Package 'epivizr'

October 7, 2014

Maintainer Hector Corrada Bravo <hcorrada@gmail.com>

Author Hector Corrada Bravo, Florin Chelaru, Llewellyn Smith, Naomi Goldstein

Version 1.2.12

License GPL-3

Title R Interface to epiviz web app

Description This package provides Websocket communication to the
epiviz web app (http://epiviz.cbcb.umd.edu) for interactive
visualization of genomic data. Objects in R/bioc interactive
sessions can be displayed in genome browser tracks or plots to
be explored by navigation through genomic regions. Fundamental
Bioconductor data structures are supported (e.g., GenomicRanges
and SummarizedExperiment objects), while providing an easy
mechanism to support other data structures. Visualizations
(using d3.js) can be easily added to the web app as well.

VignetteBuilder knitr

Depends R (>= 3.0.1), methods, Biobase, GenomicRanges (>= 1.13.47)

Imports httpuv (>= 1.3.0), rjson, IRanges

Suggests testthat, roxygen2, knitr, antiProfilesData, hgu133plus2.db,knitrBootstrap

Collate 'utils.r' 'EpivizServer-class.R' 'EpivizDeviceMgr-class.R'
'startEpiviz.R' 'EpivizData-class.R' 'EpivizTrackData-class.R'
'EpivizFeatureData-class.R' 'EpivizBlockData-class.R'
'EpivizBpData-class.R' 'register-methods.R' 'EpivizChart-class.R' 'EpivizDevice-class.R' 'zzz.R'

biocViews Visualization, Infrastructure, GUI

R topics documented:

EpivizBlockData-class   *'EpivizBlockData' objects*

Description

Data displayed only as genomic regions (no quantitative data)

Details

This class extends [`EpivizTrackData-class`](#). Like its super-class, the data containing object is constrained to be of class [`GIntervalTree`](#). Its columns field is constrained to be NULL.

Methods

plot(): Calls the blockChart method of [`EpivizDeviceMgr-class`](#).

getMeasurements(): The measurements names and ids for data encapsulated by this object. Returns a named character vector of length 1 containing name with id as the vector name.

parseMeasurement(msId=NULL): Checks msId==id.

packageData(msId=NULL): Returns list with slots start and end for ranges overlapping curQuery.

Author(s)

Hector Corrada Bravo

EpivizBlockDataPack-class

'EpivizBlockDataPack' objects

Description

Package of block data for transfer to web app.

Fields

data: (list) data is kept here for packaging. Slot names correspond to measurement ids.

Author(s)

Hector Corrada Bravo

EpivizBpData-class *'EpivizBpData' objects*

Description

Base-pair resolution quantitative data.

Details

This class extends EpivizTrackData-class directly. As such, its object field is constrained to contain a GIntervalTree object. Quantitative data for each genomic position is assumed to be stored in the elementMetadata slot of object.

Methods

.checkColumns(columns): Checks argument columns is a subset of names(mcols(object)).

.getColumns(): Returns names(mcols(object)).

plot(): Calls the lineChart method of EpivizDeviceMgr-class with columns as argument.

getMeasurements(): Returns named character vector. Entries are of format '<name>$<column>'. Vector names are of format '<id>__<column>'.

parseMeasurement(msId): Parses measurement of format '<id>__<column>'.

packageData(msId): Get data for msId for rows overlapping curQuery. Returns a list with components: min: min value for data range, max: max value for data range, data: list with slots: bp: genomic location, value: numeric data value.

Author(s)

Hector Corrada Bravo

EpivizBpDataPack-class

'EpivizBpDataPack' objects

Description

Package of base-pair level data for transfer to web app.

Fields

min: (numeric) minimum of data range.

max: (numeric) maximum of data range.

data: (list) data entry for each measurement.

Author(s)

Hector Corrada Bravo

EpivizChart-class   *'EpivizChart' objects*

Description

Encapsulate information about charts added to web app.

