Package 'HicAggR'

May 24, 2024

Type Package

Title Set of 3D genomic interaction analysis tools

Version 1.1.2

Description This package provides a set of functions useful in the analysis of 3D genomic interactions. It includes the import of standard HiC data formats into R and HiC normalisation procedures. The main objective of this package is to improve the visualization and quantification of the analysis of HiC contacts through aggregation. The package allows to import 1D genomics data, such as peaks from ATACSeq, ChIPSeq, to create potential couples between features of interest under user-defined parameters such as distance between pairs of features of interest. It allows then the extraction of contact values from the HiC data for these couples and to perform Aggregated Peak Analysis (APA) for visualization, but also to compare normalized contact values between conditions. Overall the package allows to integrate 1D genomics data with 3D genomics data, providing an easy access to HiC contact values.

biocViews Software, HiC, DataImport, DataRepresentation, Normalization, Visualization, DNA3DStructure, ATACSeq, ChIPSeq, DNaseSeq, RNASeq

License MIT + file LICENSE

URL https://bioconductor.org/packages/HicAggR, https://cuvierlab.github.io/HicAggR/, https://github.com/CuvierLab/HicAggR

BugReports https://github.com/CuvierLab/HicAggR/issues

Depends R (>= 4.2.0)

Imports InteractionSet, BiocGenerics, BiocParallel, dplyr, GenomeInfoDb, GenomicRanges, ggplot2, grDevices, IRanges, Matrix, methods, rhdf5, rlang, rtracklayer, S4Vectors, stats, utils, strawr, tibble, stringr, tidyr, gridExtra, data.table, reshape, checkmate, purrr, withr

Suggests covr, tools, kableExtra (>= 1.3.4), knitr (>= 1.45), rmarkdown, testthat (>= 3.0.0), BiocFileCache (>= 2.6.1)

VignetteBuilder knitr

Encoding UTF-8

LazyData false

RoxygenNote 7.2.3

Config/testthat/edition 3

Roxygen list(markdown = TRUE)

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

git_branch devel

git_last_commit 527fc80

git_last_commit_date 2024-05-10

Repository Bioconductor 3.20

Date/Publication 2024-05-24

Author Robel Tesfaye [aut, ctb] (<https://orcid.org/0000-0003-2358-219X>), David Depierre [aut], Naomi Schickele [ctb], Nicolas Chanard [aut], Refka Askri [ctb], Stéphane Schaak [aut, ctb], Pascal Martin [ctb], Olivier Cuvier [cre, ctb] (<https://orcid.org/0000-0003-0644-2734>)

Maintainer Olivier Cuvier <olivier.cuvier@univ-tlse3.fr>

Contents

`HicAggR-package` *HicAggR*

Description

HicAggR is a package that allows to integrate 1D genomics data with 3D genomics data. This package provides a set of functions useful in the analysis of 3D genomic interactions. It includes the import of standard HiC data formats into R and HiC normalisation procedures. The main objective of this package is to facilitate the visualization and quantification of the analysis of HiC contacts through aggregation. The package also provides options to import externally normalized HiC data to perform an in-depth analysis by investigating genome wide interactions between features of interest. The package can use 1D genomics data (such as annotation data or peaks of features of interest in a GRanges object) to form potential couples between features of interest under user specified constraints (TADs coordinates or fixed minimum/max distance). Using these formed couples it can extract HiC contact data for these specific couples. The submatrices extracted as such can then be aggregated to summarize genome-wide interactions, to perform per submatrix operation or compare between conditions. It also allows to identify couples with significantly signal-enriched pixels at specific positions (such as the central pixel) relative to background less plausible couples.

Author(s)

Nicolas Chanard

David Depierre

Robel A Tesfaye

Naomi Schickele

Refka Askri

Pascal Martin

Stéphane Schaack

Olivier Cuvier

See Also

Useful links:

- https://bioconductor.org/packages/HicAggR
- https://cuvierlab.github.io/HicAggR/
- https://github.com/CuvierLab/HicAggR
- Report bugs at https://github.com/CuvierLab/HicAggR/issues

`Aggregation` *Aggregation of matrices list.*

Description

Aggregates all the matrices of a list (or two lists in case of differential aggregation) into a single matrix. This function allows to apply different aggregation (average, sum, ...), and differential (subtraction, ratio, ...) functions.

Usage

```
Aggregation(
  ctrlMatrices = NULL,
  matrices = NULL,
  aggFun = "mean",
  diffFun = "substraction",
  scaleCorrection = FALSE,
  correctionArea = NULL,
  statCompare = FALSE
)
```

Arguments

`ctrlMatrices`	<listmatrix>: The matrices list to aggregate as control.
`matrices`	<listmatrix>: The matrices list to aggregate.
`aggFun`	: The function used to aggregate each pixel in matrix. If the parameter is a character so: • "50%" or "median" apply the median • "+" or "sum" apply the sum • other (Default) apply the mean
`diffFun`	: The function used to compute differential. If the parameter is character so: • "-", "substract" or "substraction" apply a substraction (Default) • "/" or "ratio" apply a ratio • "log2","log2-","log2/" or "log2ratio" apply a log2 on ratio • other apply a log2 on 1+ratio
`scaleCorrection`	: Whether a correction should be done on the median value take in ane noising area. (Default TRUE)
`correctionArea`	: Nested list of indice that define a noising area fore correction. List must contain in first an element "i" (row indices) then an element called "j" (columns indices). If NULL automatically take in upper left part of aggregated matrices. (Default NULL)
`statCompare`	: Whether a t.test must be apply to each pixel of the differential aggregated matrix.

Details

Aggregation

Value

A matrix

Examples

```
# Data
data(Beaf32_Peaks.gnr)
data(HiC_Ctrl.cmx_lst)
data(HiC_HS.cmx_lst)

# Index Beaf32
Beaf32_Index.gnr <- IndexFeatures(
    gRangeList = list(Beaf = Beaf32_Peaks.gnr),
    chromSizes = data.frame(seqnames = c("2L", "2R"),
        seqlengths = c(23513712, 25286936)),
    binSize = 100000
)

# Beaf32 <-> Beaf32 Pairing
```

```
Beaf_Beaf.gni <- SearchPairs(indexAnchor = Beaf32_Index.gnr)
# subset 2000 first for exemple
Beaf_Beaf.gni <- Beaf_Beaf.gni[seq_len(2000)]

# Matrices extractions center on Beaf32 <-> Beaf32 point interaction
interactions_Ctrl.mtx_lst <- ExtractSubmatrix(
    genomicFeature = Beaf_Beaf.gni,
    hicLst = HiC_Ctrl.cmx_lst,
    referencePoint = "pf"
)
interactions_HS.mtx_lst <- ExtractSubmatrix(
    genomicFeature = Beaf_Beaf.gni,
    hicLst = HiC_HS.cmx_lst,
    referencePoint = "pf"
)
interactions_Ctrl.mtx_lst <- PrepareMtxList(
    matrices = interactions_Ctrl.mtx_lst
)

# Aggregate matrices in one matrix
aggreg.mtx <- Aggregation(interactions_Ctrl.mtx_lst)


interactions_HS.mtx_lst <- PrepareMtxList(
    matrices = interactions_HS.mtx_lst
)

# Differential Aggregation
aggregDiff.mtx <- Aggregation(
    ctrlMatrices = interactions_Ctrl.mtx_lst,
    matrices = interactions_HS.mtx_lst
)
```

`BalanceHiC` *Compute HiC matrix-balancing.*

Description

Apply a matrix-balancing normalization method to a list of contacts matrix.

Usage

```
BalanceHiC(
  hicLst,
  method = "ICE",
  interactionType = NULL,
  maxIter = 50,
  qtlTh = 0.15,
  cores = 1,
```

```
  verbose = FALSE
)
```

Arguments

hicLst	<ListContactMatrix>: The HiC maps list.
method	: The kind of normalization method. One of "ICE", "VC" or "VC_SQRT" (Default "ICE")
interactionType	: "cis", "trans", c("cis", "trans"), "all". If NULL normalization is apply on cis contactMatrix then trans contactMatrix (equivalent to c("cis", "trans")). If is "all", normalization is apply on all contactMatrix at once. (Default NULL)
maxIter	: The maximum iteration number.
qtlTh	: The quantile threshold below which the bins will be ignored. (Default 0.15)
cores	: Number of cores to be used. (Default 1)
verbose	: If TRUE show the progression in console. (Default FALSE)

Details

BalanceHiC

Value

A matrices list.

