Package: bigWig (via r-universe)

October 31, 2026

Type Package

Title R Interface to Query UCSC BigWig Files

Version 0.99.0

Date 2026-10-26

Description R interface to query UCSC bigWig files using UCSC Genome Browser code (Kent et al.). Supports all bigWig access modes including HTTP access to remote files. Provides probe-level and base-pair resolution queries, strand-aware metaprofile computation, and mappability correction via bwMap objects.

biocViews Coverage, DataImport, Sequencing, Visualization

Imports boot

Depends R (>= 4.5.0)

Suggests knitr, rmarkdown, bookdown, testthat (>= 3.0.0)

VignetteBuilder knitr

License BSD_3_clause + file LICENSE

Encoding UTF-8

NeedsCompilation yes

SystemRequirements GNU make

URL https://github.com/guertinlab/bigWig

BugReports https://github.com/guertinlab/bigWig/issues

Config/pak/sysreqs make

Repository https://biocstaging.r-universe.dev

Date/Publication 2026-10-30 21:14:42 UTC

RemoteUrl https://github.com/BiocStaging/bigWig

RemoteRef HEAD

RemoteSha d9adcea7047e784a23a5edd12a27bc61890768ab

Contents

bigWig-package . . . 2
bed.utils . . . 3
bpQuery.bigWig . . . 4
bwMap . . . 6
foreach.bed . . . 8
load.bigWig . . . 9
mat.scale . . . 11
plots.bigWig . . . 12
print.bigWig . . . 13
probeQuery.bigWig . . . 14
profiles . . . 16
query.bigWig . . . 17
unload.bigWig . . . 18

Index **19**

`bigWig-package` *bigWig Query Library ~~ bigWig ~~*

Description

R interface to query bigWig files. C implementation relies on UCSC Genome Browser code.

Details

Copyright:	The code in src/jkweb is Copyright (c) 2000-2002 Jim Kent (c) 2003-2014 Regents of the University of California. All other code is Copyright (c) 2012-2014 Cornell University.
Package:	bigWig
Type:	Package
Version:	0.2-9
Date:	2014-11-20
License:	BSD_3_clause + file LICENSE
LazyLoad:	yes

bigWig files should be loaded with the `load.bigWig` function and then queries can be conducted with `query.bigWig` and related functions. There is support for loading both local and remote files (http,ftp). Some utility functions are also provided to build meta plots.

Author(s)

Andre L. Martins, copyright holder Cornell University. src/jkweb by Jim Kent, copyright by Jim Kent and Regents of the University of California.

Maintainer: André Luís Martins <alm253@cornell.edu>

References

Kent WJ, Zweig AS, Barber G, Hinrichs AS, Karolchik D. BigWig and BigBed: enabling browsing of large distributed data sets. *Bioinformatics*. 2010;26(17):2204-7

`bed.utils` *BED Transformation Utility Functions*

Description

Functions to transform BED data.frames.

Usage

```
center.bed(bed, upstreamWindow, downstreamWindow)

fiveprime.bed(bed, upstreamWindow, downstreamWindow)

threeprime.bed(bed, upstreamWindow, downstreamWindow)

downstream.bed(bed, downstreamWindow)

upstream.bed(bed, upstreamWindow)
```

Arguments

`bed`	the input BED data.frame.
`upstreamWindow`	integer number of bases to include upstream of the anchor point.
`downstreamWindow`	integer number of bases to include downstream of the anchor point.

Details

`center.bed`, `fiveprime.bed` and `threeprime.bed` generate a BED data.frame anchored on the center, 5' and 3' ends of the interval, respectively. The resulting intervals have size `upstreamWindow + downstreamWindow + 1`.

`downstream.bed` and `upstream.bed` generate a BED data.frame anchored on the 5' and 3' sites, respectively. The resulting intervals have size `downstreamWindow` and `upstreamWindow`, respectively.

Value

BED data.frame resulting from the requested transformation.