Fields

measurements: Named character vector of measurements displayed in chart. Entries are measurement names, vector names are measurement ids.

id: (character) id assigned by session manager to chart. Use getId method to get id.

mgr: ([EpivizDeviceMgr](#)) session manager object.

inDevice: (logical) TRUE if chart is defined within an [EpivizDevice-class](#) object.

type: (character) chart type.

Methods

setId(id):set id. Internal use only.

getId():get id.

setInDevice(x):set inDevice field. Internal use only.

Author(s)

Hector Corrada Bravo

EpivizData-class *'EpivizData' objects*

Description

This class serves as the basis for the epiviz data container class hierarchy.

Details

Objects of classes derived from this class define measurements in the measurement/chart design behind epiviz. Subclasses extended from this class correspond to specific data types, and are responsible for packaging data for display in the epiviz web app. In principle, each subclass corresponds to a data type in the epiviz JS framework. Use the listTypes method of EpivizDeviceMgr-class to see the R-JS class correspondence. This virtual class implements common functionality for subclass objects. Objects in this class hierarchy are created using link{register-methods}.

Fields

object: The data containing object

mgr: The epiviz session manager (object of class EpivizDeviceMgr-class)

inDevice: (logical) TRUE if object defined in a EpivizDevice-class object.

id: (character) the measurement object id determined by the session manager. Use getId method to get this field.

name: (character) the measurement object name. Measurements declared by the object are given names derived from this field. Use getName method to get this field.

columns: (character vector) the names of the columns containing measurements encapsulated by this object.

ylim: (numeric matrix) data range for measurements encapsulated by this object. These are passed to the web app to determine plot ranges.

curQuery: (GRanges object) last query made to this object.

curHits: (integer) indexes returned by last query to this object.

Methods

Below, methods tagged as 'VIRTUAL' must be implemented by subclasses.

.checkColumns(columns): Check if argument columns is valid for data object. 'VIRUTAL'.

.getColumns(): Infer a valid set of columns from data object. 'VIRTUAL'.

.checkLimits(ylim): Check if argument ylim is valid for data object. 'VIRTUAL'.

.getLimits(): Infer a valid set of ylim from data object. 'VIRTUAL'.

update(newObject, sendRequest=TRUE): updates data object to argument newObject. This method checks that the classes of newObject and the current object are the same, that the columns field is valid for newObject and updates the ylim field using the .getLimits method. If sendRequest==TRUE the web app is refreshed to reflect the data in newObject.

getId(): Get the object's id.

setId(id): Set the object's id. For internal use only.

getName(): Get the object's name.

setName(name): Set the object's name. For internal use only.

setLimits(ylim): Set the object's ylim field. Argument is checked using the .checkLimits method.

getMeasurements(): Get the object's encapsulated measurements in format appropriate for web app. VIRTUAL, for internal use only.

parseMeasurement(msId=NULL): Parse measurement id msId to extract object id and column. VIRTUAL, for internal use only.

setMgr(): Set the associated session manager. For internal use only.

setInDevice(x): Set the inDevice field to x. For internal use only.

plot(): Plot data in this object. VIRTUAL.

packageData(msId): Package data for measurement msId in appropriate format for web app. VIRTUAL.

getData(query, msId=NULL): Get data overlapping query. Subclasses should not need to implement this method. It calls the packageData method which performs class-specific data packaging.

Author(s)

Hector Corrada Bravo

EpivizDataPack-class   *'EpivizDataPack' objects*

Description

Package data to send to web app

Details

Objects of this class package the result of packageData for [EpivizData-class](#) for multiple measurements and prepares data for transfer to the epiviz web app. This is a virtual class.

Fields

length: Number of measurements packaged.

Methods

set(curData, msId, index): Set curData as the entry on index index for measurement with msId. This is a VIRTUAL method.

getData(): Return object for transfer to web app from packaged data. This is a VIRTUAL method.

Author(s)

Hector Corrada Bravo

EpivizDevice-class   *'EpivizDevice' objects*

Description

Encapsulate both measurement and chart for display in epiviz web app

Details

This provides a convenience interface to the measurement/chart interface. All measurements for a given object are encapsulated in an EpivizData-class object, and its `plot` method is called to create an EpivizChart-class object.