Examples

```
data(HiC_Ctrl.cmx_lst)

HiC_Ctrl_ICE.cmx_lst <- BalanceHiC(HiC_Ctrl.cmx_lst,
    interactionType = "cis",
    method = "ICE"
)

HiC_Ctrl_VC.cmx_lst <- BalanceHiC(HiC_Ctrl.cmx_lst,
    interactionType = c("cis", "trans"),
    method = "VC"
)

HiC_Ctrl_VC_SQRT.cmx_lst <- BalanceHiC(HiC_Ctrl.cmx_lst,
    interactionType = "all",
    method = "VC_SQRT"
)
```

Beaf32_Peaks.gnr *D.melanogaster Beaf-32 ChIP-seq.*

Description

Drosophila Melanogaster Beaf32 peaks on 2L and 2R chromosomes.

Usage

```
data(Beaf32_Peaks.gnr)
```

Format

An object of class GRanges.

Examples

```
data(Beaf32_Peaks.gnr)
Beaf32_Peaks.gnr
```

BinGRanges *Bin a GRanges.*

Description

Bin a GRanges and apply a summary method (e.g: 'mean', 'median', 'sum', 'max, 'min' ...) to a chosen numerical variable of ranges in the same bin.

Usage

```
BinGRanges(
  gRange = NULL,
  chromSizes = NULL,
  binSize = NULL,
  method = "mean",
  metadataColName = NULL,
  na.rm = TRUE,
  cores = 1,
  reduceRanges = TRUE,
  verbose = FALSE
)
```

Arguments

gRange	: A GRanges to bin.
chromSizes	<data.frame>: A data.frame where first colum corresponds to the chromosomes names, and the second column corresponds to the chromosomes lengths in base pairs.
binSize	: Width of the bins.
method	: Name of a summary method as 'mean', 'median', 'sum', 'max, 'min'. (Default 'mean')
metadataColName	: A character vector that specify the metadata columns of GRanges on which to apply the summary method.
na.rm	: A logical value indicating whether NA values should be stripped before the computation proceeds. (Default TRUE)
cores	: The number of cores. (Default 1)
reduceRanges	: Whether duplicated Bins must be reduced with the summary method (method arg). (Default TRUE)
verbose	: If TRUE, show the progression in console. (Default FALSE)

Details

BinGRanges

Value

A binned GRanges.

Examples

```
GRange.gnr <- GenomicRanges::GRanges(
    seqnames = S4Vectors::Rle(c("chr1", "chr2"), c(3, 1)),
    ranges = IRanges::IRanges(
        start = c(1, 201, 251, 1),
        end = c(200, 250, 330, 100),
        names = letters[seq_len(4)]
    ),
    strand = S4Vectors::Rle(BiocGenerics::strand(c("*")), 4),
    score = c(50, NA, 100, 30)
)
GRange.gnr
BinGRanges(
    gRange = GRange.gnr,
    chromSizes = data.frame(c("chr1", "chr2"), c(350, 100)),
    binSize = 100,
    method = "mean",
    metadataColName = "score",
    na.rm = TRUE
)
```

CompareToBackground *CompareToBackground*

Description

Computes z.test for each target couple over background couples.

Usage

```
CompareToBackground(
  hicList = NULL,
  matrices = NULL,
  indexAnchor = NULL,
  indexBait = NULL,
  genomicConstraint = NULL,
  secondaryConst.var = NULL,
  chromSizes = NULL,
  n_background = NULL,
  areaFun = "center",
  operationFun = "mean",
  bg_type = NULL,
  cores = 1,
  verbose = FALSE,
  p_adj_method = "BH",
  ...
)
```

Arguments

hicList	<ListContactMatrix>: The HiC maps list.
matrices	<listmatrix>: The matrices list.
indexAnchor	: A first indexed GRanges object used as pairs anchor (must be indexed using IndexFeatures()).
indexBait	: A second indexed GRanges object used as pairs bait (must be indexed using IndexFeatures()).
genomicConstraint	: GRanges object of constraint regions. If NULL, chromosomes in chromSizes are used as constraints (Default NULL)
secondaryConst.var	: A string defining column name containing compartment information in the metadata of anchor and bait objects. (Default NULL)
chromSizes	<data.frame>: A data.frame containing chromosomes names and lengths in base pairs.
n_background	: Number of background couples to keep. We recommend to use set.seed prior to launching this function with non null n_background. (Default NULL)

areaFun : A character or function that allows to extract an area from each matrix that composes the matrices list (Default "center"). Look at GetQuantif for more info.

operationFun : A character or function specifying the operation applied to the selected area. (Default "mean"). Look at GetQuantif for more info.

bg_type : Type of background couples to generate. Possible choices: "random_anchors", "inter_TAD", "inter_compartment", NULL (Defaults to "random_anchors"). More information in details...

cores : Number of cores used. (Default 1)

verbose details on progress? (Default: FALSE)

p_adj_method method used to adjust p.values. More from `stats::p.adjust()`. (Default: "BH")

... arguments to pass to PrepareMtxList, inorder to treat background matrices.

Details

Types of background couples possible:

- "random_anchors": picks random bins as anchors and forms couples with bait bins. If genomicConstraint is supplied, only intra-TAD random-bait couples are kept. Else intra-TAD random-bait couples within a distance constraint corresponding to the minimal and maximal distances of target couples.
- "inter_TAD": If target couples were formed using TAD information with non `NULL` genomicConstraint argument, then inter-TAD anchor-bait couples are used as background. Distance constraint applied correspond to the minimal distance of target couples and maximal width of supplied TADs.
- "inter_compartment": If `secondaryConst.var` is not `NULL` and both indexAnchor and indexBait objects contain the provided variable name, then background couples are formed between anchors and baits located in different compartments.
- "NULL": If `NULL`, `random_anchors` are set by default.

Notes on the comparison between bg and target couples: We noticed that o/e values tend to be skewed towards very long distance interactions. As a result, long distance background couples tend to influence strongly mean and sd, resulting in more long distance target couples being significant. So rather than computing z.score over all background couples, we've chosen to fit a polynomial with 2 degrees on the log(counts) vs distance data of the background couples. Z.scores are then computed per target couple by comparing residuals of the target counts as predicted by the model and the residuals of the background couples.

Value

returns a object with the z.test output for each target couple, values for the target couples and values for the background couples.

Examples

```
h5_path <- system.file("extdata",
    "Control_HIC_10k_2L.h5",
    package = "HicAggR", mustWork = TRUE
)
binSize=10000
data(Beaf32_Peaks.gnr)
data(TADs_Domains.gnr)
hicLst <- ImportHiC(
  file      = h5_path,
  hicResolution       = binSize,
  chromSizes = data.frame(seqnames = c("2L"),
  seqlengths = c(23513712)),
  chrom_1   = c("2L")
)
hicLst <- BalanceHiC(hicLst)
hicLst <- OverExpectedHiC(hicLst)
# Index Beaf32
Beaf32_Index.gnr <- IndexFeatures(
  gRangeList = list(Beaf = Beaf32_Peaks.gnr),
  chromSizes = data.frame(seqnames = c("2L"),
   seqlengths = c(23513712)),
  genomicConstraint = TADs_Domains.gnr,
  binSize = binSize
)
Beaf_Beaf.gni <- SearchPairs(indexAnchor = Beaf32_Index.gnr)

interactions_Ctrl.mtx_lst <- ExtractSubmatrix(
 genomicFeature = Beaf_Beaf.gni,
 hicLst = hicLst,
 referencePoint = "pf"
)
interactions_Ctrl.mtx_lst <- PrepareMtxList(
 matrices = interactions_Ctrl.mtx_lst
)
output_bgInterTAD = CompareToBackground(hicList = hicLst,
 matrices = interactions_Ctrl.mtx_lst,
 indexAnchor = Beaf32_Index.gnr,
 indexBait = Beaf32_Index.gnr,
 genomicConstraint = TADs_Domains.gnr,
 chromSizes = data.frame(seqnames = c("2L"),
   seqlengths = c(23513712)),
 bg_type="inter_TAD"
)
```

`CutHiC` *Cut HiC map in chunks.*

Description

Cut a mega contactMatrix (joint from multiple chromosomic maps) into a list of contactMatrix.

Usage

```
CutHiC(megaHic, verbose = FALSE)
```

Arguments

megaHic	: The HiC megamap.
verbose	: If TRUE, show the progression in console. (Default FALSE)

Details

CutHiC

Value

A matrices list.

Examples

```
data(HiC_Ctrl.cmx_lst)
Mega_Ctrl.cmx <- JoinHiC(HiC_Ctrl.cmx_lst)
CutHiC(Mega_Ctrl.cmx)
```

ExtractSubmatrix *Submatrix extraction.*

Description

Extract matrices in the HiC maps list around genomic features.

