rtracklayer

April 19, 2009

`activeView-methods` *Accessing the active view*

Description

Get the active view.

Methods

The following methods are defined by **rtracklayer**.

object = "browserSession" `activeView(object)`: Gets the active `browserView` from a browser session.

`activeView(object) <- value`: Sets the active `browserView` in a browser session.

object = "ucscView" or "argoView" `activeView(object)`: Gets a logical indicating whether this is the active view.

Examples

```
## Not run:
  session <- browseGenome(browser = "argo")
  activeView(session)
  view <- browserView(session, genomeSegment("hg18", "chr22", 20000, 30000))
  activeView(session) <- view
## End(Not run)
```

`argoSession-class` *Class "argoSession"*

Description

An object representing a session in the Argo genome browser.

Objects from the Class

Objects may be created by calls of the form `browserSession("argo", name, jar)` where `name` is the name of the session in Argo and `jar` is the path to the installed Argo jar file.

Slots

jref: Object of class "jobjRef" holding the reference to the low-level Argo object.

seqcache: Object of class "environment" that caches sequence objects after lookup from UCSC.

Extends

Class "browserSession", directly.

Methods

activeView(object) Obtains the currently active browserView for this session.

activeView<-(object) Sets the currently active view for this session.

browserView(object, segment = genomeSegment(object),track = tracks(object, segme Creates a browserView of segment with visible tracks named in track. Parameters in ... correspond to slots in the genomeSegment class and override those in segment.

browserViews(object) Gets a list of the browserView instances for this session.

close(con) Closes this session.

genomeSequence(object, segment) Gets the genome sequence in segment.

laySequence(object, sequence, name, label = name) Stores sequence under name, labeled as label in the user interface.

layTrack(object, track, name = deparse(substitute(track)), view = TRUE) Stores track under name, opening a view of the track if view is TRUE.

trackSet(object, name, segment = genomeSegment(object)) Obtains a trackSet named name from segment.

tracks(object, segment = NULL, visible = FALSE) Gets the names of tracks in segment. If visible is TRUE, only the visible tracks are returned. Note that in Argo tracks are visible on a global, not per-session nor per-view, basis.

Note

In order to use this backend, the Argo jar file must be manually downloaded from the URL in the references. The call to browserSession must then pass the path to the jar file as the jar parameter. By default, the path is the 'java' directory of the **rtracklayer** package.

Author(s)

Michael Lawrence

References

The Argo genome browser: http://www.broad.mit.edu/annotation/argo/

See Also

browserSession for creating instances of this class.

argoView-class *Class "argoView"*

Description

An object representing a view of a genome in the Argo browser.

Objects from the Class

Objects may be created through calls of the form `browserView(session, ...)` where `session` is an instance of `argoSession`.

Slots

jref: Object of class `"jobjRef"` holding a reference to the low-level Argo object.

session: Object of class `"browserSession"` to which this view belongs.

Extends

Class `"browserView"`, directly.

Methods

activeView(object) Returns a logical indicating whether this view is the active view.

close(con) Closes this view.

genomeSegment(object) Obtains the `genomeSegment` displayed by this view.

genomeSegment(object) <- value Sets the `genomeSegment` displayed by the view.

tracks(object) Gets the names of the visible tracks.

tracks<-(object) Sets the visible tracks by name. Note that in Argo track visibility is global.

Author(s)

Michael Lawrence

See Also

`browserView` for creating instances from a `argoSession`.

basicTrackLine-class *Class "basicTrackLine"*

Description

The type of UCSC track line used to annotate most types of tracks (every type except Wiggle).

Objects from the Class

Objects can be created by calls of the form `new("basicTrackLine", ...)` or parsed from a character vector track line with `as(text, "basicTrackLine")` or converted from a `wigTrackLine` using `as(wig, "basicTrackLine")`.

Slots

itemRgb: Object of class `"logical"` indicating whether each feature in a track uploaded as BED should be drawn in its specified color.

useScore: Object of class `"logical"` indicating whether the data value should be mapped to color.

group: Object of class `"character"` naming a group to which this track should belong.

db: Object of class `"character"` indicating the associated genome assembly.

offset: Object of class `"numeric"`, a number added to all positions in the track.

url: Object of class `"character"` referring to additional information about this track.

htmlUrl: Object of class `"character"` referring to an HTML page to be displayed with this track.

name: Object of class `"character"` specifying the name of the track.

description: Object of class `"character"` describing the track.

visibility: Object of class `"character"` indicating the default visible mode of the track, see `ucscTrackModes`.

color: Object of class `"integer"` representing the track color (as from `col2rgb`).

priority: Object of class `"numeric"` specifying the rank of the track.

Extends

Class `"ucscTrackLine"`, directly.

Methods

as(object, "character") Export line to its string representation.

as(object, `"wigTrackLine"`) Convert this line to a WIG track line, using defaults for slots not held in common.

Author(s)

Michael Lawrence

References

http://genome.ucsc.edu/goldenPath/help/customTrack.html#TRACK for the official documentation.

See Also

wigTrackLine for Wiggle tracks.

`browseGenome` *Browse a genome*

Description

A generic function for launching a genome browser.

Usage

```
browseGenome(tracks = trackSets(), browser = "ucsc",
             segment = genomeSegment(tracks), view = TRUE,
             trackParams = list(), viewParams = list(), ...)
```

Arguments

`tracks`	A list of `trackSet` instances, e.g. a `trackSets` instance.
`browser`	The name of the genome browser.
`segment`	The `genomeSegment` to display in the initial view.
`view`	Whether to open a view.
`trackParams`	Named list of parameters to pass to `layTrack`.
`viewParams`	Named list of parameters to pass to `browserView`.
`...`	Arguments corresponding to slots in `genomeSegment` that override those in `segment`.

Value

Returns a `browserSession`.

