Package 'PeakSegDisk'

November 19, 2019

Type Package

Title Disk-Based Constrained Change-Point Detection

Version 2019.9.27

Author Toby Dylan Hocking

Maintainer Toby Dylan Hocking <toby.hocking@r-project.org>

Description Disk-based implementation of
Functional Pruning Optimal Partitioning with up-down constraints
<arXiv:1810.00117> for single-sample peak calling
(independently for each sample and genomic problem),
can handle huge data sets (10^7 or more).

License GPL-3

Depends R (>= 2.10)

Imports data.table (>= 1.9.8)

Suggests testthat, ggplot2, future.apply, future, knitr

VignetteBuilder knitr

URL http://github.com/tdhock/PeakSegDisk

BugReports http://github.com/tdhock/PeakSegDisk/issues

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-11-18 23:10:02 UTC

R topics documented:

`ChIPreads` *Reads aligned to hg19 from two ChIP-seq experiments*

Description

These data are raw aligned reads which have been mapped to the human genome, hg19. One is sample ID McGill0004, experiment H3K36me3, chr9, chunk H3K36me3_AM_immune/8. The other is sample ID McGill0002, experiment H3K4me3, chr2, chunk H3K4me3_PGP_immune/7. The chunk ID numbers refer to parts of the McGill benchmark data set, http://members.cbio.mines-paristech.fr/~thocking/chip-seq-chunk-db/

Usage

```
data("ChIPreads")
```

Format

A data frame with 40396 observations on the following 4 variables.

`experiment` either H3K36me3 or H3K4me3

`chrom` either chr9 or chr2

`chromStart` 0-based start position of read

`chromEnd` 1-based end position of read

`count` number of times a read occured with the given chromStart/end in this sample/experiment

Details

Peak detection algorithm are typically run on a sequence of non-negative integer count data, one data point for each genomic position. These data are useful for proving that peak detection methods are robust to different sequences: (1) spatially correlated, non-independent aligned read coverage; (2) un-correlated, independent representations such as first or last read.

coef.PeakSegFPOP_df *coef PeakSegFPOP df*

Description

Create a list of data tables describing PeakSegFPOP model and data.

Usage

```
## S3 method for class 'PeakSegFPOP_df'
coef(object, ...)
```

Arguments

`object`

`...`

Value

list of data tables with named elements segments, loss, data, changes, peaks.

Author(s)

Toby Dylan Hocking

coef.PeakSegFPOP_dir *coef PeakSegFPOP dir*

Description

Compute changes and peaks to display/plot.

Usage

```
## S3 method for class 'PeakSegFPOP_dir'
coef(object, ...)
```

Arguments

`object`

`...`

Value

model list with additional named elements peaks and changes.

Author(s)

Toby Dylan Hocking

`col.name.list` *col name list*

Description

Named list of character vectors (column names of bed/bedGraph/tsv files), used to read data files, which do not contain a header / column names. Each name corresponds to a data/file type, and each value is a character vector of column names expected in that file. loss is for the coverage.bedGraph_penalty=VALUE_loss.tsv file generated by `PeakSegFPOP_file`; segments is for the coverage.bedGraph_penalty=VALUE_segments.bed generated by `PeakSegFPOP_file`; coverage is for the coverage.bedGraph file which is used as input to `PeakSegFPOP_file`.

Usage

```
"col.name.list"
```

Examples

```
library(PeakSegDisk)
r <- function(chrom, chromStart, chromEnd, coverage){
  data.frame(chrom, chromStart, chromEnd, coverage)
}
four <- rbind(
  r("chr1", 0, 10,  2),
  r("chr1", 10, 20, 10),
  r("chr1", 20, 30, 14),
  r("chr1", 30, 40, 13))
write.table(
  four, tmp <- tempfile(),
  sep="\t", row.names=FALSE, col.names=FALSE)
read.table(
  tmp, col.names=col.name.list$coverage)

pstr <- "10.5"
PeakSegFPOP_file(tmp, pstr)
outf <- function(suffix){
  paste0(tmp, "_penalty=", pstr, "_", suffix)
}
fread.first(outf("segments.bed"), col.name.list$segments)
fread.first(outf("loss.tsv"), col.name.list$loss)
```

`fread.first` *Quickly read first line*

Description

Read the first line of a text file. Useful for quickly checking if the coverage.bedGraph_penalty=VALUE_segments.bed file is consistent with the coverage.bedGraph_penalty=VALUE_loss.tsv file. (used by the `PeakSegFPOP_dir` caching mechanism)

Usage

```
fread.first(file.name, col.name.vec)
```

Arguments

`file.name`	Name of file to read.
`col.name.vec`	Character vector of column names.