Usage

```
ExtractSubmatrix(
  genomicFeature = NULL,
  hicLst = NULL,
  referencePoint = "pf",
  hicResolution = NULL,
  matriceDim = 21,
  shift = 1,
  remove_duplicates = TRUE,
  cores = 1,
  verbose = FALSE
)
```

Arguments

genomicFeature	<GRanges or PairsGRanges or GInteractions>: The genomic coordinates on which compute the extraction of HiC submatrix.
hicLst	<ListContactMatrix>: The HiC maps list.
referencePoint	: Type of extracted submatrices. "rf" for "region feature" to extract triangle-shaped matrices around regions or "pf" for "point feature" to extract square-shaped matrices around points. (Default "rf")
hicResolution	: The resolution used in hicLst. If NULL, automatically find in resolution attributes of hicLst. (Default NULL)
matriceDim	: The size of matrices in output. (Default 21).
shift	: Only when "referencePoint" is "rf". Factor defining how much of the distance between anchor and bait is extracted before and after the region (Default 1). Ex: for shift=2, extracted matrices will be: 2*regionSize+regionSize+2*regionSize.
remove_duplicates	: remove duplicated submatrices ? This avoids duplicated submatrices when both anchor and bait bins are from the same feature. ex. BEAF32-BEAF32, same submatrix twice with opposite orientations(Default TRUE)
cores	: An integer to specify the number of cores. (Default 1)
verbose	: If TRUE, show the progression in console. (Default FALSE)

Details

ExtractSubmatrix

Value

A matrices list.

Examples

```
# Data
data(Beaf32_Peaks.gnr)
data(HiC_Ctrl.cmx_lst)

# Index Beaf32
Beaf32_Index.gnr <- IndexFeatures(
    gRangeList = list(Beaf = Beaf32_Peaks.gnr),
    chromSizes = data.frame(seqnames = c("2L", "2R"),
        seqlengths = c(23513712, 25286936)),
    binSize = 100000
)

# Beaf32 <-> Beaf32 Pairing
Beaf_Beaf.gni <- SearchPairs(indexAnchor = Beaf32_Index.gnr)
Beaf_Beaf.gni <- Beaf_Beaf.gni[seq_len(2000)] # subset 2000 first for exemple

# Matrices extractions of regions defined between
# Beaf32 <-> Beaf32 interactions
```

```
interactions_RF.mtx_lst <- ExtractSubmatrix(
    genomicFeature = Beaf_Beaf.gni,
    hicLst = HiC_Ctrl.cmx_lst,
    referencePoint = "rf"
)

# Matrices extractions center on Beaf32 <-> Beaf32 pointinteraction
interactions_PF.mtx_lst <- ExtractSubmatrix(
    genomicFeature = Beaf_Beaf.gni,
    hicLst = HiC_Ctrl.cmx_lst,
    referencePoint = "pf"
)
```

`FilterInteractions` *Submatrix or Interactions filtering.*

Description

Search in a GInteraction object which interactions correspond to a target list and return a list of index or filter a matrices list according to target and a selectionFunction.

Usage

```
FilterInteractions(
  matrices = NULL,
  genomicInteractions = NULL,
  targets = NULL,
  selectionFun = function() {
     Reduce(intersect, interarctions.ndx_lst)
 }
)
```

Arguments

`matrices`	<Listmatrix>: The matrices list to filter. If is not `NULL`, the function will return the filtred matrices list, else it will return a list of index.
`genomicInteractions`	: The GInteraction object on which to compute the filter.
`targets`	: A named list that describe the target.
`selectionFun`	: A function that defines how the target variables must be crossed. (Default intersection of all targets)

Details

FilterInteractions

Value

A list of elements index or a filtred matrices list with attributes updates.

Examples

```
# Data
data(Beaf32_Peaks.gnr)
data(HiC_Ctrl.cmx_lst)

# Index Beaf32
Beaf32_Index.gnr <- IndexFeatures(
    gRangeList = list(Beaf = Beaf32_Peaks.gnr),
    chromSizes  = data.frame(seqnames = c("2L", "2R"),
        seqlengths = c(23513712, 25286936)),
    binSize    = 100000
)

# Beaf32 <-> Beaf32 Pairing
Beaf_Beaf.gni <- SearchPairs(indexAnchor = Beaf32_Index.gnr)
Beaf_Beaf.gni <- Beaf_Beaf.gni[seq_len(2000)] # subset 2000 first for exemple

# Matrices extractions center on Beaf32 <-> Beaf32 point interaction
interactions_PF.mtx_lst  <- ExtractSubmatrix(
    genomicFeature         = Beaf_Beaf.gni,
    hicLst        = HiC_Ctrl.cmx_lst,
    referencePoint = "pf"
)

# Create a target
targets <- list(
    anchor.Beaf.name = c("Beaf32_108", "Beaf32_814"),
    distance         = function(dist) {
        dist < 300000
    }
)
# We target the Beaf32<->Beaf32 interactions that are less than 300Kb away
# and have peak Beaf32_2 and Beaf32_191 as anchors (i.e. left).

# Create a selection
selectionFun <- function() {
    intersect(anchor.Beaf.name, distance)
}
# We select the Beaf32<->Beaf32 interactions that satisfy both targeting
# criteria (intersection).

# Filtration on InteractionSet (Beaf32 <-> Beaf32 Pairs)
FilterInteractions(
    genomicInteractions = Beaf_Beaf.gni,
    targets        = targets,
    selectionFun     = NULL
) |> str(max.level = 1)
# Returns a named list (the names match the targeting criteria).
```

```
# Each element is an index vector of Beaf32<->Beaf32 interactions
# that satisfy the given criteria.

# Filtration on Matrices List (Beaf32 <-> Beaf32 Extracted matrices)
FilterInteractions(
    matrices      = interactions_PF.mtx_lst,
    targets        = targets,
    selectionFun     = NULL
) |> str(max.level = 1)
# Return the same kind of result.

# Add the selection on InteractionSet Filtration
FilterInteractions(
    genomicInteractions = Beaf_Beaf.gni,
    targets = targets,
    selectionFun = selectionFun
) |> str(max.level = 1)
# This return the intersection of the index vector that satisfy both
# targeting criteria.

# Add the selection on Matrices List Filtration
FilterInteractions(
    matrices = interactions_PF.mtx_lst,
    targets = targets,
    selectionFun = selectionFun
) |> str(max.level = 1)
# This return the filtred matrices, i.e the matrices for which
# the Beaf32<->Beaf32 interactions satisfy both targeting criteria.

# Filtration with InteractionsSet as filtration criteria
targets <- list(interactions = Beaf_Beaf.gni[seq_len(2)])
FilterInteractions(
    genomicInteractions = Beaf_Beaf.gni,
    targets = targets,
    selectionFun = NULL
) |> str(max.level = 1)


# Filtration with GRanges as filtration criteria
targets <- list(first =
    InteractionSet::anchors(Beaf_Beaf.gni)[["first"]][seq_len(2)])
FilterInteractions(
    genomicInteractions = Beaf_Beaf.gni,
    targets = targets,
    selectionFun = NULL
) |> str(max.level = 1)
```

GenomicSystem *Base pairs convertion.*

Description

Convert numbers of base into string with order of magnitude (Kbp, Mbp, Gbp) and vice versa.

Usage

```
GenomicSystem(x, digits = 3)
```

Arguments

x	: The number to convert or string to convert.
digits	: The number of significant digits to be used. See signif() for more informations. (Default 3)

Details

GenomicSystem

Value

The converted number or string.

Examples

```
GenomicSystem(1540, 3)
GenomicSystem(1540, 2)
GenomicSystem("1Mbp")
GenomicSystem("1Kbp")
GenomicSystem("1k")
```

GetInfo *get some basic information about your hic file*

Description

GetInfo

Usage

```
GetInfo(file = NULL, printInfos = TRUE, returnInfos = FALSE)
```

Arguments

file	path to your file.
printInfos	print info on console? (Default: TRUE)
returnInfos	return info? (Default: FALSE)

Value

list of characters if `returnInfos` is TRUE.

Examples

```
h5_path <- system.file("extdata",
    "Control_HIC_10k_2L.h5",
    package = "HicAggR", mustWork = TRUE
)
GetInfo(h5_path)
```

`GetQuantif` *Compute quantification on extracted submatrices.*

Description

Function that computes quantification of contact frequencies in a given area and returns it in a named vector.

Usage

```
GetQuantif(
  matrices,
  areaFun = "center",
  operationFun = "mean_rm0",
  varName = NULL
)
```

Arguments

`matrices`	<Listmatrix>: A matrices list.
`areaFun`	: A character or function that allows to extract an area from each matrix that composes the matrices list (Default "center").