Author(s)

Michael Lawrence

See Also

`browserSession` and `browserView`, the two main classes for interfacing with genome browsers.

Examples

```
## Not run:
## open UCSC genome browser:
browseGenome()
## to view a specific segment:
segment <- genomeSegment("hg18", "chr22", 20000, 50000)
browseGenome(segment = segment)
## a slightly larger segment:
browseGenome(segment = segment, end = 75000)
## with a track:
track <- import(system.file("tests", "v1.gff", package = "rtracklayer"))
browseGenome(trackSets(track))

## End(Not run)
```

---

browserSession-class

Class "browserSession"

Description

An object representing a genome browser session. Each session corresponds to a set of loaded `trackSet` instances and a set of `browserView` instances. Note that this is a virtual class; a concrete implementation is provided by each backend driver.

Objects from the Class

A virtual Class: No objects may be created from it. See `browserSession` for obtaining an instance of an implementation for a particular genome browser.

Methods

This specifies the API implemented by each browser backend. Note that a backend is not guaranteed to support all operations.

`browserView(object, segment = genomeSegment(object), track = tracks(object), ...`
Constructs a `browserView` of `segment` for this session.

`browserViews(object, ...)` Gets the `browserView` instances belonging to this session.

`activeView(object, ...)` Returns the `browserView` that is currently active in the session.

`genomeSegment(object, ...)` Gets the `genomeSegment` representing the segment of the genome currently displayed by the browser (i.e. the segment shown by the active view) or a default value (possibly `NULL`) if no views exist.

`genomeSequence(object, segment = genomeSegment(object), ...)` gets a genomic sequence of `segment` from this session.

`laySequence(object, ...)` Loads a sequence into the session.

`layTrack(object, track, name = deparse(substitute(track)), view = TRUE, ...)`
Loads one or more `tracks` into the session and optionally open a `view` of the track.

`trackSet(object, segment = genomeSegment(object), name, ...)` Gets features in `segment` in the track stored as `name` as a `trackSet`.

tracks(object, ...) Gets the names of the tracks stored in this session.

close(con, ...) Close this session.

show(object, ...) Output a textual description of this session.

Author(s)

Michael Lawrence

See Also

browserSession for obtaining implementations of this class for a particular genome browser.

---

browserSession-methods *Get a genome browser session*

Description

Methods for getting browser sessions.

Methods

The following methods are defined by **rtracklayer**.

object = "character" browserSession(object, ...): Creates a browserSession from a genome browser identifier. The identifier corresponds to the prefix of the session class name (e.g. "ucsc" in "ucscSession"). The arguments in ... are passed to the initialization function of the class.

object = "browserView" Gets the browserSession for the view.

object = "missing" Calls browserSession("ucsc", ...).

---

browserView-class *Class "browserView"*

Description

An object representing a genome browser view of a particular genomeSegment of a genome.

Objects from the Class

A virtual Class: No objects may be created from it directly. See browserView for obtaining an instance of an implementation for a particular genome browser.

Slots

session: Object of class "browserSession" the browser session to which this view belongs.

Methods

This specifies the API implemented by each browser backend. Note that a backend is not guaranteed to support all operations.

browserSession(object) Obtains the browserSession to which this view belongs.

close(object) Close this view.

genomeSegment(object) Obtains the genomeSegment displayed by this view.

tracks(object) Gets the names of the visible tracks in the view.

tracks(object) <- value Sets the visible tracks by their names.

show(object) Outputs a textual description of this view.

Author(s)

Michael Lawrence

See Also

browserView for obtaining instances of this class.

---

browserView-methods

Getting browser views

Description

Methods for creating and getting browser views.

Usage

```
browserView(object, segment = genomeSegment(object),
            track = tracks(object), ...)
```

Arguments

object	The object from which to get the views.
segment	The genomeSegment to display.
track	List of track names to make visible in the view.
...	Arguments to pass to methods

Methods

The following methods are defined by **rtracklayer**.

object = "ucscSession" `browserView(object, segment = genomeSegment(object), track = tracks(object), ...)`: Creates a browserView of segment with visible tracks specified by track. track may be an instance of ucscTrackModes. Arguments in ... should override slots in segment or else match parameters to a ucscTrackModes method for creating a ucscTrackModes instance that will override modes indicated by the track parameter.

object = "argoSession" `browserView(object, segment = genomeSegment(object), track = tracks(object, segment, TRUE), ...)`: Creates a `browserView` of `segment` with visible tracks named in `track`. Parameters in `...` correspond to slots in the `genomeSegment` class and override those in `segment`.

Examples

```
## Not run:
  session <- browserSession()
  browserView(session, genomeSegment(start = 20000, end = 50000,
                                     segment = genomeSegment(session)))
  ## equivalent to above, but shorter
  browserView(session, start = 20000, end = 50000)
  ## only view "knownGene" track
  browserView(session, track = "knownGene")
## End(Not run)
```

`browserViews-methods`

Getting the browser views

Description

Methods for getting browser views.

Methods

The following methods are defined by **rtracklayer**.

Gets the instances of `browserView` in the session.

object = "ucscSession", **object = "argoSession"** Gets the instances of `browserView` in the session.

See Also

`browserView` for creating a browser view.

Examples

```
## Not run:
session <- browseGenome()
browserViews(session)
## End(Not run)
```

chrid-class *Class "chrid"*

Description

An object representing a chromosome identifier. Meant to be used for subsetting `trackSet` instances by chromosome.

Objects from the Class

Instances may be created through a call of the form `chrid(id)`, where `id` is the string identifier of the chromosome (e.g. "chr22").

Slots

`.Data`: Object of class `"character"` holding the string representation of the chromosome ID.

Extends

Class `"character"`, from data part. Class `"vector"`, by class "character", distance 2. Class `"characterORMIAME"`, by class "character", distance 2.

Methods

No methods defined with class "chrid" in the signature.