Fields

`msObject`: (EpivizData-class) object encapsulating measuremnts.

`chartObject`: (EpivizChart-class) object encapsulating a chart.

`id`: (character) device id set by session manager. Use `getId` method to get id.

Methods

`getMsObject()`: Return `msObject`.

`getChartObject()`: Return `chartObject`.

`getMsId()`: Return id of `msObject`.

`getChartId()`: Return id of `chartObject`.

`setId(id)`: Set device id. For internal use only.

`getId()`: Return device object id.

`update(newObject, sendRequest=TRUE)`: Update data object in `msObject`. Calls the `update` method of `msObject`.

Author(s)

Hector Corrada Bravo

EpivizDeviceMgr-class   *'EpivizDeviceMgr' objects*

Description

This class manages interactive visualization connections to the epiviz web app. It provides a mechanism to add measurements (from objects in the R/bioc workspace), charts (plots and tracks in the web app) or devices (a one-step combination of measurements and charts). It also provides navigation and other interactive functions. Setters and getters are defined for the pertinent fields, which should not be accessed directly. Objects of this class are usually created using the startEpiviz function.

Details

The most important aspect of interactive connections to the epiviz web app is the ability to add tracks and plots displaying data from an R/bioc session. This class is designed so that data and display are distinct, the former represented by measurements encapsulated in EpivizData-class objects, the latter by charts encapsulated by EpivizChart-class objects.

For a user to visualize data stored in an object that can be referenced by genomic location, e.g., a GRanges or SummarizedExperiment objects, they must first indicate to the session manager which measurements are provided by this object using the addMeasurements method. This creates an object derived from class EpivizData-class (using register-methods) which wraps the data object so that efficient overlap queries are performed (using GIntervalTree objects) and adds the set of measurements provided to the web app. This allows users to create charts displaying data from this object either by using the addChart method or the web app UI itself. See section 'Measurement management methods' below for more information.

Once measurements have been declared, users can use the addChart method to visualize them in one or more different plot types (see section 'Chart types' below). Objects from classes derived from EpivizData-class have plot methods that use reasonable default chart types. See section 'Chart management methods' below for more information.

We also provide a 'device' interface to simplify the steps above. Users can call the addDevice method, passing a data object to be visualized and a chart is added with the default visualization for the object. The same mechanism above is used, so both a EpivizData-class and EpivizChart-class objects are added to the session manager. See section 'Device management methods' below for more information.

Fields

url: The url used to connect to the epiviz web app

msList: List of added measurements (objects of classes that inherit from EpivizData-class)

typeMap: List of currently available measurements types

msIdCounter: Used internally to manage msList

chartList: List of currently added charts (objects of class EpivizChart-class)

chartIdCounter: Used internally to manage chartList

activeId: Id of currently active chart (not used)

chartIdMap: Character vector of chart ids assigned by the web app. Vector names are chart ids set by the session manager

deviceList: List of currently added devices (objects of class EpivizDevice-class)

deviceIdCounts: Used internally to manage deviceList

server: Object of class EpivizServer-class encapsulating the websocket connection to app

verbose: Log to console on requests received

nonInteractive: Run in non-interactive mode, for internal development use only

callbackArray: Object of class IndexedArray-class containing request callback functions

Utility methods

bindToServer(): Call the bindManager function of the server object passing this object as argument.

isClosed(): Returns TRUE if connection to app is closed.

openBrowser(url=NULL): Open browser for epiviz web app, if argument url is NULL (the default), the url field is used.

service(): Service requests from app (this call blocks the R/interactive session until loop is escaped)

stopService(): Stop the service loop

startServer(): Start the websocket connection server

stopServer(): Stops the websocket connection server. Also removes all currently added devices, charts and measurements from web app.

Navigation methods

refresh(): NOT SUPPORTED YET

navigate(chr, start, end): Navigate in web app to give genomic region

slideshow(granges, n=length(granges)): Navigate in web app to the first n regions in GenomicRanges-class object granges in succession.

