Package ‘PeakSegOptimal’

May 25, 2018

Maintainer  Toby Dylan Hocking <toby.hocking@r-project.org>
Author  Toby Dylan Hocking
Version  2018.05.25
License  GPL-3
Title  Optimal Segmentation Subject to Up-Down Constraints
Description  Computes optimal changepoint models using the
Poisson likelihood for non-negative count data,
subject to the PeakSeg constraint:
the first change must be up, second change down, third change up, etc.
For more info about the models and algorithms,
read ”A log-linear time algorithm for constrained changepoint detection”
<arXiv:1703.03352> by TD Hocking et al.
Depends  R (>= 2.10)
Imports  penaltyLearning
Suggests  PeakSegDP (>= 2016.08.06), ggplot2, testthat, data.table (>=
1.9.8)
NeedsCompilation  yes
Repository  CRAN
Date/Publication  2018-05-25 15:50:28 UTC

R topics documented:

H3K4me3_PGPImmune_chunk24 ................................. 2
H3K4me3_XJ_immune_chunk1 ............................... 2
oracleModelComplexity ................................. 3
PeakSegFPOP ............................................. 3
PeakSegFPOPhchrom ..................................... 5
PeakSegPDPA ............................................. 7
PeakSegPDPAchrom ..................................... 9
PeakSegPDPAInf ......................................... 11
PoissonLoss ............................................. 12

Index 14
H3K4me3_XJ_immune_chunk1

H3K4me3_PGP_immune_chunk24

\textit{H3K4me3\_PGP\_immune\_chunk\ 24}

\underline{Description}

This data set revealed a bug in the PeakSegPDPA solver: at one point it recovered a less likely model than PeakSegDP for McGill0091, 13 segments.

\underline{Usage}

\begin{verbatim}
data("H3K4me3\_PGP\_immune\_chunk24")
\end{verbatim}

\underline{Format}

A data frame with 66713 observations on the following 5 variables.

\underline{Source}

http://cbio.mines-paristech.fr/~thocking/chip-seq-chunk-db/

H3K4me3_XJ_immune_chunk1

\textit{H3K4me3\_XJ\_immune\_chunk\ 1}

\underline{Description}

Some test data sets for the Poisson PeakSeg Segment Neighborhood Algo.

\underline{Usage}

\begin{verbatim}
data("H3K4me3\_XJ\_immune\_chunk1")
\end{verbatim}

\underline{Format}

A data frame with 18027 observations on 5 variables: cell.type, sample.id, chromStart, chromEnd, coverage.
oracleModelComplexity

Description

Compute Oracle model complexity from paper of Cleynen et al.

Usage

oracleModelComplexity(bases, segments)

Arguments

bases
segments

Value

numeric vector of model complexity values.

Author(s)

Toby Dylan Hocking

PeakSegFPOP

Description

Find 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 memory. It recovers the exact solution to the following optimization problem. Let Z be an N-vector of count data (count.vec, non-negative integers), let W be an N-vector of positive weights (weight.vec), and let penalty be a non-negative real number. 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}^{N} c_i in 0,1 for all t<N-1), and (2) the last change is down 0=\sum_{i=1}^{N-1} c_i, and (3) Every zero-valued change-point variable has an equal segment mean after: \hat{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(count.vec, weight.vec = rep(1, length(count.vec)),
penalty = NULL)

Arguments

count.vec integer vector of length >= 3: non-negative count data to segment.
weight.vec numeric vector (same length as count.vec) of positive weights.
penalty non-negative numeric scalar: penalty parameter (smaller for more peaks, larger
for fewer peaks).

Value

List of model parameters. count.vec, weight.vec, n.data, penalty (input parameters),
cost.mat (optimal Poisson loss), ends.vec (optimal position of segment ends, 1-indexed),
mean.vec (optimal segment means), intervals.mat (number of intervals stored by the functional
pruning algorithm). To recover the solution in terms of (M,C) variables, see the example.

Author(s)

Toby Dylan Hocking

Examples

## Use the algo to compute the solution list.
library(PeakSegOptimal)
data("H3K4me3_XJ_immune_chunk1", envir=environment())
by.sample <-
  split(H3K4me3_XJ_immune_chunk1, H3K4me3_XJ_immune_chunk1$sample.id)
n.data.vec <- sapply(by.sample, nrow)
one <- by.sample[[1]]
count.vec <- one$coverage
weight.vec <- with(one, chromEnd-chromStart)
penalty <- 1000
fit <- PeakSegFPOP(count.vec, weight.vec, penalty)

## Recover the solution in terms of (M,C) variables.
change.vec <- with(fit, rev(ends.vec[ends.vec>0]))
change.sign.vec <- rep(c(1, -1), length(change.vec)/2)
end.vec <- c(change.vec, fit$n.data)
start.vec <- c(1, change.vec+1)
length.vec <- end.vec-start.vec+1
mean.vec <- rev(fit$mean.vec[1:(length(change.vec)+1)])
M.vec <- rep(mean.vec, length.vec)
C.vec <- rep(0, fit$n.data-1)
C.vec[change.vec] <- change.sign.vec
diff.vec <- diff(M.vec)
data.frame(
  change=c(C.vec, NA),
peaksegfpopchrom

### Description

Find the optimal change-points using the Poisson loss and the PeakSeg constraint. This function is a user-friendly interface to the PeakSegFPOP function.