Author(s)

Michael Lawrence

See Also

`chrid` for obtaining instances of this class.

`[.trackSet` where instances of this class may be used for subsetting `trackSet` instances by chromosome.

chrid-methods *Getting chromosome ids*

Description

Methods for creating and getting chromosome ids.

Methods

The following methods are defined by **rtracklayer**.

object = "trackSet" Gets the chromosomes of the features as a `chrid` instance.

object = "character" Coerces vector of string chromosome identifiers to a `chrid` instance.

object = "ANY" Attempts to coerce `object` by first coercing it to a character vector.

See Also

Subset trackSet instances by chromosome identifier with [.trackSet

Examples

```
chrid("chr22")
track <- import(system.file("tests", "bed.wig", package = "rtracklayer"))
chrid(track)
```

---

cpneTrack *CPNE1 SNP track*

Description

A trackSet object (created by the GGtools package) with features from a subset of the SNPs on chromosome 20 from 60 HapMap founders in the CEU cohort. Each SNP has an associated data value indicating its association with the expression of the CPNE1 gene according to a Cochran-Armitage 1df test. The top 5000 scoring SNPs were selected for the track.

Usage

```
data(cpneTrack)
```

Format

Each feature is described by its start, stop, chromosome and strand in the genome. There is a single column of data values, representing the association test scores.

Source

Vince Carey and the GGtools package.

Examples

```
data(cpneTrack)
plot(start(cpneTrack), dataVals(cpneTrack))
```

---

dataVals-methods *Get data values*

Description

Methods for getting data values.

Methods

The following methods are defined by **rtracklayer**.

object = "trackSet" Get the experimental measurements from a track.

Examples

```
track <- import(system.file("tests", "bed.wig", package = "rtracklayer"))
dataVals(track)
```

export *Export objects to connections*

Description

Exports (serializes) an object in a given format to a given connection.

Usage

```
export(object, con, format, ...)
```

Arguments

object	The object to export.
con	The connection to which the object is exported. If this is a character vector, it is assumed to be a filename and a corresponding file connection is created and then closed after exporting the object. If missing, the function will return the output as a character vector, rather than writing to a connection.
format	The format of the output. If missing and con is a filename, the format is derived from the file extension.
...	Parameters to pass to the format-specific export routine.

Details

This function delegates to another function, depending on the specified format. The name of the delegate is of the form export.format where format is specified by the format argument.

Value

If con is missing, a character vector containing the string output. Otherwise, nothing is returned.

Author(s)

Michael Lawrence

See Also

import for the reverse

Examples

```
  track <- import(system.file("tests", "v1.gff", package = "rtracklayer"))
  ## Not run: export(track, "my.gff", version = "3")
  ## equivalently,
  ## Not run: export(track, "my.gff3")
  ## or
  ## Not run:
  con <- file("my.gff3")
  export(track, con, "gff3")
  close(con)

## End(Not run)
```

```
## or as a string
export(track, format = "gff3")
```

`export-tracks` *Export tracks*

Description

These functions output `trackSet` instances in various formats.

Usage

```
export.gff(object, con, version = c("1", "2", "3"), source =
           "rtracklayer")
export.gff1(object, con, ...)
export.gff2(object, con, ...)
export.gff3(object, con, ...)
export.bed(object, con, wig = FALSE, color = NULL, ...)
export.wig(object, con,
           dataFormat = c("auto", "bed", "variableStep", "fixedStep"), ...)
export.ucsc(object, con, subformat = c("auto", "gff1", "wig", "bed"), ...)
```

Arguments

`object`	The object to export, such as a `trackSet`. If a `ucscTrackSet`, the track line information is output. In the case of `export.ucsc`, a `trackSets` object with possibly multiple tracks is supported.
`con`	The connection to which the object is exported.
`version`	The GFF version, either "1", "2" or "3" (default is "1").
`source`	The source of the GFF information, for GFF.
`wig`	Whether to output the WIG variant of BED lines, not to be used directly.
`color`	Recycled vector of colors, as interpreted by `col2rgb` for BED features. If `NULL`, the `color` column in the `featureData` is used, if any.
`dataFormat`	The format of the data lines for WIG tracks, see references. The "auto" format uses the most efficient format possible.
`subformat`	The format of the tracks within the UCSC container. If "auto", "wig" is used for numeric data, else "bed".
`...`	For `export.gff1`, `export.gff2` and `export.gff3`: arguments to pass to `export.gff`. For `export.bed` and `export.wig` : arguments to pass to methods. For `export.ucsc`: arguments to pass to `export.subformat` or to set on the slots of the `ucscTrackLine` subclass corresponding to `subformat`.

Value

If `con` is missing, a character vector containing the string output, otherwise nothing.

Author(s)

Michael Lawrence

References

GFF1 and GFF2 http://www.sanger.ac.uk/Software/formats/GFF

GFF3 http://www.sequenceontology.org/gff3.shtml

BED http://genome.ucsc.edu/goldenPath/help/customTrack.html#BED

WIG http://genome.ucsc.edu/goldenPath/help/wiggle.html

UCSC http://genome.ucsc.edu/goldenPath/help/customTrack.html

See Also

See `export` for the high-level interface to these functions.

Examples

```
dummy <- file() # dummy file connection for demo
track <- import(system.file("tests", "bed.wig", package = "rtracklayer"))
## output a track as GFF2
export.gff(track, dummy, version = "2")
## equivalently
export.gff2(track, dummy)
## output as WIG string in variableStep format
wig <- export.wig(track, dummy, dataFormat = "variableStep")
## output multiple tracks in UCSC meta-format
track2 <- import(system.file("tests", "v1.gff", package = "rtracklayer"))
## output to WIG with BED line format
export.ucsc(trackSets(track, track2), dummy, subformat = "wig", dataFormat = "bed")
```

---

genomeBrowsers *Get available genome browsers*

Description

Gets the identifiers of the loaded genome browser drivers.

