Package 'PeakSegJoint'

April 7, 2022

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

Author Toby Dylan Hocking

Version 2022.4.6

License GPL-3

URL https://github.com/tdhock/PeakSegJoint

BugReports https://github.com/tdhock/PeakSegJoint/issues

Title Joint Peak Detection in Several ChIP-Seq Samples

Description Jointly segment several ChIP-seq samples to find the peaks which are the same and different across samples. The fast approximate maximum Poisson likelihood algorithm is described in ``PeakSegJoint: fast supervised peak detection via joint segmentation of multiple count data samples'' <arXiv:1506.01286> by TD Hocking and G Bourque.

Suggests testthat, ggplot2 (>= 2.0), microbenchmark

Depends R (>= 2.14)

Imports PeakError, parallel, penaltyLearning

NeedsCompilation yes

Repository CRAN

Date/Publication 2022-04-07 19:02:29 UTC

R topics documented:

`binSum` *binSum*

Description

Compute sum of `compressed` coverage profile in bins, using fast C code.

Usage

```
binSum(compressed, bin.chromStart = 0L,
    bin.size = 1L, n.bins = 2000L,
    empty.as.zero = FALSE)
```

Arguments

`compressed`	data.frame with integer columns chromStart, chromEnd, count.
`bin.chromStart`	Base before first bin.
`bin.size`	Bin size.
`n.bins`	Number of bins.
`empty.as.zero`	Sometimes the last few bins do not have any overlapping data in `compressed`. If TRUE, set these counts to 0. If FALSE, ignore these bins (returning a data.frame with fewer than `n.bins` rows).

Value

data.frame with `n.bins` rows and columns chromStart, chromEnd, count, mean.

Author(s)

Toby Dylan Hocking

Examples

```
## bins of size 3bp.
## -1-   -3-   -5-
##    -2-   -4-
## 123456789012345 base index.
## --2---
##       --1-
##           --0-------
## Coverage profile.
profile <- data.frame(chromStart=as.integer(c(0, 6, 10)),
                      chromEnd=as.integer(c(6, 10, 10000)),
                      count=as.integer(c(2, 1, 0)))
library(PeakSegJoint)
bins <- binSum(profile,
               bin.chromStart=0L,
               bin.size=3L,
               n.bins=2000L)
library(ggplot2)
bases <- data.frame(position=1:15, base="N")
ggplot()+
  ylab("")+
  geom_text(aes(position, 0, label=base),
            data=bases)+
  geom_step(aes(chromStart+0.5, count, color=what),
            data=data.frame(profile, what="profile"),
            size=2)+
  geom_step(aes(chromStart+0.5, count, color=what),
            data=data.frame(bins, what="bin total"))+
  geom_step(aes(chromStart+0.5, mean, color=what),
            data=data.frame(bins, what="bin mean"))+
  coord_cartesian(xlim=c(0, max(bases$position)))
```

`chr7.peaks` *chr7 peaks*

Description

peaks predicted by PeakSegFPOP on chr7:50410631-53200000 in the H3K36me3_TDH_immune data set.

Usage

```
data("chr7.peaks")
```

Format

A data frame with 41 observations on the following 3 variables.

`sample.id` a character vector

`chromStart` a numeric vector

`chromEnd` a numeric vector

`clusterPeaks` *clusterPeaks*

Description

Cluster `peaks` into overlapping groups.

Usage

```
clusterPeaks(peaks)
```

Arguments

`peaks`	data.frame with columns chromStart, chromEnd.

Value

`peaks` data.frame, sorted by chromStart, with an additional column cluster.

Author(s)

Toby Dylan Hocking

Examples

```
unordered <-
  data.frame(chromStart=c(11, 12, 1, 2, 3, 6, 7),
             chromEnd=c(13, 14, 5, 8, 4, 9, 10),
             sample.id=factor(paste0("sample.", c(1, 2, 1, 2, 3, 4, 5))))
clustered <- clusterPeaks(unordered)
library(ggplot2)
bases <- geom_text(aes(position, "base", label=base),
                   data=data.frame(position=1:20, base="N"))
gg <- ggplot()+bases+ylab("")+
  scale_x_continuous(breaks=1:20)

gg+
  geom_segment(aes(chromStart+1/2, sample.id,
                   xend=chromEnd+1/2, yend=sample.id),
                   data=clustered)+
  theme_bw()+
  theme(panel.margin=grid::unit(0, "cm"))+
```

```
  facet_grid(.~cluster, labeller=label_both, scales="free", space="free")

gg+
  geom_segment(aes(chromStart+1/2, sample.id,
                   xend=chromEnd+1/2, yend=sample.id, color=factor(cluster)),
                   data=clustered)
```

`ConvertModelList` *ConvertModelList*

Description

Convert a model list from the non-repetitive format that we get from the C code to the repetitive format that is more useful for plotting.