### Usage

```r
peaksegfpopchrom(count.df, penalty = NULL)
```

### Arguments

- **count.df**: data.frame with columns `count`, `chromStart`, `chromEnd`.
- **penalty**: non-negative numeric scalar: penalty parameter (smaller for more peaks, larger for fewer peaks).

### Value

List of data.frames: segments can be used for plotting the segmentation model, loss summarizes the penalized PoissonLoss and feasibility of the computed model.
Author(s)

Toby Dylan Hocking

Examples

library(PeakSegOptimal)
data("H3K4me3_XJ_immune_chunk1", envir=environment())
sample.id <- "McGill0106"
H3K4me3_XJ_immune_chunk1$count <- H3K4me3_XJ_immune_chunk1$coverage
by.sample <-
  split(H3K4me3_XJ_immune_chunk1, H3K4me3_XJ_immune_chunk1$sample.id)
one.sample <- by.sample[[sample.id]]

penalty.constant <- 3000
fpop.fit <- PeakSegFPOPCchrom(one.sample, penalty.constant)
fpop.breaks <- subset(fpop.fit$segments, 1 < first)
library(ggplot2)
ggplot()+
  theme_bw()+
  theme(panel.margin=grid::unit(0, "lines"))+
  geom_step(aes(chromStart/1e3, coverage),
            data=one.sample, color="grey")+
  geom_segment(aes(chromStart/1e3, mean,
                  xend=chromEnd/1e3, yend=mean),
              color="green",
              data=fpop.fit$segments)+
  geom_vline(aes(xintercept=chromStart/1e3),
              color="green",
              linetype="dashed",
              data=fpop.breaks)

max.peaks <- as.integer(fpop.fit$segments$peaks[1]+1)
pdpa.fit <- PeakSegPDPAchrom(one.sample, max.peaks)
models <- pdpa.fit$modelSelection.decreasing
models$PoissonLoss <- pdpa.fit$loss[paste(models$peaks, "PoissonLoss")]
models$algorithm <- "PDPA"
fpop.fit$loss$algorithm <- "FPOP"
ggplot()+
  geom_abline(aes(slope=peaks, intercept=PoissonLoss, color=peaks),
              data=pdpa.fit$loss)+
  geom_label(aes(0, PoissonLoss, color=peaks,
                 label=paste0("s=", peaks, " "),
                 hjust=1,
                 vjust=0,
                 data=pdpa.fit$loss)+
  geom_point(aes(penalty.constant, penalized.loss, fill=algorithm),
             shape=21,
             data=fpop.fit$loss)+
  geom_point(aes(min.lambda, min.lambda=peaks + PoissonLoss,
                 fill=algorithm),
             shape=21,
Description

Find the optimal change-points using the Poisson loss and the PeakSeg constraint. For N data points and S segments, the functional pruning algorithm is O(S*NlogN) space and O(S*NlogN) time. It recovers the exact solution to the following optimization problem. Let Z be an N-vector of count data (count.vec, non-negative integers) and let W be an N-vector of positive weights (weight.vec). 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 Poisson Loss, sum_{i=1}^{N} w_i*[m_i-z_i*log(m_i)], subject to constraints: (1) there are exactly S-1 non-zero elements of C, and (2) 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), 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

PeakSegPDPA(count.vec, weight.vec = rep(1, length(count.vec)), max.segments = NULL)

Arguments

- count.vec: integer vector of count data.
- weight.vec: numeric vector (same length as count.vec) of positive weights.
- max.segments: integer of length 1: maximum number of segments (must be >= 2).

Value

List of model parameters. count.vec, weight.vec, n.data, max.segments (input parameters), cost.mat (optimal Poisson loss), ends.mat (optimal position of segment ends, 1-indexed), mean.mat (optimal segment means), intervals.mat (number of intervals stored by the functional pruning algorithm). To recover the solution in terms of (M,C) variables, see the example.