Measurement management methods

addMeasurements(obj, msName, sendRequest=TRUE, ...): Add measurements in obj, calls the register-methods on obj passing extra arguments in .... Measurements are given names derived from msName. Measurements are added to the web app itself if sendRequest==TRUE (the default). This method adds a measuremnt object of class EpivizData-class, returned by the register-methods to the msList field. Measurement ids are generated automatically using msIdCounter and are returned by this method.

.findMeasurements(msType, ms): Finds the index of objects in msList where measurements ms of type msType are defined. See below to see valid types. Argument ms is a character vector of measurement ids. This method is for internal use only.

.checkMeasurements(msType, ms, sendRequest=TRUE, ...): Checks that measurements in ms, of type msType are valid. If sendRequest==TRUE it also checks measurements are properly added to web app. This method is for internal use only.

.clearChartCaches(msObj, sendRequest=TRUE): Clears chart data caches in web app using any of the measurements defined by msObj of class ([EpivizData-class](#)). Only affects web app if sendRequest==TRUE. This method is for internal use only.

updateMeasurements(oldObject, newObject, sendRequest=TRUE): Update the data object wrapped in [EpivizData-class](#) object oldObject to newObject. The class of newObject depends on the type of data, but is checked for correctness, i.e., the the classes of the new and current data objects are the same. Argument oldObject can be a [EpivizData-class](#) object already added to the session, or a character containing the id of an object. Ids are returned by the addMeasurements method.

.getMsObject(msObjId): Get the [EpivizData-class](#) object corresponding the the id msObjId. This method is for internal use only.

rmMeasurements(msObj): Remove measurements stored in [EpivizData-class](#) object msObj. The msObj argument can be an [EpivizData-class](#) object added to the session manager, or the id of such an object.

rmAllMeasurements(): Remove all measurements added to the session manager.

listMeasurements(): List measurements added to the session manager.

getMeasurements(): Returns list of currently added measurements in format required by web app. This method is for internal use only.

getMeasurementType(x): Returns type of measurement object x of class [EpivizData-class](#).

Chart management methods

addChart(chartObject, sendRequest=TRUE, ...): Adds chart specified by chartObject of class [EpivizChart-class](#) to the session manager. The chart is added to the web app if sendRequest==TRUE (the default). The chartObject is inserted to the chartList field. Chart ids are generated automatically using chartIdCounter and are returned by this method.

.getChartObject(chartId): Returns the [EpivizChart-class](#) object corresponding to id chartId.

rmChart(chartObj): Remove chart corresponding to chartObj. Argument chartObj can be an object of class [EpivizChart-class](#) or the character id of such an object.

rmAllCharts(): Remove all charts currently in the session manager.

listCharts(): List all charts currently added to session manager.

setActive(): NOT USED

Chart types

blockChart(ms, ...): Used to display data in [GenomicRanges](#) objects as genomic regions using rectangles in a browser track. ms is a list of lists, each component describing a measurement provided by an object of class [EpivizBlockData-class](#). They can be obtained using the getMeasurements method in the [EpivizBlockData-class](#). plot method for [EpivizBlockData-class](#) calls this method.

lineChart(ms, ...): Used to display continuous data at base-pair level as a line plot in a browser track. ms is a list of lists, each component describing a measurement provided by objects of class [EpivizBpData-class](#). They can be obtained using the getMeasurements method in the [EpivizBlockData-class](#). The plot method for [EpivizBpData-class](#) calls this method.

scatterChart(x, y, ...): Used to display genomic feature data in SummarizedExperiment objects as a scatter plot. x and y are lists for measurements provided by objects of class EpivizFeatureData-class. They can be obtained using the getMeasurements method in the EpivizFeatureData-class. The plot method for EpivizFeatureData-class calls this method.

heatmapChart(ms, ...): Used to display genomic feature data in SummarizedExperiment objects as a heatmap. ms is a list of lists, each component describing a measurement provided by an object of class EpivizFeatureData-class. They can be obtained using the getMeasurements method in the EpivizFeatureData-class.