Usage

```
ConvertModelList(model.list)
```

Arguments

`model.list`	List from PeakSegJointHeuristic(...) or PeakSegJointSeveral(...).

Value

List of data.frames: segments has 1 row for each segment mean, sample, and model size (peaks, sample.id, sample.group, chromStart, chromEnd, mean); peaks is the same kind of data.frame as segments, but with only the second/peak segments; loss has one row for each model size; modelSelection has one row for each model size that can be selected, see exactModelSelection.

Author(s)

Toby Dylan Hocking

`demo.profiles` *Demo profiles*

Description

These profiles resulted in negative means in a buggy version of the PeakSegJointFaster code.

Usage

```
data("demo.profiles")
```

Format

A data frame with 51204 observations on the following 5 variables.

`sample.id` a character vector

`sample.group` a character vector

`chromStart` a numeric vector

`chromEnd` a numeric vector

`count` a numeric vector

`featureMatrixJoint` *featureMatrixJoint*

Description

Compute the feature matrix for this joint segmentation problem.

Usage

```
featureMatrixJoint(profile.list)
```

Arguments

`profile.list`

Value

Numeric feature matrix (samples x features).

Author(s)

Toby Dylan Hocking

Examples

```
library(PeakSegJoint)
data(H3K36me3.TDH.other.chunk1, envir=environment())
lims <- c(43000000, 43200000) # left
some.counts <-
  subset(H3K36me3.TDH.other.chunk1$counts,
         lims[1] < chromEnd & chromStart < lims[2])
profile.list <- ProfileList(some.counts)
featureMatrixJoint(profile.list)
```

GeomTallRect *GeomTallRect*

Description

ggproto object for geom_tallrect

Usage

```
"GeomTallRect"
```

H3K27ac.TDH.MMM4 *Some histone ChIP-seq data*

Description

This is used in the package tests.

Usage

```
data("H3K27ac.TDH.MMM4")
```

Format

A data frame with 29905 observations on the following 5 variables.

`sample.id` a factor

`chrom` a character vector

`chromStart` a numeric vector

`chromEnd` a numeric vector

`count` a numeric vector

H3K36me3.AM.immune.chunk21 *H3K36me3_AM_immune chunk 21*

Description

Should be able to detect a peak in all 21 samples.

Usage

```
data("H3K36me3.AM.immune.chunk21")
```

Format

A profile data frame with 198142 observations.

Source

http://cbio.ensmp.fr/~thocking/chip-seq-chunk-db/ data set H3K36me3_AM_immune chunk 21: http://cbio.ensmp.fr/~thock
seq-chunk-db/H3K36me3_AM_immune/21/regions.png

H3K36me3.TDH.other.chunk1 *H3K36me3 TDH other chunk 1*

Description

8 ChIP-seq samples, some with peaks in some regions, some without.

Usage

```
data("H3K36me3.TDH.other.chunk1")
```

Format

named list of 2 data.frames: "count" contains the noisy data, "regions" contains the labels.

Source

http://cbio.ensmp.fr/~thocking/chip-seq-chunk-db/ data set H3K36me3_TDH_other chunk 1.

H3K4me3.PGP.immune.chunk2 *H3K4me3 PGP immune chunk 2*

Description

In these data the heuristic algorithm does not recover a segmentation with a peak for all samples.

Usage

```
data("H3K4me3.PGP.immune.chunk2")
```

Format

A profile data frame with 36760 observations.

Source

http://cbio.ensmp.fr/~thocking/chip-seq-chunk-db/H3K4me3_PGP_immune/2/regions.png

H3K4me3.TDH.other.chunk8 *H3K4me3_TDH_other chunk 8 subset*

Description

It should be easy to recover a joint peak in at least 8 of the 10 samples.

Usage

```
data("H3K4me3.TDH.other.chunk8")
```

Format

A profile data frame (sample.id, chromStart, chromEnd, count) with 19866 observations.

Source

http://cbio.ensmp.fr/~thocking/chip-seq-chunk-db/ data set H3K4me3_TDH_other, chunk 8. http://cbio.ensmp.fr/~thocking/chip-seq-chunk-db/H3K4me3_TDH_other/8/regions.png

multiClusterPeaks *multiClusterPeaks*

Description

Cluster peaks into overlapping groups.

Usage

```
multiClusterPeaks(peaks)
```

Arguments

peaks	data.frame with columns chromStart, chromEnd.