Author(s)

André Luís Martins

Examples

```
bed <- data.frame(chrom = "chr1", start = 1000L, end = 2000L,
                  name = "gene1", score = 0L, strand = "+",
                  stringsAsFactors = FALSE)

center.bed(bed, upstreamWindow = 500, downstreamWindow = 500)
fiveprime.bed(bed, upstreamWindow = 200, downstreamWindow = 200)
threeprime.bed(bed, upstreamWindow = 200, downstreamWindow = 200)
upstream.bed(bed, upstreamWindow = 500)
downstream.bed(bed, downstreamWindow = 500)
```

---

bpQuery.bigWig *Query bigWig Files Containing Base Pair Data*

Description

Collect values (or transformations of values) from intervals that overlap the query region(s).

Usage

```
region.bpQuery.bigWig(bw, chrom, start, end,
                      strand = NA, op = "sum", abs.value = FALSE, gap.value = 0,
                       bwMap = NULL)

bed.region.bpQuery.bigWig(bw, bed,
                       strand = NA, op = "sum", abs.value = FALSE, gap.value = 0,
                           bwMap = NULL)

bed6.region.bpQuery.bigWig(bw.plus, bw.minus, bed6,
                      op = "sum", abs.value = FALSE, gap.value = 0, bwMap = NULL)

step.bpQuery.bigWig(bw, chrom, start, end, step,
                    strand = NA, op = "sum", abs.value = FALSE, gap.value = 0,
                    bwMap = NULL, with.attributes = TRUE)

bed.step.bpQuery.bigWig(bw, bed, step,
                      strand = NA, op = "sum", abs.value = FALSE, gap.value = 0,
                       bwMap = NULL, with.attributes = TRUE, as.matrix = FALSE)

bed6.step.bpQuery.bigWig(bw.plus, bw.minus, bed6, step,
                         op = "sum", abs.value = FALSE, gap.value = 0,
                        bwMap = NULL, with.attributes = TRUE, as.matrix = FALSE,
                         follow.strand = FALSE)
```

Arguments

bw	either an R object of class 'bigWig' or a character vector containing the prefix and suffix to the path of each bigWig fragment (path = <prefix><chrom><suffix>).
bw.plus	same as 'bw', but for use with plus strand queries.
bw.minus	same as 'bw', but for use with minus strand queries.
chrom	the chromosome to which the query interval belongs to.
start	the start of the query interval (zero-based, left open coordinates like in BED files). In step.bpQuery.bigWig it can be NULL, indicating a full chromosome query.
end	the end of the query interval (zero-baed, left open coordinates like in BED files). In step.bpQuery.bigWig it can be NULL, indicating a full chromosome query.
strand	+ or - character indicating the strand of the supplied coordinates.
bed	data.frame with 'chrom', 'start' and 'end' coordinates as found in a BED3 file. Further columns are ignored.
bed6	data.frame with 'chrom', 'start', 'end' and 'strand' (column 6) information as found in a BED6 file. Other columns are ignored.
step	step size in base pairs.
op	name of the operation to apply to each query interval: "sum", "avg", "wavg", "min", "max". bigWig intervals are treated as if each position within the interval has the same value as the interval.
abs.value	logical value; if TRUE, the absolute values of each probe is used as input the interval operation.
gap.value	value assigned to query intervals that have no recorded values.
as.matrix	logical value; if TRUE, in 'BED' type queries, the result is a matrix, otherwise it's a list of vectors, one per query 'BED' entry.
with.attributes	logival value; if TRUE, results are annotated with their source coordinates and/or step size.
bwMap	R object of class 'bwMap' containing the information required to call steps or regions as unmappable.
follow.strand	logical value; if TRUE, results vectors/rows corresponding to minus strand queries are reversed.

Details

step.bpQuery.bigWig can be used to, efficiently, collect chromosome-wide values by passing NULL as the start and end arguments.

Value

Returns either a vector (non-bed functions) or either a list or a matrix (bed/bed6) functions.

Author(s)

André Luís Martins

See Also

See load.bigWig to load a bigWig.

Examples

```
bw <- load.bigWig(system.file("extdata", "test_probe_bp.bigWig", package = "bigWig"))

# single-region query
region.bpQuery.bigWig(bw, "chrB", 0, 20, op = "sum")

# BED data frame query
bed <- data.frame(chrom = "chrB", start = 0L, end = 20L)
bed.region.bpQuery.bigWig(bw, bed, op = "sum")

# step query
step.bpQuery.bigWig(bw, "chrB", 0, 20, step = 5, op = "sum")

unload.bigWig(bw)
```

---

bwMap *Query mappability bigWig files*

Description

Collect mappability information for query region(s).

