Package ‘TnT’

May 9, 2026

Title Interactive Visualization for Genomic Features

Version 1.35.0

Description A R interface to the TnT javascript library (https://github.com/tntvis) to provide interactive and flexible visualization of track-based genomic data.

biocViews Infrastructure, Visualization

Depends R (>= 3.4), GenomicRanges

Imports methods, stats, utils, grDevices, htmlwidgets, jsonlite, data.table, Biobase, GenomeInfoDb, IRanges, S4Vectors, knitr

Suggests GenomicFeatures, shiny, BiocManager, rmarkdown, testthat

License AGPL-3

Encoding UTF-8

LazyData true

VignetteBuilder knitr

URL `https://github.com/Marlin-Na/TnT`

BugReports `https://github.com/Marlin-Na/TnT/issues`

Collate 'JavascriptCascade.R' 'misc.R' 'tntTracks-trackData.R' 'tntTracks.R' 'tntBoard.R' 'tntTracks-compilation.R' 'tntTracks-compositeTrack.R' 'tntTracks-constructors.R' 'trackWidget.R'

RoxygenNote 7.0.2

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

git_branch devel

git_last_commit 42cb764

git_last_commit_date 2026-04-28

Repository Bioconductor 3.24

Date/Publication 2026-05-08

Author Jialin Ma [cre, aut],
Miguel Pignatelli [aut],
Toby Hocking [aut]

Maintainer Jialin Ma `<marlin-@gmx.cn>`

Contents

`composite-track` *Composite Track*

Description

Two or more arbitrary tracks can be used to create a composite track, by which different features can be shown in the same track.

Usage

```
## S4 method for signature 'TnTTrack,TnTTrack'
merge(x, y, ...)

## S4 method for signature 'TnTTrack,missing'
merge(x, y, ...)
```

Arguments

`x, y, ...`	Track constructed with track-constructors or composite track.

Value

Returns a "CompositeTrack" object.

See Also

`http://tnt.marlin.pub/articles/examples/track-CompositeTrack.html`

Examples

```
gr <- GRanges("chr1", IRanges(c(11000, 20000, 60000), width = 2000))
gpos <- GRanges("chr1", IRanges(c(12000, 21000, 61000), width = 1), value = c(1, 2, 3))
btrack <- BlockTrack(gr, label = "Block Track", tooltip = as.data.frame(gr), color = "lightblue4")
ptrack <- PinTrack(gpos, label = "Pin Track", tooltip = as.data.frame(gpos), background = "beige")

ctrack <- merge(btrack, ptrack)
## Not run:
TnTBoard(ctrack)

## End(Not run)
```

`knit_print.TnTBoard` *Printing TnTBoard in Rmarkdown*

Description

S3 method to automatically render a TnTBoard with knitr.

Usage

```
## S3 method for class 'TnTBoard'
knit_print(x, ..., options = NULL)
```

Arguments

`x`	A TnTBoard or TnTGenome object.
`..., options`	Passed to `htmlwidget:::knit_print.htmlwidget`.

Value

`htmlwidget:::knit_print.htmlwidget` invisibly returns a character vector with "browsable_html" S3 class.

References

`knit_print`

Examples

```
track <- BlockTrack(GRanges("chr12", IRanges(c(100, 400, 700), width = 100)),
                    color = c("green", "red", "blue"))
tntboard <- TnTGenome(track)
## Not run:
knitr::knit_print(tntboard)

## End(Not run)
```

`mapcol` *Scale Qualitative Values to Color*

Description

A simple util function that scales a factor to color based on the palette function.

Usage

```
mapcol(value, palette.fun = grDevices::rainbow, ...)
```

Arguments

`value`	A factor or character vector that may have n unique values.
`palette.fun`	The palette function to generate colors. For example, `terrain.colors`.
`...`	Extra arguments passed to the palette function.

Value

A character vector as colors, with the same length of `value`. Same values in `value` will have the same color.

Examples

```
mapcol(iris$Species)
```

`range,TnTBoard-method` *Range of TnTBoard*

Description

Get combined range of all tracks in a TnTBoard, used internally.

Usage

```
## S4 method for signature 'TnTBoard'
range(x, ..., with.revmap = FALSE, ignore.strand = FALSE, na.rm = FALSE)
```

Arguments

`x`	TnTBoard.
`...`, `with.revmap`, `ignore.strand`, `na.rm`	Passed to `range,GenomicRanges-method`.

Value

GRanges.

`range-TnTTrack` *Range of Tracks*

Description

Range of Tracks

Usage

```
## S4 method for signature 'RangeBasedTrack'
range(x, ..., with.revmap = FALSE, ignore.strand = FALSE, na.rm = FALSE)

## S4 method for signature 'CompositeTrack'
range(x, ..., with.revmap = FALSE, ignore.strand = FALSE, na.rm = FALSE)
```

Arguments

`x`	A TnTTrack object.
`...`, `with.revmap`, `ignore.strand`, `na.rm`	Passed to `range,GenomicRanges-method`.