Device management methods

addDevice(obj, devName, sendRequest=TRUE, ...): Adds device for object obj. This method calls the addMeasurements method on obj and calls the plot method of the resulting EpivizData-class object. Measurements and charts added by this called are given names derived from devName. Arguemnts to addMeasurement and the plot-methods call can be given in .... Device ids are generated automatically using devIdCounter and are returned by this function. An object of class EpivizDevice-class is created and inserted into devList.

rmDevice(devObj): Remove the device corresponding to devObj: an object of class EpivizDevice-class or the id for such an object. This removes the corresponding chart and measurement objects from the session manager.

rmAllDevice(): Remove all devices added to the session manager.

updateDevice(oldObject, newObject, sendRequest=TRUE, ...): Update the data object wrapped in EpivizDevice-class object oldObject to newObject. The class of newObject depends on the type of data, but is checked for correctness, i.e., the the classes of the new and current data objects are the same. Argument oldObject can be a EpivizDevice-class object already added to the session, or a character containing the id of an object. Ids are returned by the addDevice method.

listDevices(): List all devices added to the session manager.

Data fetch methods

.initPack(msType, length=0L): Initialize a EpivizDataPack-class object of given length.

getData(measurements, chr, start, end): Query measurements for overlap with given genomic position. measurements is assumed to be a named list, with each component's name indicating measurement type as stored in msList. Each component in the list is a character vector of measurement ids. This method returns a list of the same length as measurements, each component containing the result of the overlap query in the appropriate format determined by the measurement type. See the various EpivizData-class classes for details on the result format.

Author(s)

Hector Corrada Bravo

See Also

startEpiviz EpivizData-class EpivizDataPack-class EpivizChart-class EpivizDevice-class GenomicRanges-class EpivizServer-class register-methods IndexedArray-class

Examples

```
## Not run:
  require(epivizrData)
  data(tcga_colon_example)

  mgr <- startEpiviz(openBrowser=interactive())

  # using the device interface
  blockDev <- mgr$addDevice(colon_blocks, "blocks_test", type="block")

  # using the measurement/chart interface
  # add measurements
  blockMs <- mgr$addMeasurements(colon_blocks, "blocks_test2", type="block")

  # add chart
  blockChart <- mgr$blockChart(blockMs$getMeasurements()[1])

  # using plot methods
  blockChart2 <- blockMs$plot()

  # list devices
  mgr$listDevices()

  # list measurements
  mgr$listMeasurements()

  # list charts
  mgr$listCharts()

  # remove a chart
  mgr$rmChart(blockChart2)

  # navigate to genomic region
  mgr$navigate("chr2", 10000000, 30000000)
  mgr$stopServer()

## End(Not run)
```

EpivizFeatureData-class
                                'EpivizFeatureData' objects

Description

Encapsulate data for genomic features

Details

The `object` field is constrained to be [`SummarizedExperiment`](). Data is obtained from columns of a determined `assay`. Measurement names are checked against the row names of the `colData` slot. The `rowData` is assumed to be a [`GIntervalTree`]() object, its `mcols` must contain columns PROBEID and SYMBOL.

Fields

`assay`: (integer or character) indicating assay from which data is obtained

Methods

`.checkColumns(columns)`: Checks `columns` are a subset of `names(colData(object))`.

`.getColumns()`: Returns `names(colData(object))`.

`plot()`: Calls the `scatterPlot` method of [`EpivizDeviceMgr-class`]() with the first measurements in `columns` as `x` and `y`.

`getMeasurements()`: Returns named character vector. Entries are of format '<name>$<column>'. Vector names are of format '<id>__<column>'.

`parseMeasurement(msId)`: Parses measurement of format '<id>__<column>'.