- "C" or "CENTER": 3x3 pixels at the intersection between anchor and bait.
- "UL" or "UPPER_LEFT": pixels in the uppper left square
- "UR" or "UPPER_RIGHT": pixels in the uppper right square
- "BL" or "BOTTOM_LEFT": pixels in the bottom left square
- "BR" or "BOTTOM_RIGHT": pixels in the bottom right square
- "U" or "UPPER": pixels above the center area
- "B" or "BOTTOM": pixels below the center area
- "L" or "LEFT": pixels in the left of the center area
- "R" or "RIGHT": pixels in the right of the center area
- "D" or "DONUT": pixels that surrounds the center area

operationFun : A character or function specifying the operation applied to the selected area. (Default "mean_rm0")

- "mean_rm0": apply a mean after replacing 0 with NA
- "median_rm0": apply a median after replacing 0 with NA
- "sum_rm0": apply a sum after replacing 0 with NA
- "median": apply a median
- "sum": apply a sum
- "mean" or other character: apply a mean

varName : The name of a column in GInteraction attributes of matrices used as named in the output vector (Default NULL). By default, sub-matrices IDs are used.

Details

GetQuantif

Value

A GRange object.

Examples

```
# Data
data(Beaf32_Peaks.gnr)
data(HiC_Ctrl.cmx_lst)

# Index Beaf32
Beaf32_Index.gnr <- IndexFeatures(
    gRangeList = list(Beaf = Beaf32_Peaks.gnr),
    chromSizes = data.frame(
        seqnames = c("2L", "2R"),
        seqlengths = c(23513712, 25286936)
    ),
    binSize = 100000
)

# Beaf32 <-> Beaf32 Pairing
Beaf_Beaf.gni <- SearchPairs(indexAnchor = Beaf32_Index.gnr)
Beaf_Beaf.gni <- Beaf_Beaf.gni[seq_len(2000)] # subset 2000 first for exemple

# Matrices extractions center on Beaf32 <-> Beaf32 point interaction
interactions_PF.mtx_lst <- ExtractSubmatrix(
    genomicFeature = Beaf_Beaf.gni,
    hicLst = HiC_Ctrl.cmx_lst,
    referencePoint = "pf"
)
GetQuantif(
    matrices = interactions_PF.mtx_lst,
    areaFun = "center",
    operationFun = "mean"
) |> head()
```

ggAPA *Aggregation plot*

Description

Create a ggplot object used for plot aggregation.

Usage

```
ggAPA(
  aggregatedMtx = NULL,
  title = NULL,
  trim = 0,
  tails = "both",
  colMin = NULL,
  colMid = NULL,
  colMax = NULL,
  colBreaks = NULL,
  blurPass = 0,
  boxKernel = NULL,
  kernSize = NULL,
  stdev = 0.5,
  loTri = NULL,
  colors = NULL,
  na.value = "#F2F2F2",
  colorScale = "linear",
  bias = 1,
  paletteLength = 51,
  annotate = TRUE,
  anchor.name = "Anchor",
  bait.name = "Bait",
  fixCoord = TRUE
)
```

Arguments

aggregatedMtx	: The matrix to plot. (Default NULL)
title	: The title of plot. (Default NULL)
trim	: A number between 0 and 100 that gives the percentage of trimming. (Default 0)
tails	: Which boundary must be trimmed? If it's both, trim half of the percentage in inferior and superior. see QtlThreshold. (Default "both")
colMin	: Minimal value of Heatmap, force color range. If NULL automatically find. (Default NULL)
colMid	: Center value of Heatmap, force color range. If NULL automatically find. (Default NULL)

colMax	: Maximal value of Heatmap, force color range. If NULL automatically find. (Default NULL)
colBreaks	: Repartition of colors. If NULL automatically find. (Default NULL)
blurPass	: Number of blur pass. (Default 0)
boxKernel	: If NULL automatically compute for 3 Sd. (Default NULL)
kernSize	: Size of box applied to blurr. If NULL automatically compute for 3 Sd. (Default NULL)
stdev	: SD of gaussian smooth. (Default 0.5)
loTri	: The value that replace all value in the lower triangle of matrice (Usefull when blur is apply).(Default NULL)
colors	: Heatmap color list. If NULL, automatically compute. (Default NULL)
na.value	: Color of NA values. (Default "#F2F2F2")
colorScale	: Shape of color scale on of "linear" or "density" based. (Default "linear")
bias	: A positive number. Higher values give more widely spaced colors at the high end. See ?grDevices::colorRamp for more details. (Default 1)
paletteLength	: The number of color in the palette. (Default 51)
annotate	: Should there be axis ticks? (Default: TRUE)
anchor.name	Name of anchor for annotation. (Default "Anchor")
bait.name	Name of bait for annotation. (Default "Bait")
fixCoord	Fix axes coordinates? (Default TRUE)

Details

ggAPA

Value

A ggplot object.

Examples

```
# Data
data(Beaf32_Peaks.gnr)
data(HiC_Ctrl.cmx_lst)

# Index Beaf32
Beaf32_Index.gnr <- IndexFeatures(
    gRangeList = list(Beaf = Beaf32_Peaks.gnr),
    chromSizes = data.frame(seqnames = c("2L", "2R"),
        seqlengths = c(23513712, 25286936)),
    binSize = 100000
)

# Beaf32 <-> Beaf32 Pairing
Beaf_Beaf.gni <- SearchPairs(indexAnchor = Beaf32_Index.gnr)
Beaf_Beaf.gni <- Beaf_Beaf.gni[seq_len(2000)]
```

```
# subset 2000 first for exemple

# Matrices extractions center on Beaf32 <-> Beaf32 point interaction
interactions_PF.mtx_lst <- ExtractSubmatrix(
    genomicFeature = Beaf_Beaf.gni,
    hicLst = HiC_Ctrl.cmx_lst,
    referencePoint = "pf"
)

# Aggregate matrices in one matrix
aggreg.mtx <- Aggregation(interactions_PF.mtx_lst)

# Visualization
ggAPA(
    aggregatedMtx = aggreg.mtx
)

# Add Title
ggAPA(
    aggregatedMtx = aggreg.mtx,
    title = "APA"
)

# Trim values
ggAPA(
    aggregatedMtx = aggreg.mtx,
    title = "APA 30% trimmed on upper tail of distribution",
    trim = 30,
    tails = "upper"
)
ggAPA(
    aggregatedMtx = aggreg.mtx,
    title = "APA 30% trimmed on lower tail of distribution",
    trim = 30,
    tails = "lower"
)
ggAPA(
    aggregatedMtx = aggreg.mtx,
    title = "APA 15% trimmed on each tail of distribution",
    trim = 30,
    tails = "both"
)

# Change Minimal, Central and Maximal Colors scale value
ggAPA(
    aggregatedMtx = aggreg.mtx,
    title = "APA [min 200, center 300, max 600]",
    colMin = 200,
    colMid = 300,
    colMax = 600
)

# Change Color
```

```
ggAPA(
    aggregatedMtx = aggreg.mtx,
    title = "APA",
    colors = viridis(6),
    na.value = "black"
)
ggAPA(
    aggregatedMtx = aggreg.mtx,
    title = "APA",
    colors = c("black", "white"),
)

# Change Color distribution
ggAPA(
    aggregatedMtx = aggreg.mtx,
    title = "APA [100,150,200,250,300,350,600]",
    colBreaks = c(100, 150, 200, 250, 300, 350, 600) # Choosen Breaks
)
ggAPA(
    aggregatedMtx = aggreg.mtx,
    title = "APA",
    colorScale = "density" # color distribution based on density
)
ggAPA(
    aggregatedMtx = aggreg.mtx,
    title = "APA",
    bias = 2 # (>1 wait on extremums)
)
ggAPA(
    aggregatedMtx = aggreg.mtx,
    title = "APA",
    bias = 0.5 # (<1 wait on center)
)

# Apply a Blurr
ggAPA(
    aggregatedMtx = aggreg.mtx,
    title = "APA",
    blurPass = 1,
    stdev = 0.5
)

# ggplot2 object modifications
# Since the function returns a ggplot object, it is possible
# to modify it following the ggplot2 grammar.
ggAPA(
    aggregatedMtx = aggreg.mtx,
    title = "APA",
) +
    ggplot2::labs(
        title = "New title",
        subtitle = "and subtitle"
    )
```

HiC_Ctrl.cmx_lst *In situ Hi-C control.*

Description

In situ Hi-C on non-heat treated S2 cells (Drosophila Melanogaster) with MboI on chromosome 2R and 2L download from 4DN portal (Ray J, Munn PR, et al., 2019). This data is the result of the `ImportHiC()` function.

Usage

```
data(HiC_Ctrl.cmx_lst)
```

Format

A a list of ContactMatrix objects. Each element correspond to the interaction matrix of two chromosomes.

Examples

```
data(HiC_Ctrl.cmx_lst)
HiC_Ctrl.cmx_lst
```

HiC_HS.cmx_lst *In situ Hi-C heat treated.*

Description

In situ Hi-C on heat treated S2 cells (Drosophila Melanogaster) with MboI on chromosome 2R and 2L download from 4DN portal (Ray J, Munn PR, et al., 2019). This data is the result of the `ImportHiC()` function.