Value

Returns a GRanges object.

`saveTnT` *Save a TnTBoard to an HTML file*

Description

A simple wrapper of `saveWidget`, which saves a TnTBoard/TnTGenome object to an HTML file (e.g. for sharing with others).

Usage

```
saveTnT(
  tntdef,
  file,
  selfcontained = TRUE,
  libdir = NULL,
  background = "white",
  knitrOptions = list()
)
```

Arguments

`tntdef`	A TnTBoard/TnTGenome object to save.
`file`, `selfcontained`, `libdir`, `background`, `knitrOptions`	Passed to `saveWidget`.

Value

Return NULL.

Examples

```
data <- GRanges("chr2", IRanges(c(6,9,42), width = 1),
                value = c(0.3, 0.5, 0.9))
track <- PinTrack(data, label = NULL, background = "green")
genome <- TnTGenome(list(track))
destfile <- tempfile(fileext = ".html")
destfile
saveTnT(genome, destfile)
## Not run:
utils::browseURL(destfile)

## End(Not run)
```

`seqinfo` *Seqinfo of TnTTrack and TnTBoard*

Description

Seqinfo of TnTTrack and TnTBoard

Usage

```
## S4 replacement method for signature 'RangeBasedTrack'
seqinfo(x, new2old = NULL, pruning.mode = c("error", "coarse", "fine", "tidy")) <- value

## S4 method for signature 'RangeBasedTrack'
seqinfo(x)

## S4 method for signature 'RangeBasedTrack'
seqlevelsInUse(x)

## S4 method for signature 'TnTBoard'
seqinfo(x)

## S4 method for signature 'CompositeTrack'
seqinfo(x)

## S4 replacement method for signature 'CompositeTrack'
seqinfo(x, new2old = NULL, pruning.mode = c("error", "coarse", "fine", "tidy")) <- value

## S4 method for signature 'CompositeTrack'
seqlevelsInUse(x)
```

Arguments

`x`	A TnTTrack or TnTBoard object.
`new2old, pruning.mode, value`	Passed to seqinfo method for GenomicRanges.

Value

`seqinfo` returns a SeqInfo object.

Examples

```
btrack1 <- BlockTrack(GRanges("chr1", IRanges(1, 123)))
btrack2 <- BlockTrack(GRanges("chr2", IRanges(3, 599)))
ctrack <- merge(btrack1, btrack2)
board <- TnTBoard(list(btrack1, btrack2))

seqinfo(btrack1)
seqinfo(btrack2)
seqinfo(ctrack)
seqinfo(board)
```

`strandlabel` *Display Labels with Strand*

Description

A simple util function that used internally to generate display labels of GeneTrack and TxTrack.

Usage

```
strandlabel(labels, strands)
```

Arguments

`labels`	Character vector, names of each feature.
`strands`	Factor or character vector with the same length of `labels`, can be "+", "-" or "*".

Value

A character vector that combines the labels with strand information.

Examples

```
strandlabel(c("gene1", "gene2", "gene3"), c("+", "-", "*"))
```

`TnT-shiny` *Shiny bindings for TnT*

Description

Output and render functions for using TnT within Shiny applications and interactive Rmd documents.

Usage

```
TnTOutput(outputId, width = "100%", height = "auto")

renderTnT(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

`outputId`	output variable to read from
`width, height`	Must be a valid CSS unit (like `'100%'`, `'400px'`, `'auto'`) or a number, which will be coerced to a string and have `'px'` appended.
`expr`	An expression that generates a TnTBoard/TnTGenome object.
`env`	The environment in which to evaluate `expr`.
`quoted`	Is `expr` a quoted expression (with `quote()`)? This is useful if you want to save an expression in a variable.

Value

An output or render function that enables the use of the converted htmlwidget within Shiny applications.

Examples

```
library(shiny)

ui <- fluidPage(fluidRow(
    column(width = 2, {
        "A Simple Example Here"
    }),
    column(width = 10, {
        TnTOutput("out")
    })
))
server <- function (input, output) {
    re.btrack <- reactive({
        gr <- GRanges("chr12", IRanges(100, 1000))
        BlockTrack(gr)
    })
    output$out <- renderTnT({
        TnTBoard(re.btrack())
    })
}
## Not run:
shinyApp(ui = ui, server = server)
```

```
## End(Not run)
```

`tntboard` *TnTBoard*

Description

A TnTBoard or TnTGenome object stores a list of tracks and can be automatically shown in an interactive R session or in rmarkdown output.

Usage

