Package ‘OGRE’

May 30, 2024

Type Package
Title Calculate, visualize and analyse overlap between genomic regions
Version 1.8.0
Description OGRE calculates overlap between user defined genomic region datasets. Any regions can be supplied i.e. genes, SNPs, or reads from sequencing experiments. Key numbers help analyse the extend of overlaps which can also be visualized at a genomic level.
License Artistic-2.0
Encoding UTF-8
LazyData false
Roxygen list(markdown = TRUE)
VignetteBuilder knitr
RoxygenNote 7.2.2
Imports GenomicRanges, methods, data.table, assertthat, ggplot2, Gviz, IRanges, AnnotationHub, grDevices, stats, GenomeInfoDb, shiny, shinyFiles, DT, rtracklayer, shinydashboard, shinyBS, tidyr
Depends R (>= 4.2.0), S4Vectors
Suggests testthat (>= 3.0.0), knitr (>= 1.36), rmarkdown (>= 2.11)
biocViews Software, WorkflowStep, BiologicalQuestion, Annotation, Metagenomics, Visualization, Sequencing
BugReports https://github.com/svenbioinf/OGRE/issues
URL https://github.com/svenbioinf/OGRE/
Config/testthat/edition 3
git_url https://git.bioconductor.org/packages/OGRE
git_branch RELEASE_3_19
git_last_commit 4e17fad
git_last_commit_date 2024-04-30
Repository Bioconductor 3.19
Date/Publication 2024-05-29
OGRE-package

Description

OGRE calculates overlap between user defined annotated genomic region datasets. Any regions can be supplied such as public annotations (genes), genetic variation (SNPs, mutations), regulatory elements (TFBS, promoters, CpG islands) and basically all types of NGS output from sequencing experiments. After overlap calculation, key numbers help analyse the extend of overlaps which can also be visualized at a genomic level.
**addDataSetFromHub**

**Details**

The main functions are:

- **OGREDataSetFromDir** - build an OGRE dataset from a user defined directory with GRanges annotation files.
  - **loadAnnotations** - Load dataset files containing genomic regions annotation information from hard drive

- **OGREDataSet** - build an empty OGRE dataset to flexibly add datasets from other sources like AnnotationHub or custom GRanges objects.
  - **addDataSetFromHub** - adds datasets from AnnotationHub
  - **addGRanges** - adds user defined GenomicRanges datasets

**fOverlaps** - Finds all overlaps between query and subject datasets

**sumPlot** - calculates key numbers, tables and plots

**gvizPlot** - generates a genomic plot around query elements with overlapping subject hits.

For additional information, see the package vignette, by typing **vignette("OGRE")**. Software-related questions or issues can be posted to the Bioconductor Support Site:

[https://support.bioconductor.org](https://support.bioconductor.org)

or on github:


**Author(s)**

Sven Berres, Jörg Gromoll, Marius Wöste, Sarah Sandmann, Sandra Laurentino

---

**addDataSetFromHub** *Add dataSet from AnnotationHub*

**Description**

AnnotationHub offers a wide range of annotated datasets which can be manually acquired but need some parsing to work with OGRE as detailed in vignette section "Load datasets from AnnotationHub". For convenience **addDataSetFromHub()** adds one of the predefined human dataSets of **listPredefinedDataSets()** to a OGREDataSets. Those are taken from AnnotationHub and are ready to use for OGRE. Additional information on dataSets can be found here **listPredefinedDataSets**.

**Usage**

```r
addDataSetFromHub(OGREDataSet, dataSet, type)
```
addGRanges

Arguments

OGREDataSet OGREDataSet
dataSet character Name of one predefined dataSets to add as query or subject to a
OGREDataSet. Possible dataSets can be show with listPredefinedDatasets().
type Type of dataSet, must be either query or subject. If query the dataSet will be
added as query and at the first position of OGREDataSet.

Value

OGREDataSet.

