Package 'binsegRcpp'

January 26, 2022

Type Package

Title Efficient Implementation of Binary Segmentation

Version 2022.1.24

Author Toby Dylan Hocking

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

Description Standard template library
containers are used to implement an efficient binary segmentation
algorithm, which is log-linear on average and quadratic in the
worst case.

License GPL-3

LinkingTo Rcpp

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

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

Imports data.table, Rcpp

Suggests covr, directlabels, ggplot2, testthat, knitr, markdown,
neuroblastoma

VignetteBuilder knitr

NeedsCompilation yes

Repository CRAN

Date/Publication 2022-01-26 07:42:47 UTC

R topics documented:

`binsegRcpp-package` *A short title line describing what the package does*

Description

A more detailed description of what the package does. A length of about one to five lines is recommended.

Details

This section should provide a more detailed overview of how to use the package, including the most important functions.

Author(s)

Your Name, email optional.

Maintainer: Your Name <your@email.com>

References

This optional section can contain literature or other references for background information.

See Also

Optional links to other man pages

Examples

```
## Not run:
   ## Optional simple examples of the most important functions
   ## These can be in \dontrun{} and \donttest{} blocks.

## End(Not run)
```

`binseg_normal` *Binary segmentation, normal change in mean*

Description

Efficient implementation of binary segmentation for change in mean, max normal likelihood = min square loss. Output includes columns which can be used to compute parameters for a single model in log-linear time.

Usage

```
binseg_normal(data.vec,
    max.segments = sum(!is.validation.vec),
    is.validation.vec = rep(FALSE,
        length(data.vec)),
    position.vec = seq_along(data.vec))
```

Arguments

`data.vec`	Vector of numeric data to segment.
`max.segments`	Maximum number of segments to compute, default=number of FALSE entries in `is.validation.vec`.
`is.validation.vec`	logical vector indicating which data are to be used in validation set, default=all FALSE (no validation set).
`position.vec`	integer vector of positions at which data are measured, default=1:length(`data.vec`).

Value

list with elements subtrain.borders and splits.

`segments`	number of parameters
`loss`	subtrain square loss
`validation.loss`	validation square loss
`end`	index of last data point per segment
`before.mean`	mean before changepoint
`after.mean`	mean after changepoint
`before.size`	number of data before changepoint
`after.size`	number of data after changepoint
`invalidates.index`	index of model parameter no longer used after this changepoint is used
`invalidates.after`	idem

Author(s)

Toby Dylan Hocking

Examples

```
x <- c(0.1, 0, 1, 1.1, 0.1, 0)
## Compute full path of binary segmentation models from 1 to 6
## segments.
(models <- binsegRcpp::binseg_normal(x))

## Plot loss values using base graphics.
plot(models)

## Same loss values using ggplot2.
if(require("ggplot2")){
  ggplot()+
    geom_point(aes(
      segments, loss),
      data=models$splits)
}

## Compute data table of segments to plot.
(segs.dt <- coef(models, 2:4))

## Plot data, segments, changepoints.
if(require("ggplot2")){
  ggplot()+
    theme_bw()+
    theme(panel.spacing=grid::unit(0, "lines"))+
    facet_grid(segments ~ ., labeller=label_both)+
    geom_vline(aes(
      xintercept=start.pos),
      color="green",
      data=segs.dt[1<start])+
    geom_segment(aes(
      start.pos, mean,
      xend=end.pos, yend=mean),
      data=segs.dt,
      color="green")+
    xlab("Position/index")+
    ylab("Data/mean value")+
    geom_point(aes(
      pos, x),
      data=data.frame(x, pos=seq_along(x)))
}

## Demonstration of model selection using cross-validation in
## simulated data.
seg.mean.vec <- 1:5
data.mean.vec <- rep(seg.mean.vec, each=20)
set.seed(1)
```

```
n.data <- length(data.mean.vec)
data.vec <- rnorm(n.data, data.mean.vec, 0.2)
plot(data.vec)

library(data.table)
loss.dt <- data.table(seed=1:10)[, {
  set.seed(seed)
  is.valid <- sample(rep(c(TRUE,FALSE), l=n.data))
  bs.model <- binsegRcpp::binseg_normal(data.vec, is.validation.vec=is.valid)
  bs.model$splits[, data.table(
    segments,
    validation.loss)]
}, by=seed]
loss.stats <- loss.dt[, .(
  mean.valid.loss=mean(validation.loss)
), by=segments]
plot(
  mean.valid.loss ~ segments, loss.stats,
  col=ifelse(
    mean.valid.loss==min(mean.valid.loss),
    "black",
    "red"))

selected.segments <- loss.stats[which.min(mean.valid.loss), segments]
full.model <- binsegRcpp::binseg_normal(data.vec, selected.segments)
(segs.dt <- coef(full.model, selected.segments))
if(require("ggplot2")){
  ggplot()+
    theme_bw()+
    theme(panel.spacing=grid::unit(0, "lines"))+
    geom_vline(aes(
      xintercept=start.pos),
      color="green",
      data=segs.dt[1<start])+
    geom_segment(aes(
      start.pos, mean,
      xend=end.pos, yend=mean),
      data=segs.dt,
      color="green")+
    xlab("Position/index")+
    ylab("Data/mean value")+
    geom_point(aes(
      pos, data.vec),
      data=data.frame(data.vec, pos=seq_along(data.vec)))
}
```

---

binseg_normal_cv *Binary segmentation, normal change in mean, cross-validation for model selection*

Description

Efficient implementation of binary segmentation for change in mean, with automatic model selection via cross-validation.

