_yeast, 'matrix')[seq_len(10), seq_len(10)]
as(contacts_yeast, 'data.frame')
```

`detrend` *Arithmetics with (m)cool file(s)*

Description

Different operations can be performed:

- Detrending a contact matrix, i.e. removing the distance-dependent contact trend;
- Autocorrelate a contact matrix: this is is typically done to highlight large-scale compartments;
- Divide one contact matrix by another;
- Merge multiple contact matrices;
- Serpentinify, or smooth a contact matrix out. This requires `serpentine` python package to be installed.

Usage

```
detrend(x, use.scores = "balanced")

autocorrelate(x, use.scores = "balanced", ignore_ndiags = 3)

divide(x, by, use.scores = "balanced")

merge(..., use.scores = "balanced")

serpentinify(
  x,
  use.scores = "balanced",
  use_serpentine_trend = TRUE,
  serpentine_niter = 10L,
  serpentine_ncores = 16L
)
```

Arguments

`x`	a `Contacts` object
`use.scores`	use.scores
`ignore_ndiags`	ignore N diagonals when calculating correlations
`by`	a `Contacts` object
`...`	`Contacts` objects
`use_serpentine_trend`	whether to use the trend estimated with serpentine (this requires `reticulate` and the python package `serpentine`)
`serpentine_niter`	number of iterations to use for serpentine
`serpentine_ncores`	number of CPUs to use for serpentine

Value

a Contacts object with two additional scoress: expected and detrended

a Contacts object with a single autocorrelation scores

a Contacts object with a single ratio scores

a Contacts object. Each returned scores is the sum of the corresponding scores from input Contacts.

a Contacts object with a single smoothen scores

Examples

```
#### -----
#### Detrending a contact matrix
#### -----

library(HiContacts)
contacts_yeast <- contacts_yeast()
contacts_yeast <- detrend(contacts_yeast)
scores(contacts_yeast)
#### -----
#### Auto-correlate a contact matrix
#### -----

contacts_yeast <- autocorrelate(contacts_yeast)
scores(contacts_yeast)
plotMatrix(contacts_yeast, scale = 'linear', limits = c(-1, 1), cmap = bwrColors())
#### -----
#### Divide 2 contact matrices
#### -----

contacts_yeast <- contacts_yeast()
contacts_yeast_eco1 <- contacts_yeast_eco1()
div_contacts <- divide(contacts_yeast_eco1, by = contacts_yeast)
div_contacts
plotMatrix(div_contacts, scale = 'log2', limits = c(-2, 2), cmap = bwrColors())
#### -----
#### Merge 2 contact matrices
#### -----

merged_contacts <- merge(contacts_yeast_eco1, contacts_yeast)
merged_contacts
```

---

distanceLaw *Compute the law of distance-dependent contact frequency, a.k.a. P(s)*

Description

P(s) will be approximated if no pairs are provided, or the exact P(s) will be computed if a .pairs file is added to the Contacts object using pairsFile(x) <- "...".

Usage

```
distanceLaw(
  x,
  by_chr = FALSE,
  filtered_chr = c("XII", "chrXII", "chr12", "12", "Mito", "MT", "chrM")
)

localDistanceLaw(x, coords = coords)

PsBreaks()
```

Arguments

x	A Contacts object
by_chr	by_chr
filtered_chr	filtered_chr
coords	GRanges

Value

a tibble

a tibble

tbl

Examples

```
library(HiContacts)
contacts_yeast <- contacts_yeast()
ps <- distanceLaw(contacts_yeast)
ps
local_ps <- localDistanceLaw(
    contacts_yeast,
    GenomicRanges::GRanges(
        c("telomere" = "II:1-20000", "arm" = "II:300000-700000")
    )
)
local_ps
```

---

getAnchors *Parsing (m)cool files*

Description

These functions are the workhorse internal functions used to import a .(m)cool file as GenomicInteractions (wrapped into a Contacts object by Contacts() function).

Usage

```
getAnchors(file, resolution = NULL, balanced = "cooler")

getCountsFromPair(file, pair, anchors, resolution = NULL)

getCounts(file, coords, anchors, resolution = NULL)

fetchCool(file, path, resolution = NULL, idx = NULL, ...)

lsCoolFiles(file, verbose = FALSE)

lsCoolResolutions(file, verbose = FALSE)

peekCool(file, path, resolution = NULL, n = 10)

cool2seqinfo(file, resolution = NULL)

cool2gi(file, coords = NULL, resolution = NULL)

gi2cm(gi)

cm2matrix(cm, replace_NA = NA)

pairs2gi(
  file,
  chr1.field = 2,
  start1.field = 3,
  chr2.field = 4,
  start2.field = 5,
  nThread = 16,
  nrows = Inf
)
```

Arguments

file	pairs file: <readname>\t<chr1>\t<start1>\t<chr2>\t<start2>
resolution	resolution
balanced	import balancing scores
pair	pair (e.g. S4Vectors::Pairs(GRanges("II:200000-300000"), GRanges("II:70000-100000"))).
anchors	anchors
coords	NULL, character, or GRanges. Can also be a Pairs object of paired GRanges (length of 1).
path	path
idx	idx to extract in cool file
...	...

verbose	Print resolutions in the console
n	n
gi	A GInteractions object
cm	A ContactMatrix object
replace_NA	Replace NA values
chr1.field	chr1.field
start1.field	start1.field
chr2.field	chr2.field
start2.field	start2.field
nThread	Number of CPUs to use to import the pairs file in R
nrows	Number of pairs to import

Value

anchors from (m)cool, stored as a GRanges

counts from (m)cool, stored as a tibble

counts from (m)cool, stored as a tibble

vector

vector

vector

vector

a Seqinfo object

a GenomicInteractions object

a ContactMatrix object

a dense matrix

a GenomicInteractions object

---

ggthemeHiContacts *ggplot2-related functions*

Description

ggplot2-related functions

Usage