Examples

myOGRE <- OGREDataSet()
myOGRE <- addDataSetFromHub(myOGRE,"protCodingGenes","query")

addGRanges Add GenomicRanges

Description

Add a GenomicRanges dataset to OGREDataSet

Usage

addGRanges(OGREDataSet, dataSet, type, label = NULL)

Arguments

OGREDataSet An OGREDataSet
dataSet A GRanges object. Each region needs chromosome, start, end and strand information. A unique ID and a name column must be present in the GenomicRanges object metadata. Avoid different chromosome naming conventions i.e. (chr1, CHR1, 1, I) among all datasets
type Type of dataSet, must be either query or subject. If query the dataSet will be
added as query and at the first position of OGREDataSet.
label A character that will label your GRanges object. If not supplied, the label will be
guessed from the dataset parameter.

Value

OGREDataSet.

Examples

myOGRE <- OGREDataSet()
myGRanges <- makeExampleGRanges()
myOGRE <- addGRanges(myOGRE,myGRanges,"query")
covPlot

Coverage plot

Description
Generates coverage plots of all subject datasets and stores them as a list, that can be accessed by 
metadata(OGREDataSet)$covPlot

Usage
covPlot(
  OGREDataSet,
  datasets = names(OGREDataSet)[seq(2, length(OGREDataSet))],
  nbin = 100
)

Arguments
OGREDataSet An OGREDataSet
datasets character vector of subject dataset names. Default: Generates a coverage plots for all subjects
nbin Number of bins

Value
OGREDataSet.

Examples
myOGRE <- makeExampleOGREDataSet()
myOGRE <- loadAnnotations(myOGRE)
myOGRE <- fOverlaps(myOGRE)
myOGRE <- covPlot(myOGRE)
metadata(myOGRE)$covPlot

extendGRanges Extend a GRanges object

Description
Extend(shrink) ranges of a GRanges object.

Usage
extendGRanges(OGREDataSet, name, upstream = 0, downstream = 0)
extractPromoters

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OGREDataSet</td>
<td>An OGREDataSet</td>
</tr>
<tr>
<td>name</td>
<td>character Name of the GRanges object for extending</td>
</tr>
<tr>
<td>upstream</td>
<td>int (positive or negative number)</td>
</tr>
<tr>
<td>downstream</td>
<td>int (positive or negative number)</td>
</tr>
</tbody>
</table>

Value

OGREDataSet

Examples

```r
myOGRE <- makeExampleOGREDataSet()
myOGRE <- loadAnnotations(myOGRE)
#extend range by shifting start 100 bp in upstream direction
myOGRE <- extendGRanges(myOGRE,"genes",upstream=100)
#shrinking range by shifting end 100 bp in upstream direction
myOGRE <- extendGRanges(myOGRE,"genes",downstream=-100)
#shrinking range by shifting from both sides to the center
myOGRE <- extendGRanges(myOGRE,"genes",upstream=-10,downstream=-10)
```

Description

A wrapper of GenomicRanges::promoters() to extract promoter regions of a GRanges object stored in a OGREDataSet

Usage

```r
extractPromoters(OGREDataSet, name, upstream = 2000, downstream = 200)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OGREDataSet</td>
<td>An OGREDataSet</td>
</tr>
<tr>
<td>name</td>
<td>character Name of the GRanges object</td>
</tr>
<tr>
<td>upstream</td>
<td>int (positive) upstream=2000(default)</td>
</tr>
<tr>
<td>downstream</td>
<td>int (positive) downstream=200(default)</td>
</tr>
</tbody>
</table>

Value

OGREDataSet
fOverlaps

Examples

```r
myOGRE <- makeExampleOGREDataSet()
myOGRE <- loadAnnotations(myOGRE)
myOGRE <- extractPromoters(myOGRE, "genes", upstream=2000, downstream=200)
```

---

**Description**

Finds all overlaps between query and subject(s) and stores each hit (overlap) in data table detailDT. Data table sumDT shows all overlaps of a certain subject type for all query elements. By default also partially overlaps are reported. Overlap calculation is done using GenomicRanges::findOverlaps() implementation.

**Usage**