Usage

```
data(HiC_HS.cmx_lst)
```

Format

A a list of ContactMatrix objects. Each element correspond to the interaction matrix of two chromosomes.

Examples

```
data(HiC_HS.cmx_lst)
HiC_HS.cmx_lst
```

Hue *Hue palette.*

Description

Create an Hue palette.

Usage

```
Hue(
  paletteLength = 9,
  rotation = NULL,
  hueRange = c(0, 360),
  saturation = 0.65,
  lightness = 0.65,
  alphaValue = 1,
  alpha = FALSE
)
```

Arguments

paletteLength	: Color number.
rotation	: If positive, rotates clockwise in the color space, reversing if the number is negative. If is NULL, compute rotation according to hueRange parameter. (Default NULL)
hueRange	: Degree range in color space between 0 and 360. (Default c(0,360))
saturation	: Saturation value between 0 and 1. (Default 0.65)
lightness	: Lightness value between 0 and 1. (Default 0.65)
alphaValue	: Opacity value between 0 and 1. (Default 1)
alpha	: Whether the alpha layer should be returned. (Default FALSE)

Details

Hue

Value

A vector of color.

Examples

```
Hue(paletteLength = 9)
```

ICEnorm *Compute Iterative Correction.*

Description

Compute Iterative Correction (Vanilla Count) on hic maps.

Usage

```
ICEnorm(hic, qtlTh = 0.15, maxIter = 50)
```

Arguments

hic	: The HiC maps chunk to normalize.
qtlTh	: The threshold quantile below which the bins will be ignored. (Default 0.15)
maxIter	: The maximum iteration number.

Details

ICEnorm

Value

A normalized contactMatrix

Examples

```
data(HiC_Ctrl.cmx_lst)
HiC_Ctrl_ICE.cmx <- ICEnorm(HiC_Ctrl.cmx_lst[['2L_2L']])
```

ImportHiC *Import Hic data*

Description

Import ..hic, .cool, .mcool or .bedpe data

Usage

```
ImportHiC(
  file = NULL,
  hicResolution = NULL,
  chromSizes = NULL,
  chrom_1 = NULL,
  chrom_2 = NULL,
  verbose = FALSE,
  cores = 1,
  hic_norm = "NONE",
  hic_matrix = "observed",
  cool_balanced = FALSE,
  cool_weight_name = "weight",
  cool_divisive_weights = FALSE,
  h5_fill_upper = TRUE
)
```

Arguments

file	<GRanges or PairsGRanges or GInteractions>: The genomic feature on which compute the extraction of HiC submatrix. Extension should be .hic, .cool, .mcool, .h5, .hdf5, .HDF5 or .bedpe" assuming .h5 and .hdf5 are only for cool (not mcool).
hicResolution	: The HiC resolution.
chromSizes	<data.frame>: A data.frame where first colum correspond to the chromosomes names, and the second column correspond to the chromosomes lengths in base pairs.
chrom_1	: The seqnames of first chromosmes (rows in matrix).
chrom_2	: The seqnames of second chromosmes (col in matrix). If NULL variable will be assigned value of chrom_1 (Defalt NULL).
verbose	: Show the progression in console? (Default FALSE)
cores	: An integer to specify the number of cores. (Default 1)
hic_norm	: "norm" argument to supply to strawr::straw(). This argument is for .hic format data only. Available norms can be obtained through strawr::readHicNormTypes(). (Default "NONE").
hic_matrix	: "matrix" argument to supply to strawr::straw(). This argument is for .hic format data only. Other options can be: "oe", "expected". (Default "observed").
cool_balanced	Import already balanced matrix? (Default: FALSE)
cool_weight_name	Name of the correcter in the cool file. (Default: weight). rhdf5::h5ls() to see the available correctors.
cool_divisive_weights	Does the correcter vector contain divisive biases as in hicExplorer or multiplicative as in cooltools? (Default: FALSE)
h5_fill_upper	Do the matrix in h5 format need to be transposed? (Default: TRUE)

Details

ImportHiC

If you request "expected" values when importing .hic format data, you must do yourself the "oe" by importing manually the observed counts as well.

Prior to v.0.9.0 cooltools had multiplicative weight only, so make sure your correcters are divisive or multiplicative. https://cooler.readthedocs.io/en/stable/releasenotes.html#v0-9-0

When loading hic matrix in h5 format make sure you have enough momory to load the full matrix with all chromosomes regardless of values for chrom_1 and chrom_2 arguments. The function first loads the whole matrix, then extracts matrices per chromosome for the time being, it's easier ;).

Value

A matrices list.

Examples

```
# Prepare Temp Directory
options(timeout = 3600)
temp.dir <- file.path(tempdir(), "HIC_DATA")
dir.create(temp.dir)

# Download .hic file
Hic.url <- paste0(
    "https://4dn-open-data-public.s3.amazonaws.com/",
    "fourfront-webprod/wfoutput/",
    "7386f953-8da9-47b0-acb2-931cba810544/4DNFIOTPSS3L.hic"
)
HicOutput.pth <- file.path(temp.dir, "Control_HIC.hic")
HicOutput.pth <- normalizePath(HicOutput.pth)
if(.Platform$OS.type == "windows"){
    download.file(Hic.url, HicOutput.pth, method = "auto",
    extra = "-k",mode="wb")
}else{
    download.file(Hic.url, HicOutput.pth, method = "auto", extra = "-k")
}

# Import .hic file
HiC_Ctrl.cmx_lst <- ImportHiC(
    file = HicOutput.pth,
    hicResolution = 100000,
    chrom_1 = c("2L", "2L", "2R"),
    chrom_2 = c("2L", "2R", "2R")
)

# Download .mcool file
Mcool.url <- paste0(
    "https://4dn-open-data-public.s3.amazonaws.com/",
    "fourfront-webprod/wfoutput/",
    "4f1479a2-4226-4163-ba99-837f2c8f4ac0/4DNFI8DRD739.mcool"
```

```
)
McoolOutput.pth <- file.path(temp.dir, "HeatShock_HIC.mcool")
HicOutput.pth <- normalizePath(McoolOutput.pth)
if(.Platform$OS.type == "windows"){
    download.file(Mcool.url, McoolOutput.pth, method = "auto",
    extra = "-k",mode="wb")
}else{
    download.file(Mcool.url, McoolOutput.pth, method = "auto",
    extra = "-k")
}

# Import .mcool file
HiC_HS.cmx_lst <- ImportHiC(
    file = McoolOutput.pth,
    hicResolution = 100000,
    chrom_1 = c("2L", "2L", "2R"),
    chrom_2 = c("2L", "2R", "2R")
)

# Import .h5 file
h5_path <- system.file("extdata",
    "Control_HIC_10k_2L.h5",
    package = "HicAggR", mustWork = TRUE
)
binSize=10000
hicLst <- ImportHiC(
  file      = h5_path,
  hicResolution       = binSize,
  chromSizes = data.frame(seqnames = c("2L"),
  seqlengths = c(23513712)),
  chrom_1   = c("2L")
)
```

`IndexFeatures` *Indexes GRanges on genome.*

Description

Function that indexes a GRanges object on binned genome and constraints. Needed prior HicAggR::SearchPairs() function.

Usage

```
IndexFeatures(
  gRangeList = NULL,
  genomicConstraint = NULL,
  chromSizes = NULL,
  binSize = NULL,
  method = "mean",
  metadataColName = NULL,
```

```
  cores = 1,
  verbose = FALSE
)
```

Arguments

gRangeList	<GRanges or GRangesList or listGRanges>: GRanges object, list of GRanges or GRangesList containing coordinates to index.
genomicConstraint	: GRanges object of constraint regions. Note that bins in the same constraint region only will be paired in HicAggR::SearchPairs(). If NULL chromosomes in chromSizes are used as constraints (Default NULL)
chromSizes	<data.frame>: A data.frame containing chromosomes names and lengths in base pairs (see example).
binSize	: Bin size in bp - corresponds to matrix resolution.
method	: A string defining which summary method is used on metadata columns defined in metadataColName if multiple ranges are indexed in the same bin. Use 'mean', 'median', 'sum', 'max' or 'min'. (Default 'mean")
metadataColName	: A character vector that specify the metadata columns of GRanges on which to apply the summary method if multiple ranges are indexed in the same bin.
cores	: Number of cores used. (Default 1)
verbose	: Show the progression in console? (Default FALSE)

Details

IndexFeatures

Value

A GRanges object.