Usage

```
genomeBrowsers(where = topenv(parent.frame()))
```

Arguments

`where`	The environment in which to search for drivers.

Details

This searches the specified environment for classes that extend `browserSession`. The prefix of the class name, e.g. "ucsc" in "ucscSession", is returned for each driver.

Value

A character vector of driver identifiers.

Author(s)

Michael Lawrence

See Also

browseGenome and browserSession that create browserSession implementations given an identifier returned from this function.

---

genomeSegment-class

Class "genomeSegment"

Description

An object representing a segment of a genome.

Objects from the Class

Objects are normally created by a call of the following form: genomeSegment(genome, chrom, start, end, segment). All parameters are optional and correspond to slots of the new genomeSegment instance, except for the segment parameter, which serves as the base for the new instance. The slots of the new segment match those of segment, unless the call specifies other parameters, which override the corresponding slots in segment.

Slots

genome: Object of class "character" giving the name of the genome (e.g. "hg18").

chrom: Object of class "character" giving the chromosome name (e.g. "chr22").

start: Object of class "numeric" indicating the start position of the segment on the chromosome.

end: Object of class "numeric" indicating the stop position of the segment on the chromosome.

Methods

merge(x, y) Merges two instances. The result has the same slot values as x, except the non-empty slots in y override those from x.

genome(object) Gets the genome identifier.

genome(object) <- value Sets the genome identifier.

chrom(object) Gets the chromosome identifier.

chrom(object) <- value Sets the chromosome identifier.

start(object) Gets the start position.

start(object) <- value Sets the start position.

end(object) Gets the end position.

end(object) <- value Sets the end position.

object / x Zoom out x times (expand segment by x/2 on each side).

object * x Zoom in x times (contract segment by x/2 on each side).

Author(s)

Michael Lawrence

See Also

`genomeSegment` for obtaining instances of this class.

`genomeSegment-methods`
Accessing a segment of a genome

Description

Methods for creating, getting or setting a `genomeSegment`.

Methods

The following methods are defined by **rtracklayer**.

object = "character" `genomeSegment(object, chrom = character(0), start = character(0), end = character(0), segment = genomeSegment())`: Create a `genomeSegment` instance with the same slot values as `segment`, except where overriden by one of the other parameters, which all correspond to slots in `genomeSegment`. Note that `object` corresponds to the `genome` slot.

object = "missing" `genomeSegment(object, genome = character(0), chrom = character(0), start = character(0), end = character(0), segment = new("genomeSegment"))`: Similar to above, except `object` is omitted and `genome` is an explicit parameter.

object = "trackSet" Get the segment spanned by a track.

object = "trackSets" Get the union (including gaps) of spans for all tracks.

object = "browserSession" Get the current segment (i.e. segment displayed by active view or default segment).

object = "ucscSession" Get the last accessed segment.

object = "ucscView" or "argoView" `genomeSegment(object)`: Get the segment displayed by the view.

`genomeSegment(object) <- value`: Set the segment displayed by the the view.

object = "IRanges" Obtain the genome segment that spans the range of the `IRanges` object.

Examples

```
## Create a genome segment for genome "hg18", chromosome 22
genomeSegment("hg18", "chr22")
## Explicitly specify parameters
segment <- genomeSegment(genome = "hg18", chrom = "chr22", start = 150000)
## Add an 'end' value
genomeSegment(end = 200000, segment = segment)
```

genomeSequence-methods *Retrieving a genome sequence*

Description

Methods for retrieving the sequence of a `genomeSegment` from an object.

Methods

The following methods are defined by **rtracklayer** for `genomeSequence(object, segment = genomeSegment(object), ...)`.

object = "ucscSession" `genomeSequence(object, segment = genomeSegment(object), track = "Assembly")`: Gets the sequence in `segment` and `track` from the session.

object = "argoSession" `genomeSequence(object, segment)`: Gets the sequence in `segment` from the session.

See Also

`laySequence` for storing sequences.

import *Importing objects*

Description

Imports an object from a connection according to a specified format.

Usage

```
import(con, format, text, ...)
```

Arguments

`con`	The connection through which the data is received. If this is a character vector, it is assumed to be a filename.
`format`	The format in which to expect the input. If omitted and `con` is a filename, the format is taken from the file extension.
`text`	If `con` is missing, this can be a character vector directly providing the string data to import.
`...`	Arguments to pass to the format-specific import routines.

Details

This function delegates to a format-specific function named according to the scheme `import.format` where `format` is specified by the `format` parameter.

Value

The object parsed from the connection or text.

Author(s)

Michael Lawrence

See Also

export to do the reverse.

Examples

```
track <- import(system.file("tests", "bed.wig", package = "rtracklayer"))
track <- import(system.file("tests", "v1.gff", package = "rtracklayer"), version = "1")
# or
track <- import(system.file("tests", "v1.gff", package = "rtracklayer"), "gff1")
```

---

import.gff *Importing tracks*

Description

These are the functions for importing trackSet instances from connections or text.

Usage

```
import.gff(con, version = c("1", "2", "3"), genome = "hg18")
import.gff1(con, ...)
import.gff2(con, ...)
import.gff3(con, ...)
import.bed(con, wig = FALSE, trackLine = !wig, genome = "hg18", ...)
import.wig(con, genome = "hg18", ...)
import.ucsc(con, subformat = c("auto", "gff1", "wig", "bed"),
            drop = FALSE, ...)
```

Arguments

con	The connection from which to receive the input.
version	The version of GFF ("1", "2" or "3").
genome	The genome to set on the imported track.
wig	Whether the BED lines are expected to be WIG formatted (not for public use).
trackLine	Whether the BED data has a track line (it normally does though track lines are not mandatory).
subformat	The expected subformat of the UCSC data. If "auto", automatic detection of the subformat is attempted.
drop	If TRUE and there is only one track in the UCSC data, return the track instead of a list.
...	For import.gff1, import.gff2 and import.gff3: arguments to pass to import.gff. For import.bed and import.wig: arguments to pass to methods. For import.ucsc: arguments to pass on to import.subformat.