Value

Data table with one row.

Author(s)

Toby Dylan Hocking

Examples

```
library(PeakSegDisk)
r <- function(chrom, chromStart, chromEnd, coverage){
  data.frame(chrom, chromStart, chromEnd, coverage)
}
four <- rbind(
  r("chr1", 0, 10,  2),
  r("chr1", 10, 20, 10),
  r("chr1", 20, 30, 14),
  r("chr1", 30, 40, 13))
write.table(
  four, tmp <- tempfile(),
  sep="\t", row.names=FALSE, col.names=FALSE)
pstr <- "10.5"
PeakSegFPOP_file(tmp, pstr)

outf <- function(suffix){
  paste0(tmp, "_penalty=", pstr, "_", suffix)
}
segments.bed <- outf("segments.bed")
first.seg.line <- fread.first(segments.bed, col.name.list$segments)
```

```
last.seg.line <- fread.last(segments.bed, col.name.list$segments)

loss.tsv <- outf("loss.tsv")
loss.row <- fread.first(loss.tsv, col.name.list$loss)

seg.bases <- first.seg.line$chromEnd - last.seg.line$chromStart
loss.row$bases == seg.bases
```

`fread.last` *Quickly read last line*

Description

Read the last line of a text file. Useful for quickly checking if the coverage.bedGraph_penalty=VALUE_segments.bed file is consistent with the coverage.bedGraph_penalty=VALUE_loss.tsv file. (used by the `PeakSegFPOP_dir` caching mechanism)

Usage

```
fread.last(file.name, col.name.vec)
```

Arguments

`file.name`	Name of file to read.
`col.name.vec`	Character vector of column names.

Value

Data table with one row.

Author(s)

Toby Dylan Hocking

Examples

```
library(PeakSegDisk)
r <- function(chrom, chromStart, chromEnd, coverage){
  data.frame(chrom, chromStart, chromEnd, coverage)
}
four <- rbind(
  r("chr1", 0, 10,  2),
  r("chr1", 10, 20, 10),
  r("chr1", 20, 30, 14),
  r("chr1", 30, 40, 13))
write.table(
  four, tmp <- tempfile(),
```

```
  sep="\t", row.names=FALSE, col.names=FALSE)
pstr <- "10.5"
PeakSegFPOP_file(tmp, pstr)

outf <- function(suffix){
  paste0(tmp, "_penalty=", pstr, "_", suffix)
}
segments.bed <- outf("segments.bed")
first.seg.line <- fread.first(segments.bed, col.name.list$segments)
last.seg.line <- fread.last(segments.bed, col.name.list$segments)

loss.tsv <- outf("loss.tsv")
loss.row <- fread.first(loss.tsv, col.name.list$loss)

seg.bases <- first.seg.line$chromEnd - last.seg.line$chromStart
loss.row$bases == seg.bases
```

Mono27ac *A small ChIP-seq data set in which peaks can be found using PeakSegFPOP*

Description

The data come from an H3K27ac ChIP-seq experiment which was aligned to the human reference genome (hg19), aligned read counts were used to produce the coverage data; looking at these data in a genome browser was used to produce the labels. ChIP-seq means Chromatin Immunoprecipitation followed by high-throughput DNA sequencing; it is an assay used to characterize genome-wide DNA-protein interactions. In this experiment the protein of interest is histone H3, with the specific modification K27ac (hence the name H3K27ac). Large counts (peaks) therefore indicate regions of the reference genome with high likelihood of interaction between DNA and that specific protein, in the specific Monocyte sample tested.

Usage

```
data("Mono27ac")
```

Format

A list of 2 data.tables: coverage has 4 columns (chrom, chromStart, chromEnd, count=number of aligned reads at each position on chrom:chromStart-chromEnd); labels has 4 columns (chrom, chromStart, chromEnd, annotation=label at chrom:chromStart-chromEnd). chrom refers to the chromosome on whcih the data were gathered (chr11), chromStart is the 0-based position before the first base of the data/label, chromEnd is the 1-based position which is the last base of the data/label. Therefore, each chromEnd on each row should be equal to the chromStart of the next row.