Usage

```
load.bwMap(filename, read.len, read.left.edge, threshold.fraction = 0, udcDir=NULL)

unload.bwMap(bwMap)

region.bpQuery.bwMap(bwMap, chrom, start, end, strand, op = "thresh")

bed6.region.bpQuery.bwMap(bwMap, bed6, op = "thresh")

step.bpQuery.bwMap(bwMap, chrom, start, end, step, strand,
                   op = "thresh", with.attributes = TRUE)

bed6.step.bpQuery.bwMap(bwMap, bed6, step,
                      op = "thresh", with.attributes = FALSE, as.matrix = FALSE)
```

Arguments

filename	a character string giving the name of the file to load. It can be a valid URL.
read.len	integer number representing the length (in base pairs) of reads for which this mappability file was constructed.

`read.left.edge` logical value indicating if a read is represented by it's left-most edge (5' position) or it's right-most edge (3' position).

`threshold.fraction` numeric value indicating the maximum fraction of unmappable bases in a query region for that region to still be considered mappable.

`udcDir` a character string giving the name of the folder to use as a local cache when accessing remote files. Set to NULL to use the default location (/tmp/udcCache).

`bwMap` a bigWig Mappability object

`chrom` the chromosome to which the query interval belongs to.

`start` the start of the query interval (zero-based, left open coordinates like in BED files). In `step.bpQuery.bwMap` it can be NULL, indicating a full chromosome query.

`end` the end of the query interval (zero-baed, left open coordinates like in BED files). In `step.bpQuery.bwMap` it can be NULL, indicating a full chromosome query.

`strand` + or - character indicating the strand of the supplied coordinates.

`op` name of the operation to apply to each query interval: "sum", "avg", "thresh". bigWig intervals are treated as if each position within the interval has the same value as the interval.

`step` step size in base pairs.

`bed6` data.frame with 'chrom', 'start', 'end' and 'strand' (column 6) information as found in a BED6 file. Other columns are ignored.

`with.attributes` logival value; if TRUE, results are annotated with their source coordinates and/or step size.

`as.matrix` logical value; if TRUE, in 'BED' type queries, the result is a matrix, otherwise it's a list of vectors, one per query 'BED' entry.

Details

By convention, a mappability bigWig indicates mappable positions as 0 and unmappable as 1. Raw data in underlying bigWig file marks a position as unmappable if the read that starts at that position is unmappable in the plus strand (obviously dependent on read length).

`step.bpQuery.bwMap` can be used to collect chromosome-wide values by passing NULL as the `start` and `end` arguments.

Value

Returns either a vector (non-bed functions) or either a list or a matrix (bed/bed6) functions.

Author(s)

André Luís Martins

See Also

See `load.bigWig` to load a bigWig.

Examples

```
## bwMap requires a mappability bigWig file (values 0/1 per position).
## The following illustrates typical usage with such a file:

  bwmap <- load.bwMap("mappability.bigWig", read.len = 50,
                      read.left.edge = TRUE)
  print(bwmap)
  unload.bwMap(bwmap)
```

`foreach.bed` *Foreach loop for BED files (data.frames)*

Description

A quick way to apply functions to every entry in a BED file (data.frame). Similar to running `apply`, but faster since it's taylored to the structure of a BED file. Avoids a lot of boiler plate code.

Usage

```
foreach.bed(bed, func, envir = parent.frame())
```

Arguments

`bed`	Data frame corresponding to a BED file.
`func`	Function to apply to each entry in `bed`. Function must have four arguments: 'index', 'chrom', 'start', 'end' and 'strand'. Index will be a one-based integer corresponding to the current BED line. Chrom is a character string with the chromosome name. Start and end are the coordinates for the current entry (remember that BED files are zero-based left-open intervals). Strand is a character string with the entry's strand (usually '+' or '-') or NA if the `bed` has less than 6 columns.
`envir`	Environment where the function is evaluated. Default value is `parent.frame()` which corresponds to the environment where the 'foreach.bed' was called, giving access (through «-) to the local variables.

Details

Applies the supplied function to each entry in the `bed` file (actually the data.frame corresponding to the BED file).

Value

No return value.

Note

Remember to use «- when doing assignments of variables outside the loop.