```
TnTBoard(
  tracklist,
  view.range = GRanges(),
  coord.range = IRanges(),
  zoom.allow = IRanges(),
  allow.drag = TRUE,
  use.tnt.genome = FALSE
)

TnTGenome(
  tracklist,
  view.range = GRanges(),
  coord.range = IRanges(),
  zoom.allow = IRanges(),
  allow.drag = TRUE
)
```

Arguments

`tracklist`	One track or a list of tracks to view.
`view.range`	Length-one GRanges object, sets the initial view range.
`coord.range`	Length-one IRanges object or length-two numeric vector, sets the coordinate limit of the board (i.e. minimum/maximum possible coordinate).
`zoom.allow`	Length-one IRanges object or length-two numeric vector, sets the minimum and maximum extent of the board (i.e. the limit when zooming in and zooming out).
`allow.drag`	Logical, whether drag should be allowed? Default TRUE.
`use.tnt.genome`	Logical, whether to add axis and location. ‘TnTGenome(...)‘ is essentially a wrapper to ‘TnTBoard(..., use.tnt.genome = TRUE)‘.

Value

Returns a TnTBoard or TnTGenome object which has printing method to be rendered as a html-widget.

Examples

```
track <- BlockTrack(GRanges("chr1", IRanges(start = c(100, 300, 500), width = c(10, 100, 200))))
## Not run:
TnTGenome(track)

## End(Not run)
```

`tooltip` *Access Track Tooltips*

Description

Access Track Tooltips

Usage

```
tooltip(x)

tooltip(x) <- value

## S4 method for signature 'TrackData'
tooltip(x)

## S4 method for signature 'TnTTrack'
tooltip(x)

## S4 replacement method for signature 'TrackData,data.frame'
tooltip(x) <- value

## S4 replacement method for signature 'TnTTrack,data.frame'
tooltip(x) <- value
```

Arguments

`x`	A TnTTrack object.
`value`	A data frame to replace, its row number should equal to length of data.

Value

`tooltip` returns a data frame.

Examples

```
gr <- GRanges("chr12", IRanges(c(6, 69), c(42, 135)), Name = c("my range 1", "my range 2"))
track <- BlockTrack(gr)
tooltip(track)
tooltip(track)$Width <- width(gr)
tooltip(track)
```

`track-constructors` *Track Constructors*

Description

Track Constructors

Usage

```
BlockTrack(
  range,
  label = deparse(substitute(range)),
  tooltip = mcols(range),
  color = "blue",
  background = NULL,
  height = 30
)

VlineTrack(
  pos,
  label = deparse(substitute(pos)),
  tooltip = mcols(pos),
  color = "green",
  background = NULL,
  height = 40
)

PinTrack(
  pos,
  value = mcols(pos)$value,
  domain = numeric(),
  label = deparse(substitute(pos)),
  tooltip = mcols(pos),
  color = "red",
  background = NULL,
  height = 40
)

LineTrack(
  pos,
  value = mcols(pos)$value,
  domain = numeric(),
  label = deparse(substitute(pos)),
  color = "yellow",
  background = NULL,
  height = 70
)

AreaTrack(
  pos,
  value = mcols(pos)$value,
```

```
  domain = numeric(),
  label = deparse(substitute(pos)),
  color = "pink",
  background = NULL,
  height = 70
)

GeneTrackFromTxDb(
  txdb,
  seqlevel = seqlevels(txdb),
  label = deparse(substitute(txdb)),
  color = "black",
  background = NULL,
  height = 100
)

FeatureTrack(
  range,
  label = deparse(substitute(range)),
  tooltip = mcols(range),
  names = base::names(range),
  color = "black",
  background = NULL,
  height = 200
)

GroupFeatureTrack(
  grl,
  label = deparse(substitute(grl)),
  tooltip = mcols(grl),
  names = base::names(grl),
  color = "black",
  background = NULL,
  height = 200
)

TxTrackFromTxDb(
  txdb,
  seqlevel = seqlevels(txdb),
  label = deparse(substitute(txdb)),
  color = "red",
  background = NULL,
  height = 300
)