```
ggthemeHiContacts(ticks = TRUE)
```

Arguments

ticks	ticks

Value

a custom ggplot2 theme

`HiContacts package` *HiContacts package*

Description

HiContacts: R interface to (m)cool files and other Hi-C processed file formats. HiContacts provides a collection of tools to analyse and visualize Hi-C datasets. It can import data from pairs or (m)cool files.

`plot4C` *Plotting virtual 4C profiles*

Description

Plotting virtual 4C profiles

Usage

```
plot4C(x, mapping)
```

Arguments

`x`	GRanges, generally the output of `virtual4C()`
`mapping`	aes to pass on to ggplot2

Value

ggplot

Examples

```
library(HiContacts)
contacts_yeast <- contacts_yeast()
v4C <- virtual4C(contacts_yeast, GenomicRanges::GRanges('II:490000-510000'))
plot4C(v4C, ggplot2::aes(x = center, y = score))
```

plotMatrix *Plotting a contact matrix*

Description

Plotting a contact matrix

Usage

```
plotMatrix(
  x,
  use.scores = NULL,
  scale = "log10",
  loops = NULL,
  borders = NULL,
  limits = NULL,
  dpi = 500,
  rasterize = TRUE,
  symmetrical = TRUE,
  chrom_lines = TRUE,
  cmap = NULL
)

ggMatrix(mat, ticks = TRUE, cols = afmhotrColors(), limits)
```

Arguments

x	x
use.scores	use.scores
scale	scale
loops	loops
borders	borders
limits	limits
dpi	dpi
rasterize	rasterize
symmetrical	symmetrical
chrom_lines	chrom_lines
cmap	color map
mat	mat
ticks	ticks
cols	cols

Value

ggplot

ggplot

Examples

```
library(HiContacts)
contacts_yeast <- contacts_yeast()
plotMatrix(
    contacts_yeast,
    use.scores = 'balanced',
    scale = 'log10',
    limits = c(-4, -1)
)
```

`plotPs` *Plotting a P(s) distance law*

Description

Plotting a P(s) distance law

Usage

```
plotPs(..., xlim = c(5000, 499000), ylim = c(1e-08, 1e-04))

plotPsSlope(..., xlim = c(5000, 499000), ylim = c(-3, 0))
```

Arguments

`...`	...
`xlim`	xlim
`ylim`	ylim

Value

ggplot

ggplot

Examples

```
## Single P(s)

contacts_yeast <- contacts_yeast()
ps <- distanceLaw(contacts_yeast)
plotPs(ps, ggplot2::aes(x = binned_distance, y = norm_p))
```

```
## Comparing several P(s)

contacts_yeast <- contacts_yeast()
contacts_yeast_eco1 <- contacts_yeast_eco1()
ps_wt <- distanceLaw(contacts_yeast)
ps_wt$sample <- 'WT'
ps_eco1 <- distanceLaw(contacts_yeast_eco1)
ps_eco1$sample <- 'eco1'
ps <- rbind(ps_wt, ps_eco1)
plotPs(ps, ggplot2::aes(x = binned_distance, y = norm_p, group = sample, color = sample))
plotPsSlope(ps, ggplot2::aes(x = binned_distance, y = slope))
```

`splitCoords` *Utils functions*

Description

Utilities to facilitate parsing/handling of coordinates, GInteractions, Pairs, ...

Usage

```
splitCoords(coords)

coords2char(coords, big.mark = ",")

char2coords(char)

getHicStats(hicstuff_log)

fullContactInteractions(chr, start, end, binning)

sdiag(A, k = 0) <- value

sortPairs(pairs)

asGInteractions(df)
```

Arguments

`coords`	coords
`big.mark`	big.mark
`char`	char (e.g. "II:30000-50000" or "II:30000-50000 x II:60000-80000")
`hicstuff_log`	log file generated by hicstuff
`chr`	chr
`start`	start
`end`	end

binning	binning
A	A
k	k
value	value
pairs	pairs
df	df

Value

a list containing chr, start and end

a character string

a S4Vectors::Pairs object

a list

a GenomicInteractions object

a matrix

a Pairs object

a GenomicInteractions object

---

virtual4C *Computing virtual 4C profiles*

Description

From a (m)cool pre-imported in memory, computes a 4C profile using a user-specified viewpoint.

Usage

```
virtual4C(x, viewpoint, use.scores = "balanced")
```

Arguments

x	a Contacts object
viewpoint	viewpoint, defined as a GRanges
use.scores	use.scores

Value

A tibble with the contact frequency of the viewpoint, per bin along the imported genomic range.

Examples

```
library(HiContacts)
contacts_yeast <- contacts_yeast()
v4C <- virtual4C(contacts_yeast, GenomicRanges::GRanges('II:490000-510000'))
v4C
```

Index