Value

peaks data.frame, sorted by chromStart, with an additional column cluster.

Author(s)

Toby Dylan Hocking

Examples

```
library(PeakSegJoint)
data(chr7.peaks, envir=environment())
library(ggplot2)
ggplot()+
  geom_segment(aes(
    chromStart/1e3, sample.id,
    xend=chromEnd/1e3, yend=sample.id),
    data=chr7.peaks)

clustered <- multiClusterPeaks(chr7.peaks)
clustered.list <- split(clustered, clustered$cluster)
clusters.list <- list()
for(cluster.name in names(clustered.list)){
  clusters.list[[cluster.name]] <- with(
    clustered.list[[cluster.name]], data.frame(
      cluster=cluster[1],
      clusterStart=as.integer(median(chromStart)),
      clusterEnd=as.integer(median(chromEnd))))
}
clusters <- do.call(rbind, clusters.list)
ggplot()+
  geom_segment(aes(
    chromStart/1e3, sample.id,
    color=factor(cluster),
```

```
    xend=chromEnd/1e3, yend=sample.id),
    data=clustered)+
geom_segment(aes(
  clusterStart/1e3, "clusters",
  color=factor(cluster),
  xend=clusterEnd/1e3, yend="clusters"),
  data=clusters)
```

`overflow.list` *Data set that caused integer overflow*

Description

that resulted in NaN in the flat_loss_vec.

Usage

```
data("overflow.list")
```

Format

List of 2 data frames.

`peak.at.profile.end` *peak at profile end*

Description

data set for which a peak chromEnd was identical to the data chromEnd in an initial buggy version of the solver.

Usage

```
data("peak.at.profile.end")
```

Format

a list of 27 profile data.frames.

peak1.infeasible *data where the PeakSegJoint model with 1 peak is infeasible*

Description

these data are used to test Step3.

Usage

```
data("peak1.infeasible")
```

Format

A profile data frame with 4800 observations.

PeakErrorSamples *PeakErrorSamples*

Description

Compute PeakError for several samples.

Usage

```
PeakErrorSamples(peaks,
    regions)
```

Arguments

peaks	data.frame of peaks with sample.id.
regions	data.frame of annotated region labels with sample.id.

Value

data.frame of error regions with sample.id.

Author(s)

Toby Dylan Hocking

PeakSegJointError *PeakSegJointError*

Description

Compute number of incorrect regions for every PeakSegJoint model.

Usage

```
PeakSegJointError(converted,
    problem.regions)
```

Arguments

converted	Result of ConvertModelList.
problem.regions	data.frame of annotated region labels.

Value

List of error.totals (data.frame with one row for each model size, with counts of incorrect labels), error.regions (list of data.frames with labels and error status for each model size), modelSelection (data.frame with one row for each model from exactModelSelection), target (numeric vector of length 2, lower and upper limits of target interval of log.lambda penalty values in the interval regression problem).

Author(s)

Toby Dylan Hocking

Examples

```
library(PeakSegJoint)
data(H3K36me3.TDH.other.chunk1, envir=environment())
lims <- c(43000000, 43200000) # left
some.counts <-
  subset(H3K36me3.TDH.other.chunk1$counts,
         lims[1] < chromEnd & chromStart < lims[2])
some.regions <-
  subset(H3K36me3.TDH.other.chunk1$regions,
         lims[1] < chromEnd & chromStart < lims[2])
fit <- PeakSegJointSeveral(some.counts)
converted <- ConvertModelList(fit)
error.list <- PeakSegJointError(converted, some.regions)

peaks.int.vec <- 1:3
show.peaks <- subset(converted$peaks, peaks %in% peaks.int.vec)
show.labels <- do.call(rbind, error.list$error.regions[paste(peaks.int.vec)])
```

```
if(interactive() && require(ggplot2)){
  ann.colors <-
    c(noPeaks="#f6f4bf",
      peakStart="#ffafaf",
      peakEnd="#ff4c4c",
      peaks="#a445ee")
  ggplot()+
    penaltyLearning::geom_tallrect(aes(
      xmin=chromStart/1e3, xmax=chromEnd/1e3, fill=annotation),
      alpha=0.5,
      color="grey",
      data=some.regions)+
    scale_fill_manual(values=ann.colors)+
    scale_linetype_manual("error type",
                          limits=c("correct",
                                   "false negative",
                                   "false positive"
                                   ),
                          values=c(correct=0,
                                   "false negative"=3,
                                   "false positive"=1))+
    geom_step(aes(chromStart/1e3, count),
              color="grey50",
              data=some.counts)+
    penaltyLearning::geom_tallrect(aes(
      xmin=chromStart/1e3, xmax=chromEnd/1e3, linetype=status),
      fill=NA,
      color="black",
      size=1,
      data=show.labels)+
    geom_segment(aes(chromStart/1e3, 0,
                     xend=chromEnd/1e3, yend=0),
                 size=3,
                 color="deepskyblue",
                 data=show.peaks)+
    theme_bw()+
    theme(panel.margin=grid::unit(0, "cm"))+
    facet_grid(sample.id ~ peaks, scales="free")
}
```