```r
fOverlaps(OGREDataSet, selfHits = FALSE, ignoreStrand = TRUE, ...)
```

**Arguments**

- `OGREDataSet`: A OGREDataSet.
- `selfHits`: logical if FALSE (default) ignores self hits of identical regions (with identical IDs) within datasets.
- `ignoreStrand`: logical If TRUE (default) two regions with overlapping locations on different strands are considered an overlap hit.
- `...`: Additional parameters, see GenomicRanges::findOverlaps()

**Value**

OGREDataSet.

**Examples**

```r
myOGRE <- makeExampleOGREDataSet()
myOGRE <- loadAnnotations(myOGRE)
myOGRE <- fOverlaps(myOGRE)
```
gvizPlot  Generate Gviz plot

Description

gvizPlot generates a plot around one or many given query elements with all overlapping subject hits. In addition, each generated plot can be stored in the `gvizPlots` folder get or set by `gvizPlotsFolder`. A maximum of 25 elements can be plotted per track.

Usage

```r
gvizPlot(
    OGREDataSet,
    query,
    gvizPlotsFolder = metadata(OGREDataSet)$gvizPlotsFolder,
    trackRegionLabels = setNames(rep("ID", length(OGREDataSet)), names(OGREDataSet)),
    trackShapes = setNames(rep("fixedArrow", length(OGREDataSet)), names(OGREDataSet)),
    showPlot = FALSE,
    extendPlot = c(-300, 300),
    nElements = 25
)
```

Arguments

**OGREDataSet**  A OGREDataSet.

**query**  A character vector of one or many query elements ID’s (i.e. Gene ID’s).

**gvizPlotsFolder**  A character pointing to the plot(s) output directory. If not supplied a folder is automatically generated and can be accessed by `metadata(OGREDataSet)$gvizPlotsFolder`.

**trackRegionLabels**  A labeled character vector that defines the type of label that is displayed for query and subject elements during plotting. Vector values represent the type of label and vector labels define the type of subject element. In the following example `setNames(c("ID","name"),c("genes","CGI"))` Value "ID" and label "genes" would annotate your genes with IDs taken from the ID column of your dataset. Datasets not defined in this vector are plotted without track labels.

**trackShapes**  A labeled character vector that defines the type of shape in which every dataset’s elements are displayed. Vector values represent the type of shape and vector labels define the type of subject element. In the following example `setNames(c("fixedArrow","box"),c("genes","CGI"))` Value "fixedArrow" and label "genes" would display your genes in fixedArrow and CGI as box shape. Possible values: (box, arrow, fixedArrow, ellipse, and smallArrow) Default="fixedArrow"

**showPlot**  logical If FALSE(default) plots are only saved to `gvizPlotsFolder`. If TRUE plots are additionally send to the plotting window.
extendPlot  int vector Integer vector of length two that extends the plot window to the left or right by adding the first value to query start and the second value to query end coordinates(bp). e.g. c(-1000,1000) zooms out, c(1000,-1000) zooms in and c(-1000,0) shifts the plot window to the left.

nElements integer Number of elements that are displayed in each track (Default=25). High n.elements can lead to overplotting. Use nElements=FALSE to display all elements.

Value

OGREDataSet.

Examples

myOGRE <- makeExampleOGREDataSet()
myOGRE <- loadAnnotations(myOGRE)
myOGRE <- fOverlaps(myOGRE)
myOGRE <- gvizPlot(myOGRE,query="ENSG00000142168")

listPredefinedDataSets

List predefined datasets

Description

Use listPredefinedDataSets() to receive a vector of names for predefined datasets that can be acquired from AnnotationHub that are already correctly parsed and formatted. Each of the listed names can be used as input for addDataSetFromHub(). Currently supported:

- protCodingGenes - Protein coding genes from HG19 (GRCh37) Ensembl For additional information use: getInfoOnIds(AnnotationHub(), "AH10684")
- CGI - CpG islands from HG19 UCSC For additional information use: getInfoOnIds(AnnotationHub(), "AH5086")
- SNP - Common Single Nucleotide Polymorphism from HG19 UCSC For additional information use: getInfoOnIds(AnnotationHub(), "AH5105")
- TFBS - Transcription Factor Binding Sites conserved from HG19 UCSC For additional information use: getInfoOnIds(AnnotationHub(), "AH5090")
- Promoters - Promoter and flanking regions from HG19 Ensembl (Note: This annotation is currently not included in AnnotationHub and is therefore downloaded from Ensembl’s ftp site)

Usage

listPredefinedDataSets()
Value

character vector.

Examples

listPredefinedDataSets()
**makeExampleGRanges**  
*Make an example GRanges dataset*

**Description**

makeExampleGRanges generates an example GRanges dataset.

**Usage**