TxTrackFromGRanges(
  gr,
  label = deparse(substitute(gr)),
  color = "red",
  background = NULL,
  height = 300
)
```

Arguments

`range, pos`	GRanges or IRanges object. For pos, all the width should be one.
`label`	Character, shown as label of the track on the left, could be NULL.
`tooltip`	A data frame that is parallel to range or pos.
`color`	Character vector or integer vector that sets the color of the features. It can be color names, hexadecimal string or positive integer `i` meaning `palette()[i]`, as described in `col2rgb`. It can be parallel to the data (i.e. have the same length) thus sets colors of each individual feature.
`background`	Length-one character vector that sets background of the track, could be NULL.
`height`	Length-one numeric vector that sets height of the track.
`value, domain`	‘value‘ is a numeric vector that is parallel to pos, which indicates height of features at each position for PinTrack, LineTrack and AreaTrack. ‘domain‘ is a length-two numeric vector which sets the lower and upper limit of ‘value‘ (i.e. the limit on y-axis).
`txdb, seqlevel`	The TxDb and seqlevel to extract gene or transcript from.
`names`	Character vector with the same length of data, which is used to generate display labels shown together with features when zooming in.
`grl`	For ‘GroupFeatureTrack‘ function, a GRangesList object that represents grouped ranges as data source. It is assumed that ranges in each group are on the same strand and do not overlap.
`gr`	For ‘TxTrackFromGRanges‘ function, a GRanges object that represents exons and cds as data source, and will be rendered as transcripts. Two meta-columns ("type", "tx_id") are required, "type" can be "exon" or "cds" by which ranges of "cds" will be filled with color, "tx_id" indicates the grouping.

Value

Returns an object that extends "TnTTrack" class.

See Also

You can find various examples at `http://tnt.marlin.pub/articles/examples/`, also see `composite-track` on how to create a composite track.

Examples

```
BlockTrack(range = GRanges("chr1", IRanges(199, 4000)),
           color = "green", background = "red", height = 100)
```

`trackdata` *Access Track Data*

Description

Access and modify the track data. `x$name` and `x$name <- value` are just shortcuts for `trackData(x)$name` and `trackData(x)$name <- value`, respectively.

Usage

```
trackData(x)

trackData(x) <- value

## S4 method for signature 'TnTTrack'
x$name

## S4 replacement method for signature 'TnTTrack'
x$name <- value
```

Arguments

`x`	A TnTTrack object.
`value`	Replaced value.
`name`	Passed to the inner method for track data.

Value

`trackData` on all track types except "CompositeTrack" returns an object that inherits GRanges class, which means they should behave like a GRanges. While `trackData` on "CompositeTrack" returns a list of tracks.

Examples

```
track <- BlockTrack(GRanges("chr1", IRanges(6, 54)))
class(trackData(track)) # track data of block track is an object that inherits GRanges.
#trackData(track)
ctrack <- merge(track, track)
class(trackData(ctrack)) # track data of composite track is a list of tracks
#trackData(ctrack)
```

`tracklist` *Track List in TnTBoard*

Description

The tracks of a TnTBoard are stored as a list which can be accessed or modified with these functions.

Usage

```
tracklist(tntboard)

tracklist(tntboard) <- value
```

Arguments

`tntboard`	A TnTBoard or TnTGenome object
`value`	A list of tracks

Value

`tracklist` returns a list of tracks.

Examples

```
bt <- BlockTrack(GRanges("chr21", IRanges(100, 1200)))
li.tracks <- list(bt, bt)
board <- TnTBoard(li.tracks)
tracklist(board)
## Not run:
show(board)

## End(Not run)
tracklist(board) <- list(bt)
## Not run:
show(board)

## End(Not run)
```

`trackSpec` *Track Spec*

Description

Height, background and label are common options of all tracks, use these functions to get and set them.

Usage

```
trackSpec(track, which = c("background", "height", "label"))

trackSpec(track, which = c("background", "height", "label")) <- value
```

Arguments

`track`	A TnTTrack object.
`which`	Character vector, can be "background", "height" or "label".
`value`	Value to set: background should be character, height should be numeric, label should be character. If length of `which` is bigger than one, `value` should be a list with the same length.

Value

For `trackSpec`, if length of `which` equals to one, return a scalar character or numeric, if length of `which` is bigger than one, return as a list.

Examples

```
track <- BlockTrack(GRanges("chr13", IRanges(6, 9)))
trackSpec(track, "background")
trackSpec(track, c("height", "label"))
trackSpec(track, c("height", "label")) <- list(100, "my range")
trackSpec(track, "background") <- "green"
trackSpec(track)
```

`trackWidget` *Convert a TnTBoard to htmlwidget*

Description

This function is only provided for users who are familiar with the concept of htmlwidgets-package. It explicitly convets a TnTBoard or TnTGenome object to a htmlwidget object. You do not need it in most cases.

Usage

```
trackWidget(tntdef, elementId = NULL)
```

Arguments

`tntdef`	A TnTBoard/TnTGenome object or a htmlwidget object. If it is a htmlwidget object, the function will return it as is.
`elementId`	An id for the htmlwidget (random by default).

Value

The function returns a htmlwidget object.

Examples

```
b <- TnTBoard(BlockTrack(GRanges("chr12", IRanges(1, 123))))
widget <- trackWidget(b)
class(widget)
identical(widget, trackWidget(widget))
```

Index