PeakSegJointFaster *PeakSegJointFaster*

Description

Run the PeakSegJointFaster heuristic optimization algorithm, for several bin.factor parameter values, keeping only the most likely model found. This gives an approximate solution to a multi-sample

Poisson maximum likelihood segmentation problem. Given S samples, this function computes a sequence of S+1 PeakSegJoint models, with 0, ..., S samples with an overlapping peak (maximum of one peak per sample). It also computes for G groups, the seq of G+1 models, with 0, ..., G groups with an overlapping peak.

Usage

```
PeakSegJointFaster(profiles,
    bin.factor.vec = 2:7)
```

Arguments

`profiles`	data.frame with columns sample.id, sample.group, chromStart, chromEnd, count.
`bin.factor.vec`	Size of bin pyramid. Bigger values result in slower computation.

Value

List of model fit results.

Author(s)

Toby Dylan Hocking

Examples

```
library(PeakSegJoint)
data(H3K36me3.TDH.other.chunk1, envir=environment())
some.counts <- subset(
  H3K36me3.TDH.other.chunk1$counts,
  43000000 < chromEnd &
  chromStart < 43200000)
some.counts$sample.group <- some.counts$cell.type

fit <- PeakSegJointFaster(some.counts, 2:7)

if(interactive() && require(ggplot2)){

  both <- with(fit, rbind(
    data.frame(model="sample", sample.modelSelection),
  data.frame(model="group", group.modelSelection)))
  ggplot()+
    ggtitle("model selection functions")+
    scale_size_manual(values=c(sample=2, group=1))+
    geom_segment(aes(min.log.lambda, complexity,
                     color=model, size=model,
                     xend=max.log.lambda, yend=complexity),
                 data=both)+
    xlab("log(penalty)")+
    ylab("model complexity (samples or groups with a common peak)")
```

```
}
```

PeakSegJointFasterOne *PeakSegJointFasterOne*

Description

Run the PeakSegJointFaster heuristic optimization algorithm, which gives an approximate solution to a multi-sample Poisson maximum likelihood segmentation problem. Given S samples, this function computes a sequence of S+1 PeakSegJoint models, with 0, ..., S samples with an overlapping peak (maximum of one peak per sample).

Usage

```
PeakSegJointFasterOne(profiles,
    bin.factor = 2L)
```

Arguments

profiles	List of data.frames with columns chromStart, chromEnd, count, or single data.frame with additional column sample.id.
bin.factor	Size of bin pyramid. Bigger values result in slower computation.

Value

List of model fit results, see examples to see how to use it.

Author(s)

