Package 'PeakSegDisk'

December 3, 2018

Type Package

Title Disk-Based Implementation of PeakSegFPOP

Version 2018.11.28

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), parallel

Suggests testthat, ggplot2, penaltyLearning

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-12-03 12:42:36 UTC

R topics documented:

col.name.list 2
fread.first 2
fread.last 3
mclapplyError 3
Mono27ac 4
PeakSegFPOP_disk 4
problem.PeakSegFPOP 7
problem.sequentialSearch 12

Index **16**

col.name.list *col name list*

Description

Named list of character vectors (column names of bed/bedGraph/tsv files).

Usage

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

fread.first *fread first*

Description

Read the first line of a text file.

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

fread.last *fread last*

Description

Read the last line of a text file.

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

mclapplyError *mclapplyError*

Description

mclapply with error checking.

Usage

```
mclapplyError(...)
```

Arguments

...	

Author(s)

Toby Dylan Hocking

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.

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).

Source

https://github.com/tdhock/feature-learning-benchmark, prob.dir= H3K27ac-H3K4me3_TDHAM_BP/samples/Mono1_H3K
580000

PeakSegFPOP_disk *PeakSegFPOP on disk*

Description

Run the PeakSeg Functional Pruning Optimal Partitioning algorithm,
using a file on disk (rather than in memory as in
PeakSegOptimal::PeakSegFPOP) 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 problem.PeakSegFPOP,
which calls this function then reads the result files into R.
Finds the optimal change-points using the Poisson loss and the
PeakSeg constraint. For N data points, the functional pruning
algorithm is O(N log N) time and disk space, and O(log N) memory.
It computes the exact

solution to the following optimization problem. Let Z be an N-vector of count data, typically the coverage, number of aligned DNA sequence reads in part of the genome (the fourth column of bedGraph.file, non-negative integers). Let W be an N-vector of positive weights (number of bases with the given amount of count/coverage, chromEnd - chromStart, third column of bedGraph.file - second column). Let penalty be a non-negative real number (larger for fewer peaks, smaller for more peaks). Find the N-vector M of real numbers (segment means) and (N-1)-vector C of change-point indicators in -1,0,1 which minimize the penalized Poisson Loss, penalty*sum_i=1^N_1 I(c_i=1) + sum_i=1^N w_i*[m_i-z_i*log(m_i)], subject to constraints: (1) the first change is up and the next change is down, etc (sum_i=1^t c_i in 0,1 for all t<N-1), and (2) the last change is down 0=sum_i=1^N-1c_i, and (3) Every zero-valued change-point variable has an equal segment mean after: c_i=0 implies m_i=m_i+1, (4) every positive-valued change-point variable may have an up change after: c_i=1 implies m_i<=m_i+1, (5) every negative-valued change-point variable may have a down change after: c_i=-1 implies m_i>=m_i+1. Note that when the equality constraints are active for non-zero change-point variables, the recovered model is not feasible for the strict inequality constraints of the PeakSeg problem, and the optimum of the PeakSeg problem is undefined.

Usage

```
PeakSegFPOP_disk(bedGraph.file, pen.str)
```

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

Value

A list of input parameters (bedGraph.file, penalty) and result files (segments, db, loss).

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)
names.list <- PeakSegFPOP_disk(tmp, "10.5")
unlink(names.list$db)
seg.df <- read.table(names.list$segments)
names(seg.df) <- col.name.list$segments
seg.df
loss.df <- read.table(names.list$loss)
names(loss.df) <- col.name.list$loss
loss.df
```

---

problem.PeakSegFPOP *problem PeakSegFPOP*

Description

Run PeakSegFPOP_disk 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_disk) – this is a caching mechanism that can save a lot of time.

Usage

```
problem.PeakSegFPOP(problem.dir, penalty.str)
```

Arguments

problem.dir	Path to a directory like sampleID/problems/problemID which contains a coverage.bedGraph file with the aligned read counts for one genomic segmentation problem.
penalty.str	character which can be interpreted as a non-negative numeric penalty parameter (larger values for fewer peaks). "0" means max peaks, "Inf" means no peaks. Needs to be a character because that is used to create files which cache/store the optimal solution.

Value

Named list of two data.tables: segments has one row for every segment in the optimal model, and loss has one row and contains the following columns. penalty=same as input, 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_disk.

Author(s)

Toby Dylan Hocking

Examples

```
library(PeakSegDisk)

data(Mono27ac, 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 <- problem.PeakSegFPOP(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]
changes <- fit$segments[, list(
  constraint=ifelse(diff(mean)==0, "equality", "inequality"),
  chromStart=chromEnd[-1],
  chromEnd=chromEnd[-1])]
gg <- ggplot()+theme_bw()
if(require(penaltyLearning)){
  gg <- gg+
    penaltyLearning::geom_tallrect(aes(
      xmin=chromStart, xmax=chromEnd,
```

```
      fill=annotation),
      color="grey",
      data=Mono27ac$labels)+
    scale_fill_manual("label", values=ann.colors)
}
gg <- gg+
  geom_step(aes(
    chromStart, count),
    color="grey50",
    data=Mono27ac$coverage)+
  geom_segment(aes(
    chromStart, mean,
    xend=chromEnd, yend=mean),
    color="green",
    size=1,
    data=fit$segments)+
  geom_segment(aes(
```

```
    chromStart, mean,
    xend=chromEnd, yend=mean),
    color="green",
    size=1,
    data=fit$segments)+
  geom_vline(aes(
    xintercept=chromEnd, linetype=constraint),
    color="green",
    data=changes)+
  scale_linetype_manual(values=c(inequality="dotted", equality="solid"))
gg

gg+
  coord_cartesian(xlim=c(lab.min, lab.max))
```

---

problem.sequentialSearch *problem sequentialSearch*

Description

Compute the most likely peak model with at most the number of peaks given by peaks.int. This function repeated calls problem.PeakSegFPOP 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).

Usage

```
problem.sequentialSearch(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 problem.PeakSegFPOP, 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)
```

```
## Compute optimal up-down model with 2 peaks via sequential search.
fit <- PeakSegDisk::problem.sequentialSearch(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")
```

Index

∗Topic **datasets**
Mono27ac, 4

col.name.list, 2

fread.first, 2
fread.last, 3

mclapplyError, 3
Mono27ac, 4

PeakSegFPOP_disk, 4
problem.PeakSegFPOP, 7
problem.sequentialSearch, 12