Examples

```
data(Beaf32_Peaks.gnr)
Beaf32_Index.gnr <- IndexFeatures(
    gRangeList = list(Beaf = Beaf32_Peaks.gnr),
    chromSizes = data.frame(
        seqnames = c("2L", "2R"),
        seqlengths = c(23513712, 25286936)
    ),
    binSize = 100000
)
```

JoinHiC *Merge HiC chunk.*

Description

Create mega contactMatrix from a list of contactMatrix.

Usage

```
JoinHiC(hicLst)
```

Arguments

hicLst	<ListContactMatrix>: The HiC maps list.

Details

JoinHiC

Value

A ContactMatrix.

Examples

```
data(HiC_Ctrl.cmx_lst)
Mega_Ctrl.cmx <- JoinHiC(HiC_Ctrl.cmx_lst)
```

MergeGRanges *Merge GRanges.*

Description

Merge GRanges or a list of GRanges

Usage

```
MergeGRanges(..., sortRanges = FALSE, reduceRanges = FALSE)
```

Arguments

...	<GRanges or GRangesList or listGRanges>: Some GRanges or a list of GRanges or a GRangesList.
sortRanges	: Whether the result should be sorted. (Default FALSE)
reduceRanges	: Whether the result should be reduced. See GenomicRanges::reduce for more details. (Default FALSE)

Details

MergeGRanges

Value

a GRange object.

Examples

```
GRange_1.grn <- GenomicRanges::GRanges(
    seqnames = S4Vectors::Rle(c("chr1", "chr2", "chr1"), c(1, 3, 1)),
    ranges = IRanges::IRanges(101:105, end = 111:115,
        names = letters[seq_len(5)]),
    strand = S4Vectors::Rle(BiocGenerics::strand(c("-", "+", "*", "+")),
        c(1, 1, 2, 1)),
    score = seq_len(5)
)
GRange_2.grn <- GenomicRanges::GRanges(
    seqnames = S4Vectors::Rle(c("chr1", "chr3"), c(1, 4)),
    ranges = IRanges::IRanges(106:110, end = 116:120, names = letters[6:10]),
    strand = S4Vectors::Rle(BiocGenerics::strand(c("*", "+", "-")),
        c(2, 1, 2)),
    score = 6:10
)
GRange_1.grn
GRange_2.grn
MergeGRanges(GRange_1.grn, GRange_2.grn)
GRange.lst <- list(GRange_1.grn, GRange_2.grn)
MergeGRanges(GRange.lst)
MergeGRanges(GRange.lst, reduceRanges = TRUE)
```

OrientateMatrix *Matrix orientation*

Description

Oriente extracted Matrix according to the anchors and bait order. Apply a 180° rotation follow with a transposation on a matrix or on matrices in a list according to the interactions attributes of the list.

Usage

```
OrientateMatrix(mtx, verbose = TRUE)
```

Arguments

mtx	<matrix or Listmatrix>: Matrix or matricies list to oriente
verbose	: Report the number of matrices corrected. (Default: TRUE)

Details

OrientateMatrix

Value

Oriented matrix or matricies list

Examples

```
# Data
data(Beaf32_Peaks.gnr)
data(HiC_Ctrl.cmx_lst)

# Index Beaf32 in TADs domains
Beaf32_Index.gnr <- IndexFeatures(
    gRangeList = list(Beaf = Beaf32_Peaks.gnr),
    chromSizes = data.frame(
        seqnames = c("2L", "2R"),
        seqlengths = c(23513712, 25286936)
    ),
    binSize = 100000
)

# Beaf32 <-> Beaf32 Pairing
Beaf_Beaf.gni <- SearchPairs(indexAnchor = Beaf32_Index.gnr)
Beaf_Beaf.gni <- Beaf_Beaf.gni[seq_len(2000)] # subset 2000 first for exemple

# Matrices extractions center on Beaf32 <-> Beaf32 point interaction
interactions_PF.mtx_lst <- ExtractSubmatrix(
    genomicFeature = Beaf_Beaf.gni,
    hicLst = HiC_Ctrl.cmx_lst,
    referencePoint = "pf"
)

# Matrices Orientation
oriented_Interactions_PF.mtx_lst <- OrientateMatrix(interactions_PF.mtx_lst)
```

`OverExpectedHiC` *Genomic distance bias correction.*

Description

Function that normalises HiC matrices by expected values computed per genomic distance.

Usage

```
OverExpectedHiC(
  hicLst,
  method = "mean_non_zero",
  verbose = FALSE,
  cores = 1,
  plot_contact_vs_dist = "per_seq"
)
```

Arguments

hicLst	<ListContactMatrix>: The HiC maps list.
method	Options are "mean_non_zero", "mean_total", or "lieberman". Look at details for more. (Default: "mean_non_zero")
verbose	: Show the progression in console? (Default FALSE)
cores	: Number of cores to be used. (Default 1)
plot_contact_vs_dist	Whether to plot contact vs distance curve per chromosome ("per_seq"), all chromosomes ("total") or not (NULL). (Default "per_seq")

Details

OverExpectedHiC

Methods to calculate expected values per distance:

- "mean_non_zero": for each distance, average contact value is calculated using only non-zero values.
- "mean_total": for each distance, average contact value is calculated using all values at this distance.
- "lieberman": for each distance, contact values are summed and divided by chromsome length minus distance. Only for cis contacts.

Value

A matrices list.

Examples

```
# Note: run HicAggR::BalanceHiC before OverExpectedHiC calculation.
data(HiC_Ctrl.cmx_lst)
OverExpectedHiC(HiC_Ctrl.cmx_lst)
```

PlotAPA *Draw aggregation plot.*

Description

Draw aggregation plot from aggregation matrices.

Usage

```
PlotAPA(
  aggregatedMtx = NULL,
  trim = 0,
  colMin = NULL,
  colMid = NULL,
  colMax = NULL,
  colMinCond = NULL,
  colMaxCond = NULL,
  extra_info = FALSE,
  ...
)
```

Arguments

aggregatedMtx	: The aggregated matrix.
trim	: A number between 0 and 100 that gives the percentage of triming in matrices.
colMin	: The minimal value in color scale. If Null automaticaly find.
colMid	: The middle value in color scale. If Null automaticaly find.
colMax	: The mximal value in color scale. If Null automaticaly find.
colMinCond	: Avalaible for plotting differential aggregation. The minimal value in color scale in the classsical aggregation plot. If NULL automaticaly find.
colMaxCond	: Avalaible for plotting differantial aggregation. The maxiaml value in color scale in the classsical aggregation plot. If NULL automaticaly find.
extra_info	do you want to have a recall of your arguments values? (Default FALSE)
...	additional arguments to ggAPA()

Details

PlotAPA

Value

None

Examples

```
# Data
data(Beaf32_Peaks.gnr)
data(HiC_Ctrl.cmx_lst)

# Index Beaf32
Beaf32_Index.gnr <- IndexFeatures(
    gRangeList = list(Beaf = Beaf32_Peaks.gnr),
    chromSizes = data.frame(seqnames = c("2L", "2R"),
        seqlengths = c(23513712, 25286936)),
    binSize = 100000
)

# Beaf32 <-> Beaf32 Pairing
Beaf_Beaf.gni <- SearchPairs(indexAnchor = Beaf32_Index.gnr)
Beaf_Beaf.gni <- Beaf_Beaf.gni[seq_len(2000)] # subset 2000 first for exemple

# Matrices extractions center on Beaf32 <-> Beaf32 point interaction
interactions_PF.mtx_lst <- ExtractSubmatrix(
    genomicFeature = Beaf_Beaf.gni,
    hicLst = HiC_Ctrl.cmx_lst,
    referencePoint = "pf"
)

# Aggregate matrices in one matrix
aggreg.mtx <- Aggregation(interactions_PF.mtx_lst)

# Visualization
PlotAPA(
    aggregatedMtx = aggreg.mtx
)

PlotAPA(
    aggregatedMtx= aggreg.mtx,
    trim= 20,
    colMin= -2,
    colMid= 0,
    colMax= 2,
    colMinCond = 0,
    colMaxCond = 2
)
```

`plotMultiAPA` *Draw aggregation plots for interactions with different distances.*

Description

Separates matrices based on interaction distance, performs aggregation and plots Aggregated signal for each chunk of interaction distances.

Usage

```
plotMultiAPA(submatrices = NULL, ctrlSubmatrices = NULL, ..., plot.opts = NULL)
```

Arguments

submatrices	: The matrices list to separate using interaction distances and aggregate. Chunks of distances are created with: `c(0,50000*2 ^ seq(0,5,by=1))`. Other matrices with distances over 1.6 Mb are aggregated in the same final chunk.
ctrlSubmatrices	: The matrices list to use as control condition for differential aggregation.

...	: Additional arguments to pass to [Aggregation()]. For differential aggregation plot, `submatrices` will take the matrices of the treated condition. eg:
	`plotMultiAPA(`
	`submatrices = interactions_HS.mtx_lst,`
	`ctrlSubmatrices = interactions_Ctrl.mtx_lst)```
	
	`[Aggregation()]: R:Aggregation()`

plot.opts	list of arguments to pass to ggAPA().