Toby Dylan Hocking

Examples

```
library(PeakSegJoint)
data(H3K36me3.TDH.other.chunk1, envir=environment())

some.counts <- subset(
  H3K36me3.TDH.other.chunk1$counts,
  43000000 < chromEnd &
  chromStart < 43200000 &
  sample.id %in% c("McGill0023", "McGill0022", "McGill0016", "McGill0013"))

id.df <- unique(some.counts[, c("cell.type", "sample.id")])
group.list <- split(paste(id.df$sample.id), id.df$cell.type, drop=TRUE)

loss.df.list <- list()
fit.list <- list()
```

```
for(bin.factor in 2:7){
  fit.fast <- PeakSegJointFasterOne(some.counts, bin.factor)
  fit.fast$min.loss <- sum(fit.fast$peak_loss_vec)
  fit.fast$sample.loss.diff.vec <- sort(with(fit.fast, structure(
    peak_loss_vec-flat_loss_vec, names=sample.id)))
  fit.fast$group.loss.diff.vec <- sort(sapply(group.list, function(sid.vec){
    sum(fit.fast$sample.loss.diff.vec[sid.vec])
  }))
  fit.fast$sample.loss.vec <- with(fit.fast, structure(
    sum(flat_loss_vec)+cumsum(c(0, sample.loss.diff.vec)),
    names=paste0(0:length(sample.loss.diff.vec), "samples")))
  fit.fast$group.loss.vec <- with(fit.fast, structure(
    sum(flat_loss_vec)+cumsum(c(0, group.loss.diff.vec)),
    names=paste0(0:length(group.loss.diff.vec), "groups")))
  loss.df.list[[paste(bin.factor)]] <- with(fit.fast, data.frame(
    bin.factor,
    loss=sample.loss.vec,
    peaks=0:length(sample.loss.diff.vec)))
  fit.list[[paste(bin.factor)]] <- fit.fast
}
loss.df <- do.call(rbind, loss.df.list)
fit.best <- fit.list[[which.min(sapply(fit.list, "[[", "min.loss"))]]

norm.list <- list()
profile.list <- split(some.counts, some.counts$sample.id, drop=TRUE)
for(sample.id in names(profile.list)){
  one <- profile.list[[sample.id]]
  max.count <- max(one$count)
  one$count.norm <- one$count/max.count
  norm.list[[sample.id]] <- one
}
norm.df <- do.call(rbind, norm.list)

if(interactive() && require(ggplot2)){

  peaks.df.list <- list()
  for(n.samples in 1:length(fit.best$sample.loss.diff.vec)){
    peaks.df.list[[paste(n.samples)]] <- with(fit.best, data.frame(
      samples=n.samples,
      sample.id=names(sample.loss.diff.vec)[1:n.samples],
      chromStart=peak_start_end[1],
      chromEnd=peak_start_end[2]))
  }
  peaks <- do.call(rbind, peaks.df.list)
  best.peaks <- transform(peaks, y=samples*-0.1, what="peaks")
  ggplot()+
    ggtitle("model for each sample")+
    scale_color_manual(values=c(data="grey50",
                          peaks="deepskyblue",
                          bins="black", segments="green"))+
    geom_step(aes(chromStart/1e3, count.norm, color=what),
              data=data.frame(norm.df, what="data"))+
    geom_segment(aes(chromStart/1e3, y,
```

```
                  xend=chromEnd/1e3, yend=y,
                  color=what),
              size=1,
              data=best.peaks)+
  geom_text(aes(chromStart/1e3, y,
                label=paste0(samples, " sample",
                  ifelse(samples==1, "", "s"), " "),
                color=what),
            hjust=1,
            size=3,
            vjust=0.5,
            data=best.peaks)+
  theme_bw()+
  theme(panel.margin=grid::unit(0, "cm"))+
  facet_grid(sample.id ~ ., scales="free")

## same thing but for each group.
peaks.df.list <- list()
for(n.groups in 1:length(fit.best$group.loss.diff.vec)){
  group.vec <- names(fit.best$group.loss.diff.vec[1:n.groups])
  meta.df <- do.call(rbind, lapply(group.vec, function(cell.type){
    data.frame(cell.type, sample.id=group.list[[cell.type]])
  }))
  peaks.df.list[[paste(n.groups)]] <- with(fit.best, data.frame(
    groups=n.groups,
    meta.df,
    chromStart=peak_start_end[1],
    chromEnd=peak_start_end[2]))
}
peaks <- do.call(rbind, peaks.df.list)
best.peaks <- transform(peaks, y=groups*-0.1, what="peaks")
ggplot()+
  ggtitle("model for each group")+
  scale_color_manual(values=c(data="grey50",
                      peaks="deepskyblue",
                      bins="black", segments="green"))+
  geom_step(aes(chromStart/1e3, count.norm, color=what),
            data=data.frame(norm.df, what="data"))+
  geom_segment(aes(chromStart/1e3, y,
                   xend=chromEnd/1e3, yend=y,
                   color=what),
               size=1,
               data=best.peaks)+
  geom_text(aes(chromStart/1e3, y,
                label=paste0(groups, " group",
                  ifelse(groups==1, "", "s"), " "),
                color=what),
            hjust=1,
            size=3,
            vjust=0.5,
            data=best.peaks)+
  theme_bw()+
  theme(panel.margin=grid::unit(0, "cm"))+
```

```
  facet_grid(sample.id + cell.type ~ ., scales="free")

min.df <- subset(loss.df, peaks==max(peaks))
ggplot()+
  geom_line(aes(peaks, loss, group=bin.factor), data=loss.df)+
  geom_text(aes(peaks, loss, label=bin.factor), data=min.df, hjust=0)

if(require(microbenchmark)){

  N.samples.vec <- 10^seq(1, 3, by=0.5)
  max.N <- max(N.samples.vec)
  N.bases <- 10
  rmat <- function(Nr, Nc, mu){
    matrix(rpois(Nr*Nc, mu), Nr, Nc)
  }
  set.seed(1)
  big.mat <- cbind(
    rmat(max.N, N.bases, 5),
    rmat(max.N, N.bases, 10),
    rmat(max.N, N.bases, 5))
  big.df <- data.frame(
    sample.id=as.integer(row(big.mat)),
    chromStart=as.integer(col(big.mat)-1),
    chromEnd=as.integer(col(big.mat)),
    count=as.integer(big.mat))
  full.list <- ProfileList(big.df)
  time.df.list <- list()
  for(N.samples in N.samples.vec){
    partial.list <- full.list[1:N.samples]
    result <- microbenchmark(
      Heuristic=PeakSegJointHeuristicStep2(partial.list, 2L),
      Faster=PeakSegJointFasterOne(partial.list, 2L),
      times=2L)
    time.df.list[[paste(N.samples)]] <- data.frame(
      N.samples,
      result)
  }
  time.df <- do.call(rbind, time.df.list)

  ggplot()+
    geom_point(aes(
      N.samples, time/1e9, color=expr),
      data=time.df)+
    scale_x_log10()+
    scale_y_log10("seconds")

}

}
```