`packageData(msId)`: Get data for `msId` for rows overlapping `curQuery`. Returns a list with components: `min`: min value for data range, `max`: max value for data range, `data`: list with slots: `gene`: from SYMBOL column of `mcols(rowData(object))`, `start`: feature start position, `end`: feature end position, `probe`: from PROBEID column of `mcols(rowData(object))`, `value`: numeric data value.

Author(s)

Hector Corrada Bravo

EpivizFeatureDataPack-class

'EpivizFeatureDataPack' objects

Description

Package of feature-level data for transfer to web app.

Fields

`min`: (numeric) minimum of data range.

`max`: (numeric) maximum of data range.

`gene`: (character) gene symbol vector.

`probe`: (character) probe id vector.

`start`: (integer) feature start location.

`end`: (integer) feature end location.

`data`: (list) data entry for each measurement.

Author(s)

Hector Corrada Bravo

EpivizServer-class   *EpivizServer objects*

Description

This class encapsulates the websocket connection between the R/bioc session and the epiviz web app.

Details

Epiviz uses websockets to connect R/bioc sessions to the epiviz web app. Objects of this class encapsulates websocket objects defined in the httpuv package. The Epiviz session manager defined in class EpivizDeviceMgr calls methods in this class to send requests to the web app or to respond to requests. Requests are created by this class, so actions defined in the JS side are paralleled closely here (this makes the EpivizDeviceMgr) independent of the specifics in the JS side.

Fields

port: Port for websocket connection (default 7312)

websocket: A WebSocket object from the httpuv package

server: Internal use only: A server object from the httpuv package, set by the startServer method

interrupted: Internal use only: has an interrupt been issued

socketConnected: Internal use only: is the websocket connected

msgCallback: Internal use only: callback function on websocket message receive, set by bindMgr method

requestQueue: Internal use only: Queue for unsent requests

tryPorts: Interval use only: Try more ports when opening websocket server if port requested by user is in use.

Methods

startServer(...): Sets up the server and websocket objects from httpuv package. Specifically, it sets the server field by calling startServer passing port and a set of callbacks, including a onWSOpen callback that sets the websocket field and sets the msgCallback method as the websocket$onMessage callback.

stopServer(): Stops the websocket server, calling stopServer.

service(): A blocking loop for the websocket to listen and serve requests. Calls service.

stopService(): Signals an interrupt so the service loop is stopped.

runServer(): A convenience function that runs startServer and service. stopServer is called on exit.

isClosed(): Returns TRUE if server connection is closed

bindManager(mgr): Sets the msgCallback on websocket message callback. It calls methods from argument mgr, usually an object of class `EpivizDeviceMgr`. At minimum it must be a list with components verbose and getData.

sendRequest(request): Converts argument request to JSON and sends through websocket if connected, otherwise, adds to requestQueue.

sendRequestsInQueue(): Sends all requests currently in the request queue.

emptyRequestQueue(): Empties request queue.

Web App Requests

In all below, requestId is a callback function id set by the session manager.

addMeasurements(requestId, msType, measurements): adds a set of measurements to the web app. msType indicates the type of measurement, see the typeMap field of `EpivizDeviceMgr` objects for valid measurement types. measurements is a named character vector of measurement names. names(measuremnts) are the measurement ids.

rmMeasurements(requestId, measurements, msType): removes a set of measurements. Arguments are as addMeasurements above.

addChart(requestId, chartType, measurements): adds a chart to the web app. chartType indicates the type of chart to add, see `EpivizChart-class` for valid chart types. measurements is a character vector containing measurement ids displayed in the chart.

rmChart(requestId, chartId): removes chart from the web app.chartId specifies the id of the chart to remove. This is returned in response to an addChart request.

clearChartCaches(requestId, chartIds): clears data caches for given charts. Used when objects in R/bioc session are updated.chartIds is a vector of chart ids.

navigate(requestId, chr, start, end): Make web app navigate to given genomic region.