Author(s)

André Luís Martins

Examples

```
#
# Silly example to compute sizes of Bed entries
#
# Obviously "sizes = bed[,3] - bed[,2]" is much faster than this.
#
sizes.bed <- function(bed) {
  N = dim(bed)[1]
  sizes = vector(mode="integer", length=N)

  foreach.bed(bed, function(i, chrom, start, end, strand) {
    sizes[i] <<- end - start
  })

  return(sizes)
}
```

`load.bigWig` *Load bigWig Files*

Description

Load UCSC bigWig files. Only the file header is loaded into memory, enough to perform subsequent queries. As with bigWig command line tools, the file can be on a remote server.

Usage

```
load.bigWig(filename, udcDir = NULL)
```

Arguments

`filename`	a character string giving the name of the file to load. It can be a valid URL.
`udcDir`	a character string giving the name of the folder to use as a local cache when accessing remote files. Set to NULL to use the default location (/tmp/udcCache).

Value

Returns an object of the bigWig class, which contains a C pointer (see note bellow) and the following accessible parts:

`basesCovered`	Number of bp covered by bigWig data.
`chromSizes`	Integer vector with number of bases per chromosome (same order as chroms vector.

chroms	Character vector with chromosome names for which there is data in the bigWig file.
isCompressed	Logical value indicating state of internal bigWig file compression.
isSwapped	Logical value indicating if binary representation is byte swapped (big vs little endian).
max	Numeric value containing the maximum value stored in the bigWig file.
mean	Numeric value containing the mean value across all covered bases in the bigWig file.
min	Numeric value containing the minimum value stored in the bigWig file.
primaryDataSize	Integer number of bytes used by data.
primaryIndexSize	Integer number of bytes used by index.
std	Standard deviation of bigWig values over covered bases.
version	Integer version number for bigWig file format.
zoomLevels	Integer number of zoom level supported by bigWig file.

Note

Return value contains a pointer to a C allocated structure stored as an attribute (handle_pte). Do not change this value directly!

Author(s)

André Luís Martins

References

Kent WJ, Zweig AS, Barber G, Hinrichs AS, Karolchik D. BigWig and BigBed: enabling browsing of large distributed data sets. *Bioinformatics*. 2010;26(17):2204-7

See Also

See `unload.bigWig` to safely unload a bigWig

Examples

```
bw_file <- system.file("extdata", "test3_chrU.bigWig", package = "bigWig")
bw <- load.bigWig(bw_file)
print(bw)
unload.bigWig(bw)
```

mat.scale *Data Matrix Scalling Utility Functions*

Description

Functions to scale data matrices.

Usage

```
rpkm.scale(mat, step, libSize, ...)

densityToOne.scale(mat, na.on.zero = TRUE, ...)

maxToOne.scale(mat, ...)

zeroToOne.scale(mat, ...)
```

Arguments

mat	the input data matrix; each row corresponds to a query region, columns to steps
step	step size in base pairs
libSize	total library mapped read count
na.on.zero	logical indicating if steps with zero counts should be marked as NA
...	extra unused arguments to facilitate composition when used in metaprofile.bigWig

Details

"RPKM stands for Reads Per Kilobase of transcript per Million mapped reads. FPKM stands for Fragments Per Kilobase of transcript per Million mapped reads. " from http://cole-trapnell-lab.github.io/cufflinks/.

Value

Data matrix scaled according to the requested method.

Author(s)

André Luís Martins

Examples

```
mat <- matrix(c(10, 20, 30, 40, 50, 60), nrow = 2)
rpkm.scale(mat, step = 10, libSize = 1e7)
densityToOne.scale(mat)
maxToOne.scale(mat)
zeroToOne.scale(mat)
```

plots.bigWig *Plot meta-profile(s)*

Description

Draw meta-profile plot with appropriate error regions.

Usage

```
## S3 method for class 'metaprofile'
plot(x, minus.profile = NULL, X0 = plus.profile$X0,
       draw.error = TRUE, col = c("red", "blue", "lightgrey", "lightgrey"),
     ylim = NULL, xlim = NULL, xlab = "Distance (bp)", ylab = plus.profile$name, ...)
```

Arguments

x	Meta-profile instance for sense strand
minus.profile	Optional meta-profile instance for reverse strand
X0	Numeric offset in base pairs (bp) to shift (subtract) "zero" position.
draw.error	Logical value indicating if profile error polygon should be drawn.
col	Vector of colors to use for, respectively, sense strand profile line, reverse strand profile line, sense strand error polygon, reverse strand error polygon.
ylim	The (y1, y2) limits of the plot.
xlim	The (x1, x2) limits of the plot.
xlab	Label for x-axis.
ylab	Label for y-axis.
...	other graphical parameters (see 'par').