PeakSegJointHeuristic *PeakSegJointHeuristic*

Description

Run the PeakSegJoint fast heuristic optimization algorithm, which gives an approximate solution to a multi-sample Poisson maximum likelihood segmentation problem. Given S samples, this function computes a sequence of S+1 PeakSegJoint models, with 0, ..., S samples with an overlapping peak (maximum of one peak per sample). This solver runs steps 1-3, and Step3 checks if there are any more likely models in samples with peak locations which are the same as all the models detected in Step2. This is guaranteed as of 24 July 2015 to return a feasible segmentation (seg1 < seg2 > seg3). NB: this function is mostly for internal testing purposes (search tests/testthat/*.R for 'Heuristic('). For real data use `PeakSegJointSeveral`.

Usage

```
PeakSegJointHeuristic(profiles,
    bin.factor = 2L)
```

Arguments

`profiles`	List of data.frames with columns chromStart, chromEnd, count, or single data.frame with additional column sample.id.
`bin.factor`	Size of bin pyramid. Bigger values result in slower computation.

Value

List of model fit results, which can be passed to `ConvertModelList` for easier interpretation.

Author(s)

Toby Dylan Hocking

Examples

```
library(PeakSegJoint)
data(H3K36me3.TDH.other.chunk1, envir=environment())
lims <- c(43000000, 43200000) # left
some.counts <-
  subset(H3K36me3.TDH.other.chunk1$counts,
         lims[1] < chromEnd & chromStart < lims[2])
fit <- PeakSegJointHeuristic(some.counts)
converted <- ConvertModelList(fit)
## Normalize profile counts to [0,1].
profile.list <- split(some.counts, some.counts$sample.id)
norm.list <- list()
for(sample.id in names(profile.list)){
```

```
  one <- profile.list[[sample.id]]
  max.count <- max(one$count)
  one$count.norm <- one$count/max.count
  norm.list[[sample.id]] <- one
}
norm.df <- do.call(rbind, norm.list)
best.peaks <- transform(converted$peaks, y=peaks*-0.1, what="peaks")

if(interactive() && require(ggplot2)){

  ggplot()+
    scale_color_manual(values=c(data="grey50",
                        peaks="deepskyblue",
                        bins="black", segments="green"))+
    geom_step(aes(chromStart/1e3, count.norm, color=what),
              data=data.frame(norm.df, what="data"))+
    geom_segment(aes(chromStart/1e3, y,
                     xend=chromEnd/1e3, yend=y,
                     color=what),
                 size=1,
                 data=best.peaks)+
    geom_text(aes(chromStart/1e3, y,
                  label=paste0(peaks, " peak",
                    ifelse(peaks==1, "", "s"), " "),
                  color=what),
              hjust=1,
              size=3,
              vjust=0.5,
              data=best.peaks)+
    theme_bw()+
    theme(panel.margin=grid::unit(0, "cm"))+
    facet_grid(sample.id ~ ., scales="free")

  ggplot(converted$loss, aes(peaks, loss))+
    geom_point()+
    geom_line()

}
```

PeakSegJointHeuristicStep1 *PeakSegJointHeuristicStep1*

Description

Run the first step of the PeakSegJoint fast heuristic optimization algorithm. This is the GridSearch subroutine of the JointZoom algorithm in arXiv:1506.01286. NB: this function is only for testing the C code against the R implementation (search tests/testthat/*.R for Step1). For real data see PeakSegJointSeveral.