Details

plotMultiAPA

Value

A plot with separate APAs per distance and a list of aggregated matrices as invisible output.

Examples

```
#' # Data
data(Beaf32_Peaks.gnr)
data(HiC_Ctrl.cmx_lst)
data(HiC_HS.cmx_lst)

# Index Beaf32
Beaf32_Index.gnr <- IndexFeatures(
    gRangeList = list(Beaf = Beaf32_Peaks.gnr),
    chromSizes = data.frame(seqnames = c("2L", "2R"),
        seqlengths = c(23513712, 25286936)),
    binSize = 100000
)

# Beaf32 <-> Beaf32 Pairing
Beaf_Beaf.gni <- SearchPairs(indexAnchor = Beaf32_Index.gnr)
Beaf_Beaf.gni <- Beaf_Beaf.gni[seq_len(2000)] # subset 2000 first for eg

# Matrices extractions center on Beaf32 <-> Beaf32 point interaction
interactions_Ctrl.mtx_lst <- ExtractSubmatrix(
    genomicFeature = Beaf_Beaf.gni,
```

```
    hicLst = HiC_Ctrl.cmx_lst,
    referencePoint = "pf"
)
interactions_HS.mtx_lst <- ExtractSubmatrix(
    genomicFeature = Beaf_Beaf.gni,
    hicLst = HiC_HS.cmx_lst,
    referencePoint = "pf"
)
interactions_Ctrl.mtx_lst <- PrepareMtxList(
    matrices = interactions_Ctrl.mtx_lst
)

# Aggregate matrices in one matrix
plotMultiAPA(submatrices = interactions_Ctrl.mtx_lst)


interactions_HS.mtx_lst <- PrepareMtxList(
    matrices = interactions_HS.mtx_lst
)

# Differential Aggregation
plotMultiAPA(
    submatrices = interactions_HS.mtx_lst,
    ctrlSubmatrices = interactions_Ctrl.mtx_lst,
    diffFun = "ratio",
    plot.opts = list(colors = list("blue","white","red"))
)
```

`PrepareMtxList` *Prepare matrices list for further analysis.*

Description

Prepares matrices list for further analysis (eg. Aggregation or GetQuantif). Orientation can be corrected, and per matrix transformation can be performed.

Usage

```
PrepareMtxList(
  matrices,
  minDist = NULL,
  maxDist = NULL,
  rm0 = FALSE,
  transFun = NULL,
  orientate = FALSE
)
```

Arguments

matrices	<listmatrix>: The matrices list to prepare.
minDist	: The minimal distance between anchor and bait.
maxDist	: The maximal distance between anchor and bait.
rm0	: Whether 0 should be replaced with NA. (Default FALSE)
transFun	: The function used to transform or scale values in each submatrix before aggregation. The following characters can be submitted: • "quantile" or "qtl" apply function dplyr::ntile(x,500) • "percentile" or "prct" apply percentile. • "rank" apply a ranking. • "zscore" apply a scaling. • "minmax" scales on a min to max range. • "mu" scales on mean: `(x-mean(x))/(max(x)-min(x))`. • other or NULL don't apply transformation (Default).
orientate	: Whether matrices must be corrected for orientatation before the aggregation.

Details

PrepareMtxList

Value

A matrix list ready for aggregation of values extraction.

Examples

```
# Data
data(Beaf32_Peaks.gnr)
data(HiC_Ctrl.cmx_lst)
data(HiC_HS.cmx_lst)

# Index Beaf32
Beaf32_Index.gnr <- IndexFeatures(
    gRangeList = list(Beaf = Beaf32_Peaks.gnr),
    chromSizes = data.frame(seqnames = c("2L", "2R"),
          seqlengths = c(23513712, 25286936)),
    binSize = 100000
)

# Beaf32 <-> Beaf32 Pairing
Beaf_Beaf.gni <- SearchPairs(indexAnchor = Beaf32_Index.gnr)
Beaf_Beaf.gni <- Beaf_Beaf.gni[seq_len(2000)] # subset 2000 first for exemple

# Matrices extractions center on Beaf32 <-> Beaf32 point interaction
interactions_Ctrl.mtx_lst <- ExtractSubmatrix(
    genomicFeature = Beaf_Beaf.gni,
    hicLst = HiC_Ctrl.cmx_lst,
    referencePoint = "pf"
```

```
)
interactions_HS.mtx_lst <- ExtractSubmatrix(
    genomicFeature = Beaf_Beaf.gni,
    hicLst = HiC_HS.cmx_lst,
    referencePoint = "pf"
)
interactions_Ctrl.mtx_lst <- PrepareMtxList(
    matrices = interactions_Ctrl.mtx_lst
)

# Aggregate matrices in one matrix
aggreg.mtx <- Aggregation(interactions_Ctrl.mtx_lst)


interactions_HS.mtx_lst <- PrepareMtxList(
    matrices = interactions_HS.mtx_lst
)

# Differential Aggregation
aggregDiff.mtx <- Aggregation(
    ctrlMatrices = interactions_Ctrl.mtx_lst,
    matrices = interactions_HS.mtx_lst
)
```

`preparePlotgardener` *preparePlotgardener*

Description

This function allows to obtain a dataframe that can be used with plotgardener's plotHicTriangle, plotHicRectangle, plotHicSquare. It is equivalent to plotgardener's readHic function.

Usage

```
preparePlotgardener(
  hicList = NULL,
  ctrlHicList = NULL,
  submatrices = NULL,
  submatrix.name = NULL,
  diffFun = "log2ratio",
  which_chrom = NULL,
  which_range = NULL
)
```

Arguments

`hicList`	hicList from which to extract data
`ctrlHicList`	hicList to use as control for a differential analysis.

submatrices	list of submatrices.
submatrix.name	name of submatrix to focus on. use names(submatrices). If submatrices is not NULL, this argument must not be NULL.
diffFun	The function used to compute differential between counts column of hicList and ctrlHicList. If the argument is a character, possible choices are: • "-", "substract" or "substraction" apply a substraction (Default) • "/" or "ratio" apply a ratio • "log2","log2-","log2/" or "log2ratio" apply a log2 on ratio • other apply a log2 on 1+ratio
which_chrom	a combination of chromsome names use names(hicList) to see the possible names.
which_range	a GRanges object or a string of type "chr1:1-100". see StrToGranges().

Value

funcions returns a data.frame with 3 columns: chrom, altchrom and counts. chrom being bin names of rows, altchrom being bin names of columns and counts being the counts. see also strawr::straw() & plotgardener::readHic()

Examples

```
data(HiC_Ctrl.cmx_lst)
data(HiC_HS.cmx_lst)
preparePlotgardener(
  hicList = HiC_HS.cmx_lst,
  ctrlHicList = HiC_Ctrl.cmx_lst,
  which_chrom = "2L_2L",
  diffFun = "substract")|> head()
```

QtlThreshold *Find threshold for outliers based on quantiles.*

Description

Find threshold for outliers triming based on quantiles.

Usage

```
QtlThreshold(x = NULL, prctThr = 5, tails = "both")
```

Arguments

x	: Numeric vector.
prctThr	: Percentage (0-100) threshold. (Default 5)
tails	: Bounds to return, "lower", "upper" or "both". (Default "both")

Details

QtlThreshold

Value

Numerical vector of thresholds values for outliers triming.

Examples

```
set.seed(1111)
x <- 0:100
x <- sort(x)
x
QtlThreshold(x, prctThr = 5, tails = "lower")
QtlThreshold(x, prctThr = 5, tails = "both")
QtlThreshold(x, prctThr = 5, tails = "upper")
```

ReduceRun *Apply a function over two RLE*

Description

Apply a function on the values over two RLE and return one RLE.

Usage

```
ReduceRun(firstRle, secondRle, reduceMethod = "paste", ...)
```

Arguments

firstRle	: First rle.
secondRle	>: Second rle.
reduceMethod	: Name of a function to apply e.g paste, sum, mean.
...	<...>: Other parameter for the reduce function.

Details

ReduceRun

Value

Reduced Rle

ResizeMatrix *Resize a matrix*

Description

Resize a numericam matrix in new dimension.

Usage

```
ResizeMatrix(mtx, newDim = dim(mtx))
```

Arguments

mtx	: A numerical matrix to resize.
newDim	: The number of rows and cols in resized matrix.

Details

ResizeMatrix

Value

Resized matrix.

SearchPairs *Creates pairs from genomic index.*

Description

Creates pairs of coordinates from indexed anchor and bait genomic coordinates according to distance constraints.