Author(s)

Toby Dylan Hocking
Examples

```r
## Use the algo to compute the solution list.
data("H3K4me3_XJ_immune_chunk1", envir=environment())
by.sample <-
  split(H3K4me3_XJ_immune_chunk1, H3K4me3_XJ_immune_chunk1$sample.id)
n.data.vec <- sapply(by.sample, nrow)
one <- by.sample[[1]]
count.vec <- one$count
weight.vec <- with(one, chromEnd-chromStart)
max.segments <- 19L
fit <- PeakSegPDPA(count.vec, weight.vec, max.segments)

## Recover the solution in terms of \(M, C\) variables.
n.segs <- 11L
change.vec <- fit$ends.mat[n.segs, 2:n.segs]
change.sign.vec <- rep(c(1, -1), length(change.vec)/2)
end.vec <- c(change.vec, fit$n.data)
start.vec <- c(1, change.vec+1)
length.vec <- end.vec-start.vec+1
mean.vec <- fit$mean.mat[n.segs, 1:n.segs]
M.vec <- rep(mean.vec, length.vec)
C.vec <- rep(0, fit$n.data-1)
C.vec[change.vec] <- change.sign.vec
diff.vec <- diff(M.vec)
data.frame(
  change=c(C.vec, NA),
  mean=M.vec,
  equality.constraint.active=c(sign(diff.vec) != C.vec, NA))
stopifnot(cumsum(sign(C.vec)) %in% c(0, 1))

## Compute Poisson loss of M.vec and compare to the value reported
## in the fit solution list.
rbind(
  PoissonLoss(count.vec, M.vec, weight.vec),
  fit$cost.mat[n.segs, fit$n.data])

## Plot the number of intervals stored by the algorithm.
PDPA.intervals <- data.frame(
  segments=as.numeric(row(fit$intervals.mat)),
  data=as.numeric(col(fit$intervals.mat)),
  intervals=as.numeric(fit$intervals.mat))
some.intervals <- subset(PDPA.intervals, segments<data & 1<segments)
library(ggplot2)
ggplot()+
  theme_bw()+
  theme(panel.margin=grid::unit(0, "lines")+)
  facet_grid(segments ~ .)+
  geom_line(aes(data, intervals), data=some.intervals)+
  scale_y_continuous(
    "intervals stored by the\n    constrained optimal segmentation algorithm",
    breaks=c(20, 40))
```
Description

Find the optimal change-points using the Poisson loss and the PeakSeg constraint. This function is a user-friendly interface to the PeakSegPDPA function.

Usage

PeakSegPDPAchrom(count.df, max.peaks = NULL)

Arguments

count.df data.frame with columns count, chromStart, chromEnd.
max.peaks integer > 0: maximum number of peaks.

Value

List of data.frames: segments can be used for plotting the segmentation model, loss describes model loss and feasibility, modelSelection.feasible describes the set of all linear penalty (lambda) values which can be used to select the feasible models, modelSelection.decreasing selects from all models that decrease the Poisson loss relative to simpler models (same as PeakSegFPOP).

Author(s)

Toby Dylan Hocking

Examples

## samples for which pdpa recovers a more likely model, but it is ## not feasible for the PeakSeg problem (some segment means are ## equal).
sample.id <- "McGill0322"
sample.id <- "McGill0079"
sample.id <- "McGill0106"
n.peaks <- 3
library(PeakSegOptimal)
data("H3K4me3_XJ_immune_chunk1", envir=environment())
H3K4me3_XJ_immune_chunk1$count <- H3K4me3_XJ_immune_chunk1$coverage
by.sample <-
  split(H3K4me3_XJ_immune_chunk1$H3K4me3_XJ_immune_chunk1$sample.id)
one.sample <- by.sample[[sample.id]]
pdpap.fit <- PeakSegPDPAchrom(one.sample, 9L)
pdpap.segs <- subset(pdpap.fit$segments, n.peaks == peaks)
both.segs.list <- list(pdpap=data.frame(pdpap.segs, algorithm="PDPA"))
pdpa.breaks <- subset(pdpa.segs, 1 < first)
pdpa.breaks$feasible <- ifelse(
  diff(pdpa.segs$mean) == 0, "infeasible", "feasible")
both.breaks.list <- list(pdpa=data.frame(pdpa.breaks, algorithm="PDPA"))
if(require(PeakSegDP)){
  dp.fit <- PeakSegDP(one.sample, 9L)
  dp.segs <- subset(dp.fit$segments, n.peaks == peaks)
  dp.breaks <- subset(dp.segs, 1 < first)
  dp.breaks$feasible <- "feasible"
  both.segs.list$dp <- data.frame(dp.segs, algorithm="cDPA")
  both.breaks.list$dp <- data.frame(dp.breaks, algorithm="cDPA")
}
both.segs <- do.call(rbind, both.segs.list)
both.breaks <- do.call(rbind, both.breaks.list)
library(ggplot2)
ggplot()+
  theme_bw()+
  theme(panel.margin=grid::unit(0, "lines"))+
  facet_grid(algorithm ~ , scales="free")+
  geom_step(aes(chromStart/1e3, coverage),
            data=one.sample, color="grey")+
  geom_segment(aes(chromStart/1e3, mean,
                   chromEnd/1e3, yend=mean),
               color="green",
               data=both.segs)+
  scale_linetype_manual(values=c("feasible"="dotted", "infeasible"="solid"))+
  geom_vline(aes(xintercept=chromStart/1e3, linetype=feasible),
             color="green",
             data=both.breaks)