Usage

```
PeakSegJointHeuristicStep1(profiles,
    bin.factor = 2L)
```

Arguments

profiles	List of data.frames with columns chromStart, chromEnd, count, or single data.frame with additional column sample.id.
bin.factor	Size of bin pyramid. Bigger values result in slower computation.

Value

List of model fit results, which can be passed to ConvertModelList for easier interpretation.

Author(s)

Toby Dylan Hocking

Examples

```
library(PeakSegJoint)
library(ggplot2)
data(H3K36me3.TDH.other.chunk1, envir=environment())
lims <- c(43000000, 43200000) # left
some.counts <-
  subset(H3K36me3.TDH.other.chunk1$counts,
         lims[1] < chromEnd & chromStart < lims[2])
fit <- PeakSegJointHeuristicStep1(some.counts)
## Compute bins to show on plot as well.
profile.list <- split(some.counts, some.counts$sample.id)
bin.list <- list()
norm.list <- list()
for(sample.id in names(profile.list)){
  one <- profile.list[[sample.id]]
  max.count <- max(one$count)
  bins <- binSum(one, fit$bin_start_end[1], fit$bases_per_bin, fit$n_bins)
  stopifnot(fit$n_bins == nrow(bins))
  bins$mean <- with(bins, count/(chromEnd-chromStart))
  bins$mean.norm <- bins$mean/max.count
  stopifnot(bins$count >= 0)
  one$count.norm <- one$count/max.count
  norm.list[[sample.id]] <- one
  bin.list[[sample.id]] <- data.frame(sample.id, bins)
}
bin.df <- do.call(rbind, bin.list)
norm.df <- do.call(rbind, norm.list)
converted <- ConvertModelList(fit)
best.peaks <- transform(converted$peaks, y=peaks*-0.1, what="peaks")

if(require(ggplot2) && interactive()){
```

```
  ggplot()+
    scale_color_manual(values=c(data="grey50",
                                peaks="deepskyblue",
                                bins="black", segments="green"))+
    geom_step(aes(chromStart/1e3, count.norm, color=what),
              data=data.frame(norm.df, what="data"))+
    geom_segment(aes(chromStart/1e3, mean.norm,
                     xend=chromEnd/1e3, yend=mean.norm,
                     color=what),
                 data=data.frame(bin.df, what="bins"))+
    geom_segment(aes(chromStart/1e3, y,
                     xend=chromEnd/1e3, yend=y,
                     color=what),
                 size=1,
                 data=best.peaks)+
    geom_text(aes(chromStart/1e3, y,
                  label=paste0(peaks, " peak",
                               ifelse(peaks==1, "", "s"), " "),
                  color=what),
              hjust=1,
              size=3,
              vjust=0.5,
              data=best.peaks)+
    theme_bw()+
    theme(panel.margin=grid::unit(0, "cm"))+
    facet_grid(sample.id ~ ., scales="free")

}
```

`PeakSegJointHeuristicStep2` *PeakSegJointHeuristicStep2*

Description

Run the first and second steps of the PeakSegJoint fast heuristic optimization algorithm. Step2 the SearchNearPeak subroutine described in the JointZoom Algorithm of arXiv:1506.01286, and it is guaranteed to return feasible segmentations (seg1 < seg2 > seg3). NB: this function is only for testing the C code against the R implementation (search tests/testthat/*.R files for Step2). For real data see `PeakSegJointSeveral`.

Usage

```
PeakSegJointHeuristicStep2(profiles,
    bin.factor = 2L)
```

Arguments

profiles	List of data.frames with columns chromStart, chromEnd, count, or single data.frame with additional column sample.id.
bin.factor	Size of bin pyramid. Bigger values result in slower computation.

Value

List of model fit results, which can be passed to ConvertModelList for easier interpretation.

Author(s)

Toby Dylan Hocking

PeakSegJointSeveral *PeakSegJointSeveral*

Description

Run the PeakSegJoint heuristic segmentation algorithm with several different bin.factor values, keeping only the models with lowest Poisson loss for each peak size. This algorithm gives an approximate solution to the following multi-sample constrained maximum likelihood segmentation problem. If there are S samples total, we look for the most likely common peak in $s \in 0, ..., S$ samples. We solve the equivalent minimization problem using the Poisson loss seg.mean - count.data * log(seg.mean), from the first base to the last base of profiles. The optimization variables are the segment means, of which there can be either 1 value (no peak) or 3 values (peak) in each sample. If there are 3 segments then two constraints are applied: (1) the changes in mean must occur at the same position in each sample, and (2) the changes must be up and then down (mean1 < mean2 > mean3).