Author(s)

Hector Corrada Bravo

See Also

EpivizDeviceMgr-class, EpivizChart-class, httpuv, startServer, service, WebSocket

Examples

```
## Not run:
mgr <- list(getData=function(measurements, chr, start, end) {
                      return(chr)
                    },
            verbose=TRUE)

server <- epivizr:::EpivizServer$new(port=7123L)
server$bindManager(mgr)
server$startServer()
```

```
browseURL("http://localhost:7123/")
tryCatch(server$service(), interrupt=function(int) invisible())

## End(Not run)
```

EpivizTrackData-class   *'EpivizTrackData' objects*

Description

Data container for data displayed in tracks

Details

This class extends EpivizData-class directly. The data containing object is constrained to be of class GIntervalTree.

Author(s)

Hector Corrada Bravo

IndexedArray-class   *Indexed array*

Description

This class is used by EpivizDeviceMgr-class objects to store requests callbacks.

Fields

nextId Next integer id to return when item is appended

items Stored items

Methods

append(item): Append item to array, returns id of appended item

get(id): Return item with given id

empty(): Empty the array

Author(s)

Hector Corrada Bravo

See Also

EpivizDeviceMgr-class

Examples

```
array <- epivizr:::IndexedArray$new()
aId <- array$append("a")
array$get(aId)
```

Queue-class   *A Queue*

Description

A first-in-first-out data structure. Used by `EpivizServer-class` objects to queue requests.

Fields

`items` Items stored in queue

Methods

`push(item)`: Push item into queue

`pop()`: Pop item on top of queue

`empty()`: Empty the queue

Author(s)

Hector Corrada Bravo

See Also

`EpivizServer-class`

Examples

```
theQ <- epivizr:::Queue$new()
theQ$push("a")
theQ$pop()
```

---

register-methods   *Encapsulate data object in Epiviz*

Description

This method encapsulates data objects in R/bioc to be used for interactive visualization. It converts
GenomicRanges objects to GIntervalTree objects when appropriate. It returns objects from the
EpivizData-class hierarchy.

Usage

```
## S4 method for signature GenomicRanges
register(object, columns=NULL, type=c("block", "bp"), ...)
## S4 method for signature SummarizedExperiment
register(object, columns=NULL, assay=1)
## S4 method for signature ExpressionSet
register(object, columns=NULL, annotation=NULL)
```

Arguments

object	data object to encapsulate
columns	columns in object to use. All columns are used if NULL
type	for GenomicRanges objects, use block representation or base-pair level quantitative representation
assay	for SummarizedExperiment and ExpressionSet objects, determine which assay to use
annotation	for ExpressionSet objects, determine the annotation to use to obtain feature genomic coordinates. If NULL, then annotation(object) is used
...	arguments passed to EpivizData-class subclasses constructors

Details

This function converts GenomicRanges objects to GIntervalTree objects when appropriate. For
example, the rowData slot in SummarizedExperiment objects. Genomic coordinates for features
in ExpressionSet objects are obtained used the AnnotationDb interface from the provided annotation.

Value

Returns objects from the EpivizData-class hierarchy depending on the provided object. When
object is a GenomicRanges object and type=="block", a EpivizBlockData-class object is returned; a EpivizBpData-class object is returned if type=="bp". For SummarizedExperiment and
ExpressionSet objects, a EpivizFeatureData-class object is returned. For SummarizedExperiment
objects, columns SYMBOL and PROBEID are assumed to be present in mcols(rowData(object)).

Author(s)

Hector Corrada Bravo

See Also

EpivizData-class and subclasses, EpivizDeviceMgr-class, GenomicRanges, SummarizedExperiment, GIntervalTree, ExpressionSet

---

startEpiviz   *Start the epiviz interface*

Description

Create an epiviz session manager which can be used to add and delete tracks and plots in the browser web app.