Usage

```
SearchPairs(
  indexAnchor = NULL,
  indexBait = NULL,
  minDist = NULL,
  maxDist = NULL,
  exclude_duplicates = TRUE,
  exclude_self_interactions = TRUE,
  verbose = FALSE,
  cores = 1
)
```

Arguments

indexAnchor	: A first indexed GRanges object used as pairs anchor (must be indexed using IndexFeatures()).
indexBait	: A second indexed GRanges object used as pairs bait (must be indexed using IndexFeatures()). If NULL, indexAnchor is used instead (Default NULL)
minDist	: Minimal distance between anchors and baits. (Default NULL)
maxDist	: Maximal distance between anchors and baits. (Default NULL)
exclude_duplicates	Should duplicated pairs ("2L:100_2L:150" & "2L:150_2L:100") be removed? (Default: TRUE)
exclude_self_interactions	Should pairs between the same bin ("2L:100_2L:100") be excluded? (Default: TRUE)
verbose	: Show the progression in console? (Default FALSE)
cores	: Number of cores to use. (Default 1)

Details

SearchPairs

Value

A GInteractions object.

Examples

```
# Data
data(Beaf32_Peaks.gnr)

# Index Beaf32
Beaf32_Index.gnr <- IndexFeatures(
    gRangeList = list(Beaf = Beaf32_Peaks.gnr),
    chromSizes = data.frame(seqnames = c("2L", "2R"),
        seqlengths = c(23513712, 25286936)),
    binSize = 100000
)

# Beaf32 <-> Beaf32 Pairing
Beaf_Beaf.gni <- SearchPairs(indexAnchor = Beaf32_Index.gnr)
```

SeqEnds *Get all sequences lengths.*

Description

Get all sequences lengths for each ranges of a GRanges object.

Usage

```
SeqEnds(gRanges)
```

Arguments

gRanges : A GRanges object.

Details

SeqEnds

Value

An integer vector.

Examples

```
GRange.grn <- GenomicRanges::GRanges(
    seqnames = S4Vectors::Rle(c("chr1", "chr2", "chr1"), c(1, 3, 1)),
    ranges = IRanges::IRanges(101:105, end = 111:115,
        names = letters[seq_len(5)]),
    strand = S4Vectors::Rle(BiocGenerics::strand(c("-", "+", "*", "+")),
        c(1, 1, 2, 1)),
    seqinfo = c(chr1 = 200, chr2 = 300),
    score = seq_len(5)
)
SeqEnds(GRange.grn)
```

StrToGRanges *Convert String to GRanges.*

Description

Convert ranges describe with string (i.e seqname:start-end:strand) in GRanges object.

Usage

```
StrToGRanges(stringRanges)
```

Arguments

`stringRanges` : Strings to convert on GRanges.

Details

StrToGRanges

Value

A GRanges object.

Examples

```
StrToGRanges("chr1:1-100:+")
StrToGRanges(c("chr1:1-100:+", "chr2:400-500:-", "chr1:10-50:*"))
```

`SwitchMatrix` *Change values of HiC map.*

Description

Change values in matrix with observed, balanced, observed/expected or expected values according to what are be done in hic.

Usage

```
SwitchMatrix(hicLst, matrixKind = c("obs", "norm", "o/e", "exp"))
```

Arguments

`hicLst` <ListContactMatrix>: The HiC maps list.

`matrixKind` : The kind of matrix you want.

Details

SwitchMatrix

Value

A ContactMatrix list.

Examples

```
# Data
data(HiC_Ctrl.cmx_lst)

# Preprocess HiC
HiC.cmx_lst <- HiC_Ctrl.cmx_lst |>
    BalanceHiC(
        interactionType = "cis",
        method = "ICE"
    ) |>
    OverExpectedHiC()

# Switch values in matrix
HiC_Ctrl_Obs.cmx_lst <- SwitchMatrix(HiC.cmx_lst, matrixKind = "obs")
HiC_Ctrl_Norm.cmx_lst <- SwitchMatrix(HiC.cmx_lst, matrixKind = "norm")
HiC_Ctrl_oe.cmx_lst <- SwitchMatrix(HiC.cmx_lst, matrixKind = "o/e")
HiC_Ctrl_exp.cmx_lst <- SwitchMatrix(HiC.cmx_lst, matrixKind = "exp")
```

TADs_Domains.gnr *D.melanogaster TADs.*

Description

Drosophila Melanogaster TADs on chromosome 2R and 2L (F.Ramirez, 2018)

Usage

```
data(TADs_Domains.gnr)
```

Format

An object of class GRanges.

Examples

```
data(TADs_Domains.gnr)
TADs_Domains.gnr
```

TrimOutliers *Remove outliers.*

Description

Replace values of a numerical vector that are below a minimal thresholds and/or above maximal thresholds.

Usage

```
TrimOutliers(x, thr = SdThreshold(x), clip = FALSE)
```

Arguments

x	: Numeric vector.
thr	: Numeric vector of length 2. first value is minimal threshold, second value maximal threshold (Default find threshold based on standarrd deviation. see SdThreshold function)
clip	: If TRUE the values out of bounds are replaced with thresholds values. If FALSE the Values out of bound are replaced with NA (Default FALSE).

Details

TrimOutliers

Value

Trimed Numerical vector.

TSS_Peaks.gnr *D.melanogaster Transcription starting sites.*

Description

Data from a CHip Seq experiment

Usage

```
data(TSS_Peaks.gnr)
```

Format

An object of class GRanges.

Examples

```
data(TSS_Peaks.gnr)
TSS_Peaks.gnr
```

VCnorm *Compute Vanilla Count Correction.*

Description

Compute Vanilla Count or Vanilla Count square root correction normalization on hic maps.

Usage

```
VCnorm(hic = NULL, qtlTh = 0.15, vcsqrt = TRUE)
```

Arguments

hic	: The HiC maps chunk to normalize.
qtlTh	: The threshold quantile below which the bins will be ignored. (Default 0.15)
vcsqrt	: Whether the square root should be applied. (Default TRUE)

Details

VCnorm

Value

A matrices list.

Examples

```
# Data
data(HiC_Ctrl.cmx_lst)

HiC_Ctrl_VC.cmx <- VCnorm(HiC_Ctrl.cmx_lst[["2L_2L"]])
HiC_Ctrl_VC_SQRT.cmx <- VCnorm(HiC_Ctrl.cmx_lst[["2L_2L"]], vcsqrt = TRUE)
```

`viridis` *viridis palette.*

Description

Create a viridis palette.

Usage

```
viridis(
  paletteLength = NULL,
  space = "rgb",
  interpolationMethod = "linear",
  bias = 1
)
```

Arguments

`paletteLength`	: Color number.
`space`	: A character string; interpolation in RGB or CIE Lab color spaces. See ?grDevices::colorRamp for more details. (Default "rgb")
`interpolationMethod`	: Use spline or linear interpolation. See ?grDevices::colorRamp for more details. (Default "linear")
`bias`	: A positive number. Higher values give more widely spaced colors at the high end. See ?grDevices::colorRamp for more details. (Default 1)

Details

viridis

Value

A vector of color.

Examples

```
viridis(9)
```

WrapFunction *Convert string to function.*

Description

Wrap a string into a function.

Usage

```
WrapFunction(...)
```

Arguments

...	: A string that could be parse and eval as a function.

Details

WrapFunction

Value

The result of the function or a function.

YlGnBu *YlGnBu palette.*

Description

Create a YlGnBu palette.

Usage

```
YlGnBu(
  paletteLength = NULL,
  space = "rgb",
  interpolationMethod = "linear",
  bias = 1
)
```

Arguments

paletteLength	: Color number.
space	: A character string; interpolation in RGB or CIE Lab color spaces. See ?grDevices::colorRamp for more details. (Default "rgb")
interpolationMethod	: Use spline or linear interpolation. See ?grDevices::colorRamp for more details. (Default "linear")
bias	: A positive number. Higher values give more widely spaced colors at the high end. See ?grDevices::colorRamp for more details. (Default 1)

Details

YlGnBu

Value

A vector of color.

Examples

```
YlGnBu(9)
```

YlOrRd *YlOrRd palette.*

Description

Create a YlOrRd palette.

Usage

```
YlOrRd(
  paletteLength = NULL,
  space = "rgb",
  interpolationMethod = "linear",
  bias = 1
)
```

Arguments

paletteLength	: Color number.
space	: A character string; interpolation in RGB or CIE Lab color spaces. See ?grDevices::colorRamp for more details. (Default "rgb")
interpolationMethod	: Use spline or linear interpolation. See ?grDevices::colorRamp for more details. (Default "linear")

`bias` : A positive number. Higher values give more widely spaced colors at the high end. See `?grDevices::colorRamp` for more details. (Default 1)

Details

YlOrRd

Value

A vector of color.

Examples

```
YlOrRd(9)
```

Index