Value

An instance of trackSet or one of its subclasses, except for import.ucsc, which returns a trackSets instance, unless there is one track and the drop parameter is TRUE.

Author(s)

Michael Lawrence

References

GFF1 and GFF2 http://www.sanger.ac.uk/Software/formats/GFF

GFF3 http://www.sequenceontology.org/gff3.shtml

BED http://genome.ucsc.edu/goldenPath/help/customTrack.html#BED

WIG http://genome.ucsc.edu/goldenPath/help/wiggle.html

UCSC http://genome.ucsc.edu/goldenPath/help/customTrack.html

See Also

import for the high-level interface to these routines.

Examples

```
# import a GFF V2 file
gff <- import.gff(system.file("tests", "v2.gff", package = "rtracklayer"), version = "2
# or
gff <- import.gff2(system.file("tests", "v2.gff", package = "rtracklayer"))

# import a WIG file
wig <- import.wig(system.file("tests", "bed.wig", package = "rtracklayer"))
# or
wig <- import.ucsc(system.file("tests", "bed.wig", package = "rtracklayer"),
                   subformat = "wig", drop = TRUE)
```

---

laySequence-methods

Load a sequence

Description

Methods for loading sequences.

Methods

The following methods are defined by **rtracklayer** for the laySequence(object, sequence, ...) generic.

laySequence(object, sequence, name, label = name): Load sequence into object under the name name and labeled by label in the user interface.

object = "argoSession", sequence = "character"

`layTrack-methods` *Laying tracks*

Description

Methods for loading `trackSet`s into genome browsers.

Usage

```
layTrack(object, track, name = deparse(substitute(track)), view = FALSE, ...)
```

Arguments

`object`	A `browserSession` into which the track is loaded.
`track`	The track(s) to load.
`name`	The name(s) of the track(s) being loaded.
`view`	Whether to create a view of the track after loading it.
`...`	Arguments to pass on to methods.

Methods

The following methods are defined by **rtracklayer**. A browser session implementation must implement a method for either `trackSet` or `trackSets`. The base `browserSession` class will delegate appropriately.

object = "browserSession", track = "trackSet" Load this track into the session.

object = "browserSession", track = "trackSets" Load all tracks into the session.

object = "ucscSession", track = "trackSets" `layTrack(object, track, name = deparse(substitut view = FALSE, format = "gff")`: Load the tracks into the session using the specified `format`.

object = "argoSession", track = "trackSet" Load the track into the session.

See Also

`trackSet` for getting `trackSet` instances from a session.

Examples

```
## Not run:
  session <- browserSession()
  track <- import(system.file("tests", "v1.gff", package = "rtracklayer"))
  layTrack(session, track, "My Track")
## End(Not run)
```

trackData *Get track data*

Description

Convert the track features and experimental measurements to a `data.frame`.

Usage

```
trackData(object, ...)
```

Arguments

`object`	An object with track information, e.g. `trackSet`.
`...`	Arguments to pass to methods

Value

A `data.frame` with a column for each feature variable and experimental sample. An additional column `seqMid` is added that holds the mid-point of each feature (helpful for plotting).

Author(s)

Michael Lawrence

See Also

`chrom`, `start`, `dataVals`, etc for accessing individual elements of the track.

trackSet-class *Class "trackSet"*

Description

An object representing a genome annotation track. Based on `eSet`, with feature information (chrom, start, end, strand, ...) stored in the `featureData` and data values from experimental measurements stored in the `assayData`.

Objects from the Class

Objects of the class are generally created in two different ways, depending on how the experimental data are provided. The first way is to pass an instance of `AssayData` as the `dataVals` parameter in `new("trackSet", assayData, phenoData = annotatedDataFrameFrom(assayData, byrow=FALSE), featureData = annotatedDataFrameFrom(assayData, byrow=TRUE), experimentData = new("MIAME"), annotation = character(), genome = "hg18")`. The `AssayData` object must contain a matrix or data.frame named `dataVals`, which should contain the experimental measurements. Alternatively, the experimental measurements may be passed as the `dataVals` parameter in `new("trackSet", phenoData = annotatedDataFrameFrom(assayDat byrow=FALSE), featureData = annotatedDataFrameFrom(assayData, byrow=TRUE), experimentData = new("MIAME"), annotation = character(), dataVals = matrix(), genome = "hg18")`.

Slots

genome: Object of class "character" identifying the genome to which this track pertains. Should be specified according to the conventions of the UCSC browser (e.g. "hg18"), if possible.

assayData: Object of class "AssayData" holding the experimental measurements.

phenoData: Object of class "AnnotatedDataFrame" holding the experimental design matrix.

featureData: Object of class "AnnotatedDataFrame" holding the feature information, generally including columns chrom (chromosome name), start (numeric start position), end (numeric end position) and strand (DNA strand: "+", "-" or NA). Note that the intervals are closed, i.e. the selected region is [start,stop].

experimentData: Object of class "MIAME" with experimental metadata.

annotation: Object of class "character" identifying the annotation resource.

.__classVersion__: Object of class "Versions" specifying the version of the class.

Extends

Class "eSet", directly. Class "VersionedBiobase", by class "eSet", distance 2. Class "Versioned", by class "eSet", distance 3.