Value

Called for its side effect (plotting); returns NULL invisibly.

Author(s)

André Luís Martins

See Also

See metaprofile.bigWig to generate a meta-profile from a bigWig instance.

Examples

```
bw <- load.bigWig(system.file("extdata", "test_probe_bp.bigWig", package = "bigWig"))
bed <- data.frame(chrom = "chrP", start = 0L, end = 50L)
prof <- metaprofile.bigWig(bed, bw)

  plot(prof)

unload.bigWig(bw)
```

`print.bigWig` *Print Information on bigWig Files*

Description

Print information on loaded UCSC bigWig files.

Usage

```
## S3 method for class 'bigWig'
print(x, ...)
```

Arguments

`x`	a bigWig object
`...`	other further arguments passed to or from other methods.

Details

Displays information bigWig instance in the same format as the command line tool bigWigInfo.

Value

Called for its side effect (printing); returns the bigWig object invisibly.

Author(s)

André Luís Martins

See Also

See `load.bigWig` to load a bigWig

Examples

```
bw <- load.bigWig(system.file("extdata", "test3_chrU.bigWig", package = "bigWig"))
print(bw)
unload.bigWig(bw)
```

probeQuery.bigWig *Query bigWig Files Containing Probe Data*

Description

Collect values (or transformations of values) from intervals (probes) that overlap the query region(s).

Usage

```
region.probeQuery.bigWig(bw, chrom, start, end,
                         op = "wavg", abs.value = FALSE, gap.value = NA)

bed.region.probeQuery.bigWig(bw, bed,
                             op = "wavg", abs.value = FALSE, gap.value = NA)

bed6.region.probeQuery.bigWig(bw.plus, bw.minus, bed6,
                              op = "wavg", abs.value = FALSE, gap.value = NA)

step.probeQuery.bigWig(bw, chrom, start, end, step,
                       op = "wavg", abs.value = FALSE, gap.value = NA,
                       with.attributes = TRUE)

bed.step.probeQuery.bigWig(bw, bed, step,
                           op = "wavg", abs.value = FALSE, gap.value = NA,
                           with.attributes = TRUE, as.matrix = FALSE)

bed6.step.probeQuery.bigWig(bw.plus, bw.minus, bed6, step,
                            op = "wavg", abs.value = FALSE, gap.value = NA,
                            with.attributes = TRUE, as.matrix = FALSE,
                            follow.strand = FALSE)
```

Arguments

bw	either an R object of class 'bigWig' or a character vector containing the prefix and suffix to the path of each bigWig fragment (path = <prefix><chrom><suffix>).
bw.plus	same as 'bw', but for use with plus strand queries.
bw.minus	same as 'bw', but for use with minus strand queries.
chrom	the chromosome to which the query interval belongs to.
start	the start of the query interval (zero-based, left open coordinates like in BED files). In step.probeQuery.bigWig it can be NULL, indicating a full chromosome query.
end	the end of the query interval (zero-baed, left open coordinates like in BED files). In step.probeQuery.bigWig it can be NULL, indicating a full chromosome query.

bed	data.frame with 'chrom', 'start' and 'end' coordinates as found in a BED3 file. Further columns are ignored.
bed6	data.frame with 'chrom', 'start', 'end' and 'strand' (column 6) information as found in a BED6 file. Other columns are ignored.
step	step size in base pairs.
op	name of the operation to apply to each query interval: "sum", "avg", "wavg", "min", "max". "wavg" computes the weighted average of overlapping probes, where each probe is weighted by the fraction of the (probe covered) query interval that it covers.
abs.value	logical value; if TRUE, the absolute values of each probe is used as input the interval operation.
gap.value	value assigned to query intervals that have no intercepting probes.
as.matrix	logical value; if TRUE, in 'BED' type queries, the result is a matrix, otherwise it's a list of vectors, one per query 'BED' entry.
with.attributes	logival value; if TRUE, results are annotated with their source coordinates and/or step size.
follow.strand	logical value; if TRUE, results vectors/rows corresponding to minus strand queries are reversed.