```r
makeExampleGRanges()
```

**Value**

OGREDataSet.

**Examples**

```r
myGRanges <- makeExampleGRanges()
```

---

**makeExampleOGREDataSet**  
*Make a example OGRE dataset*

**Description**

makeExampleOGREDataSet generates an example OGREDataSet from dataset files stored in OGRE’s extdata directory.

**Usage**

```r
makeExampleOGREDataSet()
```

**Value**

OGREDataSet.

**Examples**

```r
myOGRE <- makeExampleOGREDataSet()
```
Description

Builds a OGREDataset as a GenomicRangesList for storing and analysing datasets which can be added by addDataSetFromHub() or addGRanges(). Use BuildOGREDataSetFromDir for adding dataSets stored as files.

Usage

OGREDataSet()

Value

A OGREDataset.

Examples

myOGRE <- OGREDataSet()

Description

Builds a OGREDataset from user specified directories containing datasets for which an overlap between query and subject is to be calculated. A OGREDataset is a GenomicRangesList which stores datasets in a list like structure and possible metadata information.

Usage

OGREDataSetFromDir(queryFolder, subjectFolder)

Arguments

queryFolder A character path pointing to the directory where your query dataset is located.
subjectFolder A character path pointing to the directory where your subject dataset(s) are located.

Value

A OGREDataset.
Examples

```r
myQueryFolder <- file.path(system.file('extdata', package = 'OGRE'), "query")
mySubjectFolder <- file.path(system.file('extdata', package = 'OGRE'), "subject")
myOGRE <- OGREDataSetFromDir(queryFolder=myQueryFolder, subjectFolder=mySubjectFolder)
```

Description

Plots overlap histograms of all subject datasets and stores them as a list, that can be accessed by `metadata(myOGRE)$hist`.

Usage

```r
plotHist(OGREDataSet, plot0 = FALSE)
```

Arguments

- `OGREDataSet`: An OGREDataSet
- `plot0`: plot0=FALSE (default) plots a histogram of all dataset elements with overlaps, excluding elements without overlaps. plot0=FALSE also includes elements without overlaps.

Value

OGREDataSet.

Examples

```r
myOGRE <- makeExampleOGREDataSet()
myOGRE <- loadAnnotations(myOGRE)
myOGRE <- fOverlaps(myOGRE)
myOGRE <- plotHist(myOGRE)
metadata(myOGRE)$hist
```
**readDataSetFromFolder**  
*Read dataset(s) from folder*

**Description**

`readDataSetFromFolder()` scans `queryFolder` and `subjectFolder` for either .RDS/.rds or .CSV/.csv files and adds them to a OGREDataSet. Each region needs chromosome, start, end and strand information. (tabular file columns must be named accordingly). A unique ID and a name column must be present in the GenomicRanges object’s metadata and tabular file.

**Usage**

```r
readDataSetFromFolder(OGREDataSet, type)
```

**Arguments**

- **OGREDataSet** A OGREDataSet.
- **type** character and one of query/subject.

**Value**

A OGREDataSet.

**Examples**

```r
myOGRE <- makeExampleOGREDataSet()
myOGRE <- readDataSetFromFolder(myOGRE,type="query")
myOGRE <- readDataSetFromFolder(myOGRE,type="subject")
```

**readQuery**  
*Read query dataset*

**Description**

`readQuery()` scans `queryFolder` for a GRanges object stored as .RDS/.rds or .gff .GFF file and attaches it to the OGREDataSet.

**Usage**