Usage

```
startEpiviz(port = 7312L, localURL = NULL, useDevel = FALSE,
  chr = "chr11", start = 99800000, end = 103383180, debug = FALSE,
 proxy = TRUE, workspace = NULL, scripts=NULL, gists=NULL, openBrowser = TRUE,
  daemonized = .epivizrCanDaemonize(), verbose = FALSE,
  nonInteractive = FALSE, tryPorts = FALSE)
```

Arguments

port	(integer) port for websocket server
localURL	(character) use this url for the epiviz server instead of the standard URL (http://epiviz.cbcb.umd.edu). For example localURL="http://localhost/epiviz" when serving from the local host
useDevel	(logical) use the devel branch of the epiviz server (http://epiviz-dev.cbcb.umd.edu)
chr	(character) chromosome to browse to on startup
start	(integer) start position to browse to on startup
end	(integer) end position to browse to on startup
debug	(logical) start the epiviz browser in debug mode
proxy	(logical) start the epiviz browser using the data proxy. This is useful when hosting the web app locally, but want to use data hosted in the epiviz server. (DEPRECATED, currently ignored)
workspace	(character) a workspace id to load in the epiviz web app on startup
scripts	(character vector) URLs for JavaScript plugin scripts to be imported when epiviz is loaded (see http://epiviz.cbcb.umd.edu/help for details).
gists	(integer vector) Ids for github gists (http://gist.github.com) containing JavaScript plugin scripts to be imported when epiviz is loaded (see http://epiviz.cbcb.umd.edu/help for details).

openBrowser   (logical) browse to the epiviz URL before exiting function

daemonized   (logical) use non-blocking version if available (UNIX-like only)

verbose   (logical) display log messages on websocket requests

nonInteractive   (logical) run in non-interactive mode, for development purposes only

tryPorts   (logical) try more ports if port number given by argument port is in use.

Value

an object of class EpivizDeviceMgr.

Author(s)

Hector Corrada Bravo

See Also

EpivizDeviceMgr-class

Examples

```
## Not run:
mgr <- startEpiviz(openBrowser=FALSE)
mgr$startServer()
mgr$stopServer()

## End(Not run)
```

---

tcga_colon_example   *Example methylation data for epivizr vignette*

Description

This package contains example results from methylation analysis of human chromosome 11 using the minfi-package package of TCGA 450k beadarray samples.

Usage

```
data(tcga_colon_example)
```

Format

Two GRanges objects. The first colon_blocks, described large regions of methylation difference between tumor and normal samples. It has the following as metadata:

value average smooth methylation difference within block

area block area estimate (abs(value) * length)

cluster id of cluster blockgroup within which block occurs

`indexStart` index of first cluster in block

`indexEnd` index of last cluster in block

`L` number of clusters in block

`clusterL` number of probes in block

`p.value` permutation p.value based on difference conditioned on length

`fwer` family-wise error rate estimate based on difference conditioned on length

`p.valueArea` permutation p.value based on area

`fwerArea` family-wise error rate estimate based on area

The second, `colon_curves`, is probe cluster-level methylation estimates. It has the following as element metadata:

`id` probe cluster id

`type` probe cluster type

`blockgroup` probe cluster block group

`diff` raw methylation percentage difference between normal and tumor

`smooth` smooth methylation percentage difference between normal and tumor

`normalMean` mean methylation estimate for normal samples

`cancerMean` mean methylation estimate for cancer samples

Author(s)

Hector Corrada Bravo

Source

TCGA project: https://tcga-data.nci.nih.gov/tcga/

See Also

minfi-package

Examples

```
data(tcga_colon_example)
show(colon_blocks)
show(colon_curves)
```

---

tcga_colon_expression   *Example exon-level RNAseq data from TCGA project for epivizr.*

Description

A [SummarizedExperiment](#) object containing exon-level counts from RNAseq data from the TCGA project. Only exons in human chromosome 11 are included.

Usage

```
data(tcga_colon_expression)
```

Format

A [SummarizedExperiment](#) object with 12,800 rows (exons) and 40 samples.

assay(colonSE) exon-level count matrix

colData(colonSE) a `DataFrame` containing sample information. Normal/Tumor status is given in column `sample_type`

Source

TCGA project: https://tcga-data.nci.nih.gov/tcga/

Examples

```
data(tcga_colon_expression)
show(colonSE)
table(colData(colonSE)$sample_type)
```

Index