Usage

```
PeakSegJointSeveral(profiles,
    bin.factors = 2:7)
```

Arguments

profiles	data.frame or list of them from ProfileList.
bin.factors	integer vector of optimization parameters >= 2. Larger values are slower. Using more values is slower since it tells the algorithm to search more of the model space, yielding solution which is closer to the global optimum.

Value

List of model fit results, which can be passed to ConvertModelList for easier interpretation.

Author(s)

Toby Dylan Hocking

Examples

```
library(PeakSegJoint)
data(H3K4me3.TDH.other.chunk8, envir=environment())
bf.vec <- c(2, 3, 5)
fit.list <-
  list(several=PeakSegJointSeveral(H3K4me3.TDH.other.chunk8, bf.vec))
for(bf in bf.vec){
  fit.list[[paste(bf)]] <-
    PeakSegJointHeuristicStep2(H3K4me3.TDH.other.chunk8, bf)
}
loss.list <- list()
segs.by.peaks.fit <- list()
for(fit.name in names(fit.list)){
  fit <- fit.list[[fit.name]]
  loss.list[[fit.name]] <- sapply(fit$models, "[[", "loss")
  converted <- ConvertModelList(fit)
  segs.by.peaks <- with(converted, split(segments, segments$peaks))
  for(peaks in names(segs.by.peaks)){
    model.segs <- segs.by.peaks[[peaks]]
    if(is.data.frame(model.segs)){
      segs.by.peaks.fit[[peaks]][[fit.name]] <-
        data.frame(fit.name, model.segs)
    }
  }
}
do.call(rbind, loss.list)

segs1 <- do.call(rbind, segs.by.peaks.fit[["10"]])
breaks1 <- subset(segs1, min(chromStart) < chromStart)
if(interactive() && require(ggplot2)){
  ggplot()+
    ggtitle(paste("PeakSegJointSeveral runs PeakSegJointHeuristic",
                  "and keeps only the most likely model"))+
    geom_step(aes(chromStart/1e3, count),
              color="grey50",
              data=H3K4me3.TDH.other.chunk8)+
    geom_vline(aes(xintercept=chromStart/1e3),
               data=breaks1,
               color="green",
               linetype="dashed")+
    geom_segment(aes(chromStart/1e3, mean,
                     xend=chromEnd/1e3, yend=mean),
                 size=1,
                 color="green",
                 data=segs1)+
    theme_bw()+
    theme(panel.margin=grid::unit(0, "cm"))+
    facet_grid(sample.id ~ fit.name, scales="free")
}

segs.by.peaks <- list()
```

```
for(peaks in 8:10){
  segs.by.peaks[[paste(peaks)]] <-
    data.frame(peaks, segs.by.peaks.fit[[paste(peaks)]][["several"]])
}
segs <- do.call(rbind, segs.by.peaks)
breaks <- subset(segs, min(chromStart) < chromStart)
if(interactive() && require(ggplot2)){
  ggplot()+
    ggtitle("PeakSegJoint models with 8-10 peaks")+
    geom_step(aes(chromStart/1e3, count),
              color="grey50",
              data=H3K4me3.TDH.other.chunk8)+
    geom_vline(aes(xintercept=chromStart/1e3),
               data=breaks,
               color="green",
               linetype="dashed")+
    geom_segment(aes(chromStart/1e3, mean,
                     xend=chromEnd/1e3, yend=mean),
                 size=1,
                 color="green",
                 data=segs)+
    theme_bw()+
    theme(panel.margin=grid::unit(0, "cm"))+
    facet_grid(sample.id ~ peaks, scales="free")
}
```

PoissonLoss *PoissonLoss*

Description

Compute the weighted Poisson loss function, which is `seg.mean - count * log(seg.mean)`. The edge case is when the mean is zero, in which case the probability mass function takes a value of 1 when the data is 0 (and 0 otherwise). Thus the log-likelihood of a maximum likelihood segment with mean zero must be zero.

Usage

```
PoissonLoss(count, seg.mean,
    weight = 1)
```

Arguments

`count`

`seg.mean`

`weight`

Author(s)

Toby Dylan Hocking

Examples

```
PoissonLoss(1, 1)
PoissonLoss(0, 0)
PoissonLoss(1, 0)
PoissonLoss(0, 1)
```

`ProfileList` *ProfileList*

Description

Convert a data.frame or list of `profiles` to a list that can be passed to .Call("PeakSegJointHeuristic...").

Usage

```
ProfileList(profiles)
```

Arguments

`profiles`	List of data.frames with columns chromStart, chromEnd, count, or single data.frame with additional column sample.id.

Value

Named list of data.frames with columns chromStart, chromEnd, count.

Author(s)

Toby Dylan Hocking

Index