Source

UCI Machine Learning Repository, chipseq data set, problem directory H3K27ac-H3K4me3_TDHAM_BP/samples/Mono1_
580000 Links: https://archive.ics.uci.edu/ml/datasets/chipseq for the UCI web page;
https://github.com/tdhock/feature-learning-benchmark for a more detailed explanation.

`PeakSegFPOP_df` *PeakSeg penalized solver for data.frame*

Description

Write data frame to disk then run `PeakSegFPOP_dir` solver.

Usage

```
PeakSegFPOP_df(count.df, pen.num, base.dir = tempdir())
```

Arguments

`count.df`	data.frame with columns count, chromStart, chromEnd.
`pen.num`	Non-negative numeric scalar.
`base.dir`	base.dir/chrXX-start-end/coverage.bedGraph will be written, where chrXX is the chrom column, start is the first chromStart position, and end is the last chromEnd position.

Value

List of solver results, same as `PeakSegFPOP_dir`.

Author(s)

Toby Dylan Hocking

Examples

```
## Simulate a sequence of Poisson count data.
sim.seg <- function(seg.mean, size.mean=15){
  seg.size <- rpois(1, size.mean)
  rpois(seg.size, seg.mean)
}
set.seed(1)
seg.mean.vec <- c(1.5, 3.5, 0.5, 4.5, 2.5)
z.list <- lapply(seg.mean.vec, sim.seg)
z.rep.vec <- unlist(z.list)

## Plot the simulated data sequence.
library(ggplot2)
count.df <- data.frame(
```

```
  position=seq_along(z.rep.vec),
  count=z.rep.vec)
gg.count <- ggplot()+
  geom_point(aes(
    position, count),
    shape=1,
    data=count.df)
gg.count

## Plot the true changes.
n.segs <- length(seg.mean.vec)
seg.size.vec <- sapply(z.list, length)
seg.end.vec <- cumsum(seg.size.vec)
change.vec <- seg.end.vec[-n.segs]+0.5
change.df <- data.frame(
  changepoint=change.vec)
gg.change <- gg.count+
  geom_vline(aes(
    xintercept=changepoint),
    data=change.df)
gg.change

## Plot the run-length encoding of the same data.
z.rle.vec <- rle(z.rep.vec)
chromEnd <- cumsum(z.rle.vec$lengths)
coverage.df <- data.frame(
  chrom="chrUnknown",
  chromStart=c(0L, chromEnd[-length(chromEnd)]),
  chromEnd,
  count=z.rle.vec$values)
gg.rle <- gg.change+
  geom_segment(aes(
    chromStart+0.5, count, xend=chromEnd+0.5, yend=count),
    data=coverage.df)
gg.rle

## Fit a peak model and plot the segment means.
fit <- PeakSegDisk::PeakSegFPOP_df(coverage.df, 10.5)
gg.rle+
  geom_segment(aes(
    chromStart+0.5, mean, xend=chromEnd+0.5, yend=mean),
    color="green",
    data=fit$segments)

## Default plot method shows data as geom_step.
(gg <- plot(fit))

## Plot data as points to verify the step representation.
gg+
  geom_point(aes(
    position, count),
    color="grey",
    shape=1,
```

```
data=count.df)
```

PeakSegFPOP_dir *PeakSeg penalized solver with caching*

Description

Main function/interface for the PeakSegDisk package. Run the low-level solver, PeakSegFPOP_file, on one genomic segmentation problem directory, and read the result files into R. Actually, this function will first check if the result files are already present (and consistent), and if so, it will simply read them into R (without running PeakSegFPOP_file) – this is a caching mechanism that can save a lot of time. To run the algo on an integer vector, use PeakSegFPOP_vec; for a data.frame, use PeakSegFPOP_df. To compute the optimal model for a given number of peaks, use sequentialSearch_dir.

Usage

```
PeakSegFPOP_dir(problem.dir, penalty.param, db.file = NULL)
```

Arguments

problem.dir	Path to a directory like sampleID/problems/chrXX-start-end which contains a coverage.bedGraph file with the aligned read counts for one genomic segmentation problem. Note that the standard coverage.bedGraph file name is required; for full flexibility the user can run the algo on an arbitrarily named file via PeakSegFPOP_file (see that man page for an explanation of how storage on disk happens).
penalty.param	non-negative numeric penalty parameter (larger values for fewer peaks), or character scalar which can be interpreted as such. 0 means max peaks, Inf means no peaks.
db.file	character scalar: file for writing temporary cost function database – there will be a lot of disk writing to this file. Default NULL means to write the same disk where the input bedGraph file is stored; another option is tempfile() which may result in speedups if the input bedGraph file is on a slow network disk and the temporary storage is a fast local disk.