```r
readQuery(OGREDataSet)
```

**Arguments**

- **OGREDataSet** A OGREDataSet.

**Value**

A OGREDataSet.
**readSubject**

Read subject datasets

**Description**

readSubject() scans SubjectFolder for GRanges objects stored as .RDS/.rds or .gff .GFF files and attaches them to the OGREDataSet.

**Usage**

readSubject(OGREDataSet)

**Arguments**

OGREDataSet A OGREDataSet.

**Value**

A OGREDataSet.

---

**SHREC**

SHREC SHiny interface for REgion Comparison

**Description**

SHREC() is a graphical user interface for OGRE

**Usage**

SHREC()

**Value**

Runs GUI, this function normally does not return
subsetGRanges  
*Subset a GRanges object*

**Description**

Subsets a GRanges object with reference to it's ID column using a ID vector.

**Usage**

```r
subsetGRanges(OGREDataSet, IDs, name)
```

**Arguments**

- **OGREDataSet**: An OGREDataSet
- **IDs**: character vector with IDs used to subset the GRanges object defined in name
- **name**: character Name of the GRanges object for subsetting. One of the GRanges objects in a OGREDataSet

**Value**

OGREDataSet.

**Examples**

```r
myOGRE <- makeExampleOGREDataSet()
myOGRE <- loadAnnotations(myOGRE)
myOGRE <- subsetGRanges(myOGRE, c("ENSG00000142168", "ENSG00000256715"), "genes")
```

summarizeOverlap  
*Calculates min/max/average overlap*

**Description**

Calculates min/max/average overlap for all datasets using `summary()`. Results can be accessed by `metadata(OGREDataSet)$summaryDT` which is a `list()` of two `data.table` objects. The first one includes elements without any overlap at all and the second provides summary numbers for all elements that have at least one overlap.

**Usage**

```r
summarizeOverlap(OGREDataSet)
```

**Arguments**

- **OGREDataSet**: An OGREDataSet
**sumPlot**

**Value**

OGREDataSet.

**Examples**

```r
myOGRE <- makeExampleOGREDataSet()
myOGRE <- loadAnnotations(myOGRE)
myOGRE <- fOverlaps(myOGRE)
myOGRE <- summarizeOverlap(myOGRE)
metadata(myOGRE)$summaryOT
```

---

**sumPlot**  
*Generate summary plot*

**Description**

`sumPlot()` calculates key numbers i.e. (total number of overlaps, number of overlaps per subject...) to help with an exploratory data evaluation and displays them in an informative barplot.

**Usage**

```r
sumPlot(OGREDataSet)
```

**Arguments**

- **OGREDataSet**  
  A OGREDataSet.

**Value**

OGREDataSet.

**Examples**

```r
myOGRE <- makeExampleOGREDataSet()
myOGRE <- loadAnnotations(myOGRE)
myOGRE <- fOverlaps(myOGRE)
myOGRE <- sumPlot(myOGRE)
```
Index

* internal
  readQuery, 14
  readSubject, 15
* package
  OGRE-package, 2
  addDataSetFromHub, 3
  addGRanges, 3, 4
  covPlot, 5
  extendGRanges, 5
  extractPromoters, 6
  fOverlaps, 3, 7
  gvizPlot, 3, 8
  listPredefinedDataSets, 3, 9
  loadAnnotations, 3, 10
  makeExampleGRanges, 11
  makeExampleOGREDataSet, 11
  OGRE-package, 2
  OGREDataSet, 3
  OGREDataSetFromDir, 3, 12
  plotHist, 13
  readDataSetFromFolder, 14
  readDataSetFromFolder(), 14
  readQuery, 14
  readQuery(), 14
  readSubject, 15
  readSubject(), 15
  SHREC, 15
  subsetGRanges, 16
  summarizeOverlap, 16
  sumPlot, 3, 17