Methods

object[i, j] Subsets an instance, where i is an feature index, feature name or chrid for selecting features and j is a sample index or name for selecting samples from the experimental data.

chrid(object) Obtains an instance of chrid, which is a vector with an element representing the chromosome for each feature in this track.

dataVals(object) Retrieves the matrix containing the experimental data values.

chrom(object) Retrieves the chromosome identifiers of the features.

start(x) Retrieves the start positions of the features.

end(x) Retrieves the end positions of the features. Note that intervals are closed, so this should be the index of the last base to include in the feature.

strand(object) Retrieves the strand ("+", "-" or NA) of the features.

genomeSegment(object) obtain the genome segment spanned by this track (including gaps).

trackData(object) Obtain a representation of this track as a data.frame. The result is the column-wise combination of the featureData slot with the dataVals matrix. The function also adds a seqMid column, which holds the mid-point of the segment of each feature. This is useful for e.g. plotting.

genome(object) Get the genome slot.

as(trackSet, "data.frame") Equivalent to trackData above.

trackSets(object) Create a trackSets instance initially containing only this track.

export.bed(object, con, wig = FALSE) Export track in Browser Extended Display (BED) format.

export.gff(object, con, version = c("1", "2", "3"), source = "rtracklayer") Export track in General Feature Format (GFF).

export.ucsc(object, con, subformat = c("gff", "wig"), name = deparse(substitute(
Export track in the meta format of the (UCSC) browser.

export.wig(object, con, dataFormat = c("bed", "variableStep", "fixedStep")) export track in Wiggle (WIG) format.

Author(s)

Michael Lawrence

See Also

`layTrack` for loading a track into a genome browser. `import` to create a `trackSet` from a file, `export` to write a `trackSet` to a file.

`trackSet-methods` *Get a trackSet*

Description

Methods for getting a `trackSet`.

Usage

```
trackSet(object, ...)
```

Arguments

`object`	An object from which to obtain a `trackSet`.
`...`	Arguments passed on to methods.

Methods

The following methods are defined by **rtracklayer**.

`trackSet(object, dataVals = NULL, ...)`: A constructor that sets `object` directly as the `featureData` slot and `dataVals` into the `dataVals` slot, if non-`NULL`. The arguments in `...` should correspond to slots in the `trackSet` class.

object = "AnnotatedDataFrame", object = "data.frame" `trackSet(object, dataVals = NULL, ...)`: coerces `object` to `AnnotatedDataFrame` and delegates to it.

object = "IRanges" `trackSet(object, chrom, strand = NA, dataVals = NULL, ...)`: Constructor that takes the start and end of each feature from `object`. The chromsome, strand, data values and other slots are specified by `chrom`, `strand`, `dataVals` and `...`, respectively.

object = "character" `trackSet(object, start = 1, end = start + span - 1, strand = NA, span = 1, dataVals = NULL, breaks = NULL, ...)`: a constructor that takes `object` to be the chromosome values. `start`, `end`, `strand` and `dataVals` specify the starts, ends, strands and data values, respectively, of the features. `span`, if given, should be a scalar integer indicating widths of the features If `end` is specified with `span`, it must be length one and then the region from `start` to `end` is broken up into features of width `span`. If `breaks` is specified, `span` must be omitted and `start` and `end` must be scalars. Then the features are formed by breaking the region from `start` to `end` at the positions specified in `breaks`. Way too complicated for its own good.

object = "ucscSession" `trackSet(object, name, segment = genomeSegment(object), format = "bed", table = NULL)`: Get a track from the session by its `name` in format `format`. Some built-in tracks have multiple series, each identified by a `table` name.

object = "argoSession" `trackSet(object, name, segment = genomeSegment(object))`: Get a track from the session by its `name`.

See Also

layTrack for loading a trackSet into a session.

Examples

```
## Not run:
session <- browseGenome()
track <- import(system.file("tests", "v1.gff", package = "rtracklayer"))
session <- layTrack(session, track, "My Track", format = "gff1", view = TRUE)
track <- trackSet(session, name = "My Track")
## End(Not run)
```

---

trackSets-class *Class "trackSets"*

Description

A list of trackSet instances.

Objects from the Class

Objects are normally created using the trackSets function.

Slots

.Data: Object of class "list" containing the tracks.

Extends

Class "list", from data part. Class "vector", by class "list", distance 2. Class "AssayData", by class "list", distance 2.

Methods

export.ucsc(object, con, subformat = c("gff", "wig", "bed"), name = names(object
Export all tracks in UCSC format.

genomeSegment(object) Obtain the union of the segments spanned by each track in the list.

Author(s)

Michael Lawrence

See Also

trackSets for obtaining instances of this class.

layTrack for loading the tracks into a genome browser.

trackSets-methods *Getting a list of trackSets*

Description

Methods for creating and getting a `trackSets` object: a list of `trackSet` instances.

Methods

The following methods are defined by **rtracklayer** for the `trackSets(object, ...)` generic.

object = "trackSet" Create an instance with track `object` and any additional tracks in `...`.

object = "missing" Creates an empty instance.

object = "list" Coerces a regular list of `trackSet` instances.

tracks-methods *Accessing track names*

Description

Methods for getting and setting track names.

Methods

The following methods are defined by **rtracklayer** for **getting** track names via the generic `tracks(object, ...)`.

Get the tracks loaded in the session.

object = "ucscSession", object = "ucscTrackModes" Get the visible tracks according to the modes (all tracks not set to "hide").

object = "ucscView" Get the visible tracks in the view.

object = "argoSession" `tracks(object, segment, visible = FALSE)`: Gets the tracks on `segment` in the session. Restricted to visible tracks if `visible` is `TRUE`.

object = "argoView" Gets the tracks visible in the view. Note that this is a global property in Argo.

The following methods are defined by **rtracklayer** for **setting** track names via the generic `tracks(object) <- value`.

object = "ucscTrackModes" Sets the tracks that should be visible in the modes. All specified tracks with mode "hide" in `object` are set to mode "full". Any tracks in `object` that are not specified in the value are set to "hide". No other modes are changed.

object = "ucscView" Sets the visible tracks in the view. This opens a new web browser with only the specified tracks visible.

object = "argoView" Sets the visible tracks globally (not just in the given view).

ucscSession-class *Class "ucscSession"*

Description

An implementation of browserSession for the UCSC genome browser.