Value

Returns either a vector (non-bed functions) or either a list or a matrix (bed/bed6) functions.

Author(s)

André Luís Martins

See Also

See load.bigWig to load a bigWig.

Examples

```
bw <- load.bigWig(system.file("extdata", "test_probe_bp.bigWig", package = "bigWig"))

# single-region probe query
region.probeQuery.bigWig(bw, "chrP", 0, 50, op = "wavg")

# BED query
bed <- data.frame(chrom = "chrP", start = 0L, end = 50L)
bed.region.probeQuery.bigWig(bw, bed, op = "sum")

# step query
step.probeQuery.bigWig(bw, "chrP", 0, 50, step = 5, op = "wavg")

unload.bigWig(bw)
```

profiles *Data Profile Utility Functions*

Description

Functions to create summary profiles out of data matrices.

Usage

```
quantiles.metaprofile(mat, quantiles = c(0.875, 0.5, 0.125))

subsampled.quantiles.metaprofile(mat, quantiles = c(0.875, 0.5, 0.125), fraction = 0.10,
                                 n.samples = 1000)

confinterval.metaprofile(mat, alpha = 0.05)

bootstrapped.confinterval.metaprofile(mat, alpha = 0.05, n.samples = 300)

metaprofile.bigWig(bed, bw.plus, bw.minus = NULL, step = 1, name = "Signal",
            matrix.op = NULL, profile.op = subsampled.quantiles.metaprofile, ...)
```

Arguments

mat	the input data matrix; each row corresponds to a query region, columns to steps.
quantiles	vector of size three with top, middle and bottom quantile breaks to use in creating the summary profile.
fraction	fraction of the data (query regions) to include in each subsample.
n.samples	number of data samples to generate.
alpha	alpha value for confidence intervals (confidence level = 1 - alpha).
bed	the input BED data.frame defining the set of query regions.
bw.plus	either an R object of class 'bigWig' or a character vector containing the prefix and suffix to the path of each bigWig fragment (path = <prefix><chrom><suffix>).
bw.minus	same as 'bw.plus', but for use with minus strand queries.
step	step size in base pairs.
name	character vector describing the data.
matrix.op	matrix scalling function to apply to the data.
profile.op	summary profile funcion.
...	extra arguments to be passed to matrix.op and/or profile.op.

Value

*.profile functions return a list with 'top', 'middle' and 'bottom' profiles and the 'step' value if present as an attribute in the input matrix. 'step' item is required for the `plot.profile.bigWig` function. profile.bigWig returns a list with 'name', 'X0' (= 0), 'step' and the 'top', 'middle' and 'bottom' profiles.

Author(s)

André Luís Martins

Examples

```
bw <- load.bigWig(system.file("extdata", "test_probe_bp.bigWig", package = "bigWig"))
bed <- data.frame(chrom = "chrP", start = 0L, end = 50L)
mat <- bed.step.bpQuery.bigWig(bw, bed, step = 5, as.matrix = TRUE)
quantiles.metaprofile(mat)
confinterval.metaprofile(mat)
unload.bigWig(bw)
```

---

query.bigWig *Query bigWig Files*

Description

Collect the intersection of the query interval with the interval-value pairs contained in the bigWig file.

Usage

```
query.bigWig(bigWig, chrom, start, end, clip = TRUE)
```

Arguments

bigWig	a bigWig object to be queried.
chrom	the chromosome to which the query interval belongs to.
start	the start of the query interval (zero-based, left open coordinates like in BED files).
end	the end of the query interval (zero-baed, left open coordinates like in BED ifles).
clip	logical value; if TRUE bigWig regions are clipped to the query interval.

Value

Returns a N by 3 matrix with columns (start, end, value) where N is the number of intervals contained in the bigWig file that intersect with the query interval. Result coordinates are clipped to the query interval (unless `clip == FALSE`).

Author(s)

André Luís Martins

See Also

See `load.bigWig` to load a bigWig.

Examples

```
bw <- load.bigWig(system.file("extdata", "test3_chrU.bigWig", package = "bigWig"))
query.bigWig(bw, "chrU", 0, 50)
unload.bigWig(bw)
```

---

unload.bigWig *Unload bigWig Files*

Description

Release memory associated with the bigWig object.