Details

Finds the optimal change-points using the Poisson loss and the PeakSeg constraint (changes in mean alternate between non-decreasing and non-increasing). For N data points, the functional pruning algorithm is $O(\log N)$ memory. It is $O(N \log N)$ time and disk space. It computes the exact solution to the optimization problem in vignette("Examples",package="PeakSegDisk").

Value

Named list of two data.tables:

`segments`	has one row for every segment in the optimal model,
`loss`	has one row and contains the following columns:

penalty same as input parameter

segments number of segments in optimal model

peaks number of peaks in optimal model

bases number of positions described in bedGraph file

bedGraph.lines number of lines in bedGraph file

total.loss total Poisson loss = $\sum_i m_i - z_i * \log(m_i)$ = mean.pen.cost*bases-penalty*peaks

mean.pen.cost mean penalized cost = (total.loss+penalty*peaks)/bases

equality.constraints number of adjacent segment means that have equal values in the optimal solution

mean.intervals mean number of intervals/candidate changepoints stored in optimal cost functions – useful for characterizing the computational complexity of the algorithm

max.intervals maximum number of intervals

megabytes disk usage of *.db file

seconds timing of PeakSegFPOP_file

Author(s)

Toby Dylan Hocking

Examples

```
data(Mono27ac, package="PeakSegDisk", envir=environment())
data.dir <- file.path(
  tempfile(),
  "H3K27ac-H3K4me3_TDHAM_BP",
  "samples",
  "Mono1_H3K27ac",
  "S001YW_NCMLS",
  "problems",
  "chr11-60000-580000")
dir.create(data.dir, recursive=TRUE, showWarnings=FALSE)
write.table(
  Mono27ac$coverage, file.path(data.dir, "coverage.bedGraph"),
  col.names=FALSE, row.names=FALSE, quote=FALSE, sep="\t")

## Compute one model with penalty=1952.6
(fit <- PeakSegDisk::PeakSegFPOP_dir(data.dir, 1952.6))

## Visualize that model.
ann.colors <- c(
```

```
  noPeaks="#f6f4bf",
  peakStart="#ffafaf",
  peakEnd="#ff4c4c",
  peaks="#a445ee")
library(ggplot2)
lab.min <- Mono27ac$labels[1, chromStart]
lab.max <- Mono27ac$labels[.N, chromEnd]

plist <- coef(fit)
gg <- ggplot()+
  theme_bw()+
  geom_rect(aes(
    xmin=chromStart/1e3, xmax=chromEnd/1e3,
    ymin=-Inf, ymax=Inf,
    fill=annotation),
    color="grey",
    alpha=0.5,
    data=Mono27ac$labels)+
  scale_fill_manual("label", values=ann.colors)+
  geom_step(aes(
    chromStart/1e3, count),
    color="grey50",
    data=Mono27ac$coverage)+
  geom_segment(aes(
    chromStart/1e3, mean,
    xend=chromEnd/1e3, yend=mean),
    color="green",
    size=1,
    data=plist$segments)+
  geom_vline(aes(
    xintercept=chromEnd/1e3, linetype=constraint),
    color="green",
    data=plist$changes)+
  scale_linetype_manual(
    values=c(
      inequality="dotted",
      equality="solid"))
gg

gg+
  coord_cartesian(xlim=c(lab.min, lab.max)/1e3, ylim=c(0, 10))

## Default plotting method only shows model.
(gg <- plot(fit))

## Data can be added on top of model.
gg+
  geom_step(aes(
    chromStart, count),
    color="grey50",
    data=Mono27ac$coverage)
```

PeakSegFPOP_file *PeakSegFPOP using disk storage*

Description

Run the PeakSeg Functional Pruning Optimal Partitioning algorithm, using a file on disk to store the O(N) function piece lists, each of size O(log N). This is a low-level function that just runs the algo and produces the result files (without reading them into R), so normal users are recommended to instead use PeakSegFPOP_dir, which calls this function then reads the result files into R.