Objects from the Class

Objects can be created by calls of the form browserSession("ucsc", url = "http://genome.ucsc.edu bin", ...). The arguments in ... correspond to libcurl options, see getCurlHandle. Setting these options may be useful e.g. for getting past a proxy.

Slots

url: Object of class "character" holding the base URL of the UCSC browser.

hguid: Object of class "numeric" holding the user identification code.

views: Object of class "environment" containing a list stored under the name "instances". The list holds the instances of browserView for this session.

Extends

Class "browserSession", directly.

Methods

browserView(object, segment = genomeSegment(object), track = tracks(object), ...
Creates a browserView of segment with visible tracks specified by track. track may be an instance of ucscTrackModes. Arguments in ... should override slots in segment or else match parameters to a ucscTrackModes method for creating a ucscTrackModes instance that will override modes indicated by the track parameter.

browserViews(object) Gets the browserView instances for this session.

genomeSegment(object) Gets the genomeSegment last displayed in this session.

genomeSequence(object, segment, track = "Assembly") Gets the sequence in segment and track.

layTrack(object, track, name = names(track), view = TRUE, format = "gff", ...)
Loads a track, stored under name and formatted as format. The arguments in ... are passed on to export.ucsc, so they could be slots in a ucscTrackLine subclass or parameters to pass on to the export function for format.

trackSet(object, segment = genomeSegment(), name = deparse(substitute(object)),
Retrieves a trackSet with features in segment from track named name. Some built-in tracks have multiple series, each stored in a separate database table. A specific table may be retrieved by passing its name in the table parameter.

tracks(object) Gets the names of the tracks stored in the session.

ucscTrackModes(object) Gets the default view modes for the tracks in the session.

ucscTable(object) Gets the database table in segment from track as a data.frame.

Author(s)

Michael Lawrence

See Also

browserSession for creating instances of this class.

---

ucscTable *Get a database table from UCSC*

Description

A generic function for getting a database table from UCSC. This is mostly for internal use though some may find it useful.

Usage

```
ucscTable(object, segment, track, table)
```

Arguments

object	The e.g. ucscSession
segment	The genomeSegment for which to retrieve the data.
track	The name of the track to which the table belongs.
table	The name of the table.

Value

A data.frame of the table.

Author(s)

Michael Lawrence

References

The UCSC Table Browser: http://genome.ucsc.edu/cgi-bin/hgTables.

See Also

trackSet for getting tracks, genomeSequence for getting sequences.

ucscTrackLine-class *Class "ucscTrackLine"*

Description

An object representing a "track line" in the UCSC format. There are two concrete types of track lines: basicTrackLine (used for most types of tracks) and wigTrackLine (used for Wiggle tracks). This class only declares the common elements between the two.

Objects from the Class

Objects can be created by calls of the form new("ucscTrackLine", ...) or parsed from a character vector track line with as(text, "ucscTrackLine"). But note that UCSC only understands one of the subclasses mentioned above.

Slots

name: Object of class "character" specifying the name of the track.

description: Object of class "character" describing the track.

visibility: Object of class "character" indicating the default visible mode of the track, see ucscTrackModes.

color: Object of class "integer" representing the track color (as from col2rgb).

priority: Object of class "numeric" specifying the rank of this track.

Methods

as(object, "character") Export line to its string representation.

Author(s)

Michael Lawrence

References

http://genome.ucsc.edu/goldenPath/help/customTrack.html#TRACK for the official documentation.

See Also

basicTrackLine (used for most types of tracks) and wigTrackLine (used for Wiggle tracks).

ucscTrackModes-class
Class "ucscTrackModes"

Description

A vector of view modes ("hide", "dense", "full", "pack", "squish") for each track in a UCSC view.

Objects from the Class

Objects may be created by calls of the form `ucscTrackModes(object = character(), hide = character(), dense = character(), pack = character(), squish = character(), full = character())`, where `object` should be a character vector of mode names (with its `names` attribute specifying the corresponding track names). The other parameters should contain track names that override the modes in `object`.

Slots

`.Data`: Object of class `"character"` holding the modes ("hide", "dense", "full", "pack", "squish"), with its `names` attribute holding corresponding track names.

`labels`: Object of class `"character"` holding labels (human-readable names) corresponding to each track/mode.

Extends

Class `"character"`, from data part. Class `"vector"`, by class "character", distance 2. Class `"characterORMIAME"`, by class "character", distance 2.

Methods

`tracks(object)` Gets the names of the visible tracks (those that do not have mode "hide").

`tracks(object) <- value` Sets the names of the visible tracks. Any tracks named in `value` are set to "full" if the are currently set to "hide" in this object. Any tracks not in `value` are set to "hide". All other modes are preserved.

`object[i]` Gets the track mode of the tracks indexed by `i`, which can be any type of index supported by character vector subsetting. If `i` is a character vector, it indexes first by the internal track IDs (the `names` on `.Data`) and then by the user-level track names (the `labels` slot).

`object[i] <- value` Sets the track modes indexed by `i` (in the same way as in `object[i]` above) to those specified in `value`.

Author(s)

Michael Lawrence

See Also

`ucscView` on which track view modes may be set.

ucscTrackModes-methods
Accessing UCSC track modes

Description

Generics for getting and setting UCSC track visibility modes ("hide", "dense", "full", "pack", "squish").

Methods

The following methods are defined by **rtracklayer** for **getting** the track modes through the generic `ucscTrackModes(object, ...)`.

`function(object, hide = character(), dense = character(), pack = character(), squish = character(), full = character())` Creates an instance of `ucscTrackModes` from `object`, a character vector of mode names, with the corresponding track names given in the `names` attribute. Note that `object` can be a `ucscTrackModes` instance. The other parameters are character vectors naming the tracks for each mode and overriding the modes specified by `object`.

object = "character", object = "missing" The same interface as above, except `object` defaults to an empty character vector.

object = "ucscView" Gets modes for tracks in the view.

object = "ucscSession" Gets default modes for the tracks in the session. These are the modes that will be used as the default for a newly created view.