Usage

```
unload.bigWig(bigWig)
```

Arguments

bigWig	a bigWig object

Value

Called for its side effect; returns NULL invisibly.

Author(s)

André Luís Martins

See Also

See load.bigWig to load a bigWig.

Examples

```
bw <- load.bigWig(system.file("extdata", "test3_chrU.bigWig", package = "bigWig"))
unload.bigWig(bw)
```

Index

∗ **bigWig**
bpQuery.bigWig, 4
foreach.bed, 8
load.bigWig, 9
plots.bigWig, 12
print.bigWig, 13
probeQuery.bigWig, 14
query.bigWig, 17
∗ **bigwig**
unload.bigWig, 18
∗ **bpQuery**
bwMap, 6
∗ **bwMap**
bwMap, 6
∗ **foreach**
foreach.bed, 8
∗ **free**
unload.bigWig, 18
∗ **load**
load.bigWig, 9
∗ **metaprofile**
plots.bigWig, 12
∗ **package**
bigWig-package, 2
∗ **plot**
plots.bigWig, 12
∗ **print**
print.bigWig, 13
∗ **query**
bpQuery.bigWig, 4
probeQuery.bigWig, 14
query.bigWig, 17
∗ **read**
load.bigWig, 9
∗ **unload**
unload.bigWig, 18

bed.region.bpQuery.bigWig
*(*bpQuery.bigWig*)*, 4
bed.region.probeQuery.bigWig
*(*probeQuery.bigWig*)*, 14
bed.step.bpQuery.bigWig
*(*bpQuery.bigWig*)*, 4
bed.step.probeQuery.bigWig
*(*probeQuery.bigWig*)*, 14
bed.utils, 3
bed6.region.bpQuery.bigWig
*(*bpQuery.bigWig*)*, 4
bed6.region.bpQuery.bwMap *(*bwMap*)*, 6
bed6.region.probeQuery.bigWig
*(*probeQuery.bigWig*)*, 14
bed6.step.bpQuery.bigWig
*(*bpQuery.bigWig*)*, 4
bed6.step.bpQuery.bwMap *(*bwMap*)*, 6
bed6.step.probeQuery.bigWig
*(*probeQuery.bigWig*)*, 14
bigWig *(*bigWig-package*)*, 2
bigWig-package, 2
bootstrapped.confinterval.metaprofile
*(*profiles*)*, 16
bpQuery.bigWig, 4
bwMap, 6

center.bed *(*bed.utils*)*, 3
confinterval.metaprofile *(*profiles*)*, 16

densityToOne.scale *(*mat.scale*)*, 11
downstream.bed *(*bed.utils*)*, 3

fiveprime.bed *(*bed.utils*)*, 3
foreach.bed, 8

load.bigWig, *6, 7*, 9, *13*, *15*, *17, 18*
load.bwMap *(*bwMap*)*, 6

mat.scale, 11
maxToOne.scale *(*mat.scale*)*, 11
metaprofile.bigWig, *12*
metaprofile.bigWig *(*profiles*)*, 16

plot.metaprofile *(*plots.bigWig*)*, 12
plots.bigWig, 12
print *(*print.bigWig*)*, 13
print.bigWig, 13
print.bwMap *(*bwMap*)*, 6
probeQuery.bigWig, 14
profiles, 16

quantiles.metaprofile *(*profiles*)*, 16
query.bigWig, 17

region.bpQuery.bigWig *(*bpQuery.bigWig*)*, 4
region.bpQuery.bwMap *(*bwMap*)*, 6
region.probeQuery.bigWig *(*probeQuery.bigWig*)*, 14
rpkm.scale *(*mat.scale*)*, 11

step.bpQuery.bigWig *(*bpQuery.bigWig*)*, 4
step.bpQuery.bwMap *(*bwMap*)*, 6
step.probeQuery.bigWig *(*probeQuery.bigWig*)*, 14
subsampled.quantiles.metaprofile *(*profiles*)*, 16

threeprime.bed *(*bed.utils*)*, 3

unload.bigWig, *10*, 18
unload.bwMap *(*bwMap*)*, 6
upstream.bed *(*bed.utils*)*, 3

zeroToOne.scale *(*mat.scale*)*, 11