Usage

```
PeakSegFPOP_file(bedGraph.file, pen.str, db.file = NULL)
```

Arguments

bedGraph.file	character scalar: tab-delimited tabular text file with four columns: chrom, chromStart, chromEnd, coverage. The algorithm creates a large temporary file in the same directory, so make sure that there is disk space available on that device.
pen.str	character scalar that can be converted to a numeric scalar via as.numeric: non-negative penalty. More penalty means fewer peaks. "0" and "Inf" are OK. Character is required rather than numeric, so that the user can reliably find the results in the output files, which are in the same directory as bedGraph.file, and named using the penalty value, e.g. coverage.bedGraph_penalty=136500650856.439_loss.tsv
db.file	character scalar: file for writing temporary cost function database – there will be a lot of disk writing to this file. Default NULL means to write the same disk where the input bedGraph file is stored; another option is tempfile() which may result in speedups if the input bedGraph file is on a slow network disk and the temporary storage is a fast local disk.

Value

A named list of input parameters, and the temporary cost function database file size in megabytes.

Author(s)

Toby Dylan Hocking

Examples

```
r <- function(chrom, chromStart, chromEnd, coverage){
  data.frame(chrom, chromStart, chromEnd, coverage)
}
four <- rbind(
  r("chr1", 0, 10,  2),
  r("chr1", 10, 20, 10),
  r("chr1", 20, 30, 14),
```

```
  r("chr1", 30, 40, 13))
dir.create(prob.dir <- tempfile())
coverage.bedGraph <- file.path(prob.dir, "coverage.bedGraph")
write.table(
  four, coverage.bedGraph,
  sep="\t", row.names=FALSE, col.names=FALSE)
pstr <- "10.5"
result.list <- PeakSegDisk::PeakSegFPOP_file(coverage.bedGraph, pstr)
dir(prob.dir)

## segments file can be read to see optimal segment means.
outf <- function(suffix){
  paste0(coverage.bedGraph, "_penalty=", pstr, suffix)
}
segments.bed <- outf("_segments.bed")
seg.df <- read.table(segments.bed)
names(seg.df) <- col.name.list$segments
seg.df

## loss file can be read to see optimal Poisson loss, etc.
loss.tsv <- outf("_loss.tsv")
loss.df <- read.table(loss.tsv)
names(loss.df) <- col.name.list$loss
loss.df
```

PeakSegFPOP_vec *PeakSeg penalized solver for integer vector*

Description

Convert integer data vector to run-length encoding, then run PeakSegFPOP_df.

Usage

```
PeakSegFPOP_vec(count.vec, pen.num)
```

Arguments

count.vec	integer vector, noisy non-negatve count data to segment.
pen.num	Non-negative numeric scalar.

Value

List of solver results, same as PeakSegFPOP_dir.

Author(s)

Toby Dylan Hocking

Examples

```
## Simulate a sequence of Poisson data.
sim.seg <- function(seg.mean, size.mean=15){
  seg.size <- rpois(1, size.mean)
  rpois(seg.size, seg.mean)
}
set.seed(1)
seg.mean.vec <- c(1.5, 3.5, 0.5, 4.5, 2.5)
z.list <- lapply(seg.mean.vec, sim.seg)
z.rep.vec <- unlist(z.list)

## Plot the simulated data.
library(ggplot2)
count.df <- data.frame(
  position=seq_along(z.rep.vec),
  count=z.rep.vec)
gg.count <- ggplot()+
  geom_point(aes(
    position, count),
    shape=1,
    data=count.df)
gg.count

## Plot the true changepoints.
n.segs <- length(seg.mean.vec)
seg.size.vec <- sapply(z.list, length)
seg.end.vec <- cumsum(seg.size.vec)
change.vec <- seg.end.vec[-n.segs]+0.5
change.df <- data.frame(
  changepoint=change.vec)
gg.change <- gg.count+
  geom_vline(aes(
    xintercept=changepoint),
    data=change.df)
gg.change

## Fit a peak model and plot it.
fit <- PeakSegDisk::PeakSegFPOP_vec(z.rep.vec, 10.5)
gg.change+
  geom_segment(aes(
    chromStart+0.5, mean, xend=chromEnd+0.5, yend=mean),
    color="green",
    data=fit$segments)

## A pathological data set.
z.slow.vec <- 1:length(z.rep.vec)
fit.slow <- PeakSegDisk::PeakSegFPOP_vec(z.slow.vec, 10.5)
rbind(fit.slow$loss, fit$loss)
```

plot.PeakSegFPOP_df *plot PeakSegFPOP df*

Description

Plot a PeakSeg model with attached data.

Usage

```
## S3 method for class 'PeakSegFPOP_df'
plot(x, ...)
```

Arguments

x

...

Value

a ggplot.

Author(s)

Toby Dylan Hocking

plot.PeakSegFPOP_dir *plot PeakSegFPOP dir*

Description

Plot a PeakSeg model with attached data.