## samples for which pdpa recovers some feasible models that the
## heuristic dp does not.
sample.id.vec <- c(
sample.id <- sample.id.vec[4]
one.sample <- by.sample[[sample.id]]
pdpa.fit <- PeakSegPDPAchrom(one.sample, 9L)
gg.loss <- ggplot()+
  scale_color_manual(values=c("TRUE"="black", "FALSE"="red"))+
  scale_size_manual(values=c(cDPA=1.5, PDPA=3))+
  scale_fill_manual(values=c(cDPA="white", PDPA="black"))+
  guides(color=guide_legend(override.aes(fill="black")))+
  geom_point(aes(peaks, PoissonLoss,
                 size=algorithm, fill=algorithm, color=feasible),
             shape=21,
             data=data.frame(pdpa.fit$loss, algorithm="PDPA"))
if(require(PeakSegDP)){
  dp.fit <- PeakSegDP(one.sample, 9L)
gg.loss <- gg.loss+
  geom_point(aes(peaks, error,
                 size=algorithm, fill=algorithm),
             shape=21,
Description

Find the optimal change-points using the Poisson loss and the PeakSeg constraint. This function is an interface to the C++ code which always uses -Inf for the first interval’s lower limit and Inf for the last interval’s upper limit – it is for testing the number of intervals between the two implementations.

Usage

```
PeakSegPDPAInf(count.vec, weight.vec = rep(1, length(count.vec)),
              max.segments = NULL)
```

Arguments

count.vec integer vector of count data.
weight.vec numeric vector (same length as count.vec) of positive weights.
max.segments integer of length 1: maximum number of segments (must be >= 2).

Value

List of model parameters. count.vec, weight.vec, n.data, max.segments (input parameters), cost.mat (optimal Poisson loss), ends.mat (optimal position of segment ends, 1-indexed), mean.mat (optimal segment means), intervals.mat (number of intervals stored by the functional pruning algorithm). To recover the solution in terms of (M,C) variables, see the example.

Author(s)

Toby Dylan Hocking
Examples

```r
## Use the algo to compute the solution list.
library(PeakSegOptimal)
data("H3K4me3_XJ_immune_chunk1", envir=environment())
by.sample <-
  split(H3K4me3_XJ_immune_chunk1$sample.id)
nt.data.vec <- sapply(by.sample, nrow)
one <- by.sample[[1]]
count.vec <- one$coverage
weight.vec <- with(one, chromEnd-chromStart)
max.segments <- 19L

library(data.table)
ic.list <- list()
for(fun.name in c("PeakSegPDPA", "PeakSegPDPAInf")){
  fun <- get(fun.name)
  fit <- fun(count.vec, weight.vec, max.segments)
  ic.list[[fun.name]] <- data.table(
    fun.name,
    segments=as.numeric(row(fit$intervals.mat)),
    data=as.numeric(col(fit$intervals.mat)),
    cost=as.numeric(fit$cost.mat),
    intervals=as.numeric(fit$intervals.mat))
}
ic <- do.call(rbind, ic.list)[0 < intervals]
intervals <- dcast(ic, data + segments ~ fun.name, value.var="intervals")
cost <- dcast(ic, data + segments ~ fun.name, value.var="cost")
not.equal <- cost[PeakSegPDPA != PeakSegPDPAInf]
stopifnot(nrow(not.equal)==0)

intervals[, increase := PeakSegPDPAInf-PeakSegPDPA]
table(intervals$increase)
quantile(intervals$increase)
ic[, list(
  mean=mean(intervals),
  max=max(intervals)
), by=list(fun.name)]
```

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.
PoissonLoss

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)
Index

*Topic datasets
   H3K4me3_PGP_immune_chunk24, 2
   H3K4me3_XJ_immune_chunk1, 2
   H3K4me3_PGP_immune_chunk24, 2
   H3K4me3_XJ_immune_chunk1, 2

oracleModelComplexity, 3

PeakSegFPOP, 3
PeakSegFPOPChrom, 5
PeakSegPDPA, 7
PeakSegPDPAChrom, 9
PeakSegPDPAInf, 11
PoissonLoss, 12