The following methods are defined by **rtracklayer** for **setting** the track modes through the generic `ucscTrackModes(object) <- value`.

object = "ucscView", value = "ucscTrackModes" Sets the modes for the tracks in the view.

object = "ucscView", value = "character" Sets the modes from a character vector of mode names, with the corresponding track names given in the `names` attribute.

See Also

`tracks` and `tracks<-` for just getting or setting which tracks are visible (not of mode "hide").

Examples

```
# Tracks "foo" and "bar" are fully shown, "baz" is hidden
modes <- ucscTrackModes(full = c("foo", "bar"), hide = "baz")
# Update the modes to hide track "bar"
modes2 <- ucscTrackModes(modes, hide = "bar")
```

ucscTrackSet-class *Class "ucscTrackSet"*

Description

Each track in UCSC has an associated ucscTrackLine that contains metadata on the track.

Objects from the Class

Objects can be created by calls of the form new("ucscTrackSet", assayData, phenoData, featureData, experimentData, annotation, ...). See trackSet for more details.

Slots

trackLine: Object of class "ucscTrackLine" holding track metadata.

genome: Object of class "character" identifying the genome of this track, e.g. "hg18".

assayData: Object of class "AssayData" holding the experimental measurements under dataVals.

phenoData: Object of class "AnnotatedDataFrame" holding the experimental design matrix.

featureData: Object of class "AnnotatedDataFrame" holding feature information, generally including columns chrom (chromosome), start (start position), end (end position) and strand (strand on the DNA: "+", "-", or NA).

experimentData: Object of class "MIAME" holding experimental metadata.

annotation: Object of class "character" referring to the annotation dataset.

.__classVersion__: Object of class "Versions" holding version information.

Extends

Class "trackSet", directly. Class "eSet", by class "trackSet", distance 2. Class "VersionedBiobase", by class "trackSet", distance 3. Class "Versioned", by class "trackSet", distance 4.

Methods

export.bed(object, con, wig = FALSE, trackLine = !wig) Exports the track and its track line (if trackLine is TRUE) to con in the Browser Extended Display (BED) format.

export.gff(object) Exports the track and its track line (as a comment) to con in the General Feature Format (GFF).

export.ucsc(object) Exports the track and its track line to con in the UCSC meta-format.

Author(s)

Michael Lawrence

See Also

import and export for reading and writing tracks to and from connections (files), respectively.

ucscView-class *Class "ucscView"*

Description

An object representing a view of a genome in the UCSC browser.

Objects from the Class

Objects are created by calling `browserView(session, ...)` where `session` is a `ucscSession`.

Slots

hgsid: Object of class `"numeric"`, which identifies this view to UCSC.

session: Object of class `"browserSession"` to which this view belongs.

Extends

Class `"browserView"`, directly.

Methods

activeView(object) Obtains a logical indicating whether this view is the active view.

genomeSegment(object) Obtains the `genomeSegment` displayed by this view.

genomeSegment(object) <- value Sets the `genomeSegment` displayed by this view.

tracks(object) Gets the names of the visible tracks in this view.

tracks(object) <- value Sets the visible tracks by name.

ucscTrackModes(object) Obtains the `ucscTrackModes` for this view.

ucscTrackModes(object) <- value Sets the `ucscTrackModes` for this view. The `value` may be either a `ucscTrackModes` instance or a character vector that will be coerced by a call to `ucscTrackModes`.

Author(s)

Michael Lawrence

See Also

`browserView` for creating instances of this class.

`wigTrackLine-class` *Class "wigTrackLine"*

Description

A UCSC track line for Wiggle tracks.

Objects from the Class

Objects can be created by calls of the form `new("wigTrackLine", ...)` or parsed from a character vector track line with `as(text, "wigTrackLine")` or converted from a `basicTrackLine` using `as(basic, "wigTrackLine")`.

Slots

`altColor`: Object of class `"integer"` giving an alternate color, as from `col2rgb`.

`autoScale`: Object of class `"logical"` indicating whether to automatically scale to min/max of the data.

`gridDefault`: Object of class `"logical"` indicating whether a grid should be drawn.

`maxHeightPixels`: Object of class `"numeric"` of length three (max, default, min), giving the allowable range for the vertical height of the graph.

`graphType`: Object of class `"character"`, specifying the graph type, either "bar" or "points".

`viewLimits`: Object of class `"numeric"` and of length two specifying the data range (min, max) shown in the graph.

`yLineMark`: Object of class `"numeric"` giving the position of a horizontal line.

`yLineOnOff`: Object of class `"logical"` indicating whether the `yLineMark` should be visible.

`windowingFunction`: Object of class `"character"`, one of "maximum", "mean", "minimum", for removing points when the graph shrinks.

`smoothingWindow`: Object of class `"numeric"` giving the window size of a smoother to pass over the graph.

`name`: Object of class `"character"` specifying the name of the track.

`description`: Object of class `"character"` describing the track.

`visibility`: Object of class `"character"` indicating the default visible mode of the track, see `ucscTrackModes`.

`color`: Object of class `"integer"` representing the track color (as from `col2rgb`).

`priority`: Object of class `"numeric"` specifying the rank of this track.

Extends

Class `"ucscTrackLine"`, directly.

Methods

as(object, "character") Export line to its string representation.

as(object, `"basicTrackLine"`) Convert this line to a basic UCSC track line, using defaults for slots not held in common.

Author(s)

Michael Lawrence

References

Official documentation: http://genome.ucsc.edu/goldenPath/help/wiggle.html.

See Also

export.wig for exporting tracks in the Wiggle format.

Index