Usage

```
## S3 method for class 'PeakSegFPOP_dir'
plot(x, ...)
```

Arguments

x

...

Value

a ggplot.

Author(s)

Toby Dylan Hocking

`sequentialSearch_dir` *Compute PeakSeg model with given number of peaks*

Description

Compute the most likely peak model with at most the number of peaks given by `peaks.int`. This function repeated calls `PeakSegFPOP_dir` with different penalty values, until either (1) it finds the `peaks.int` model, or (2) it concludes that there is no `peaks.int` model, in which case it returns the next simplest model (with fewer peaks than `peaks.int`). The first pair of penalty values (0, Inf) is run in parallel via the user-specified future plan, if the future.apply package is available.

Usage

```
sequentialSearch_dir(problem.dir, peaks.int, verbose = 0)
```

Arguments

`problem.dir`	problemID directory in which coverage.bedGraph has already been computed. If there is a labels.bed file then the number of incorrect labels will be computed in order to find the target interval of minimal error penalty values.
`peaks.int`	int: target number of peaks.
`verbose`	Print messages?

Value

Same result list from `PeakSegFPOP_dir`, with an additional component "others" describing the other models that were computed before finding the optimal model with `peaks.int` (or fewer) peaks. Additional loss columns are as follows: under=number of peaks in smaller model during binary search; over=number of peaks in larger model during binary search; iteration=number of times PeakSegFPOP has been run.

Author(s)

Toby Dylan Hocking

Examples

```
## Create simple 6 point data set discussed in supplementary
## materials. GFPOP/GPDPA computes up-down model with 2 peaks, but
## neither CDPA (PeakSegDP::cDPA) nor PDPA (jointseg)
r <- function(chrom, chromStart, chromEnd, coverage){
  data.frame(chrom, chromStart, chromEnd, coverage)
}
```

```
supp <- rbind(
  r("chr1", 0, 1,  3),
  r("chr1", 1, 2, 9),
  r("chr1", 2, 3, 18),
  r("chr1", 3, 4, 15),
  r("chr1", 4, 5, 20),
  r("chr1", 5, 6, 2)
)
data.dir <- file.path(tempfile(), "chr1-0-6")
dir.create(data.dir, recursive=TRUE)
write.table(
  supp, file.path(data.dir, "coverage.bedGraph"),
  sep="\t", row.names=FALSE, col.names=FALSE)

## register a parallel future plan to compute the first two
## penalties in parallel during the sequential search.
if(interactive() && requireNamespace("future"))future::plan("multiprocess")

## Compute optimal up-down model with 2 peaks via sequential search.
fit <- PeakSegDisk::sequentialSearch_dir(data.dir, 2L)

library(ggplot2)
ggplot()+
  theme_bw()+
  geom_point(aes(
    chromEnd, coverage),
    data=supp)+
  geom_segment(aes(
    chromStart+0.5, mean,
    xend=chromEnd+0.5, yend=mean),
    data=fit$segments,
    color="green")
```

`wc2int` *wc2int*

Description

Convert wc output to integer number of lines.

Usage

```
wc2int(wc.output)
```

Arguments

`wc.output`	Character scalar: output from wc.

Value

integer

Author(s)

Toby Dylan Hocking

`writeBedGraph` *Write bedGraph file*

Description

Write a data.frame in R to a bedGraph file on disk.

Usage

```
writeBedGraph(count.df, coverage.bedGraph)
```

Arguments

`count.df`	data.frame with four columns: chrom, chromStart, chromEnd, count.
`coverage.bedGraph`	file path where data will be saved in plain text / bedGraph format.

Value

NULL (same as write.table).

Author(s)

Toby Dylan Hocking

Examples

```
library(PeakSegDisk)
data(Mono27ac, envir=environment())
coverage.bedGraph <- file.path(
  tempfile(),
  "H3K27ac-H3K4me3_TDHAM_BP",
  "samples",
  "Mono1_H3K27ac",
  "S001YW_NCMLS",
  "problems",
  "chr11-60000-580000",
  "coverage.bedGraph")
dir.create(
  dirname(coverage.bedGraph),
  recursive=TRUE, showWarnings=FALSE)
```

```
writeBedGraph(Mono27ac$coverage, coverage.bedGraph)
fread.first(coverage.bedGraph, col.name.list$coverage)
fread.last(coverage.bedGraph, col.name.list$coverage)
```

Index