Usage

```
binseg_normal_cv(data.vec,
    max.segments = length(data.vec),
    position.vec = seq_along(data.vec),
    n.validation.sets = 100L,
    prop.validation = 0.5)
```

Arguments

`data.vec`	Vector of numeric data to segment.
`max.segments`	Maximum number of segments to compute, default=length(`data.vec`).
`position.vec`	integer vector of positions at which data are measured, default=1:length(`data.vec`).
`n.validation.sets`	Number of validation sets.
`prop.validation`	Proportion of validation set.

Author(s)

Toby Dylan Hocking

Examples

```
seg.mean.vec <- 1:5
data.mean.vec <- rep(seg.mean.vec, each=20)
set.seed(1)
n.data <- length(data.mean.vec)
data.vec <- rnorm(n.data, data.mean.vec, 0.2)
plot(data.vec)
(fit <- binsegRcpp::binseg_normal_cv(data.vec))
seg.dt <- coef(fit)
model.color <- "red"
seg.dt[, segments(start.pos, mean, end.pos, mean, col=model.color)]
seg.dt[start>1, abline(v=start.pos, col=model.color)]

## plot method shows number of times selected.
plot(fit)

if(requireNamespace("neuroblastoma")){
  data(neuroblastoma, package="neuroblastoma", envir=environment())
  library(data.table)
  profiles.dt <- data.table(neuroblastoma$profiles)
  one.chrom <- profiles.dt[profile.id=="4" & chromosome=="2"]
  fit <- one.chrom[, binsegRcpp::binseg_normal_cv(
```

```
    logratio, position.vec=position)]
  selected.segs <- coef(fit)
  if(require(ggplot2)){
    ggplot()+
      geom_point(aes(
        position, logratio),
        data=one.chrom)+
      geom_segment(aes(
        start.pos, mean,
        xend=end.pos, yend=mean),
        data=selected.segs,
        color=model.color)+
      geom_vline(aes(
        xintercept=start.pos),
        data=selected.segs[start>1],
        color=model.color)
  }
}
```

`case.colors` *case colors*

Description

Character vector giving default colors for cases, ordered from worst to best.

Usage

```
"case.colors"
```

`case.sizes` *case sizes*

Description

Numeric vector giving default sizes for cases.

Usage

```
"case.sizes"
```

coef.binseg_normal *coef binseg normal*

Description

Compute a data table of segment start/end/mean values for all models given by `segments`.

Usage

```
## S3 method for class 'binseg_normal'
coef(object,
    segments = 1:min(nrow(object$splits),
        10), ...)
```

Arguments

`object`	data.table from `binseg_normal`.
`segments`	integer vector, model sizes in number of `segments`.
`...`	ignored.

Value

data.table with one row for each segment.

Author(s)

Toby Dylan Hocking

coef.binseg_normal_cv *coef binseg normal cv*

Description

Compute a data table of segment start/end/mean values for all models given by `segments`.

Usage

```
## S3 method for class 'binseg_normal_cv'
coef(object,
    segments = max(nrow(object$splits)),
    ...)
```

Arguments

object	data.table from binseg_normal_cv.
segments	integer vector, model sizes in number of segments. default=number of selected segments.
...	ignored.

Value

data.table with one row for each segment.

Author(s)

Toby Dylan Hocking

---

get_complexity *get complexity*

Description

Get empirical and extreme split counts.

Usage

```
get_complexity(models)
```

Arguments

models	data.table from binseg_normal.

Value

data.table with one row per model size, and column splits with number of splits to check after computing that model size. Column case has values best, worst, empirical.

Author(s)

Toby Dylan Hocking

Examples

```
## Example 1: empirical=worst case.
data.vec <- rep(c(0,1), l=10)
plot(data.vec)
bs.model <- binsegRcpp::binseg_normal(data.vec)
split.counts <- binsegRcpp::get_complexity(bs.model)
plot(split.counts)
```

```
## Example 2: empirical=best case.
data.vec <- 1:20
plot(data.vec)
bs.model <- binsegRcpp::binseg_normal(data.vec)
split.counts <- binsegRcpp::get_complexity(bs.model)
plot(split.counts)

## Example 3: empirical case between best/worst.
seg.mean.vec <- 1:5
data.mean.vec <- rep(seg.mean.vec, each=10)
set.seed(1)
data.vec <- rnorm(length(data.mean.vec), data.mean.vec, 0.2)
plot(data.vec)
bs.model <- binsegRcpp::binseg_normal(data.vec)
split.counts <- binsegRcpp::get_complexity(bs.model)
plot(split.counts)

if(require("ggplot2")){
  ggplot()+
    geom_line(aes(
      segments, splits, color=case, size=case),
      data=split.counts$iterations[case!="empirical"])+
    geom_point(aes(
      segments, splits, color=case),
      data=split.counts$iterations[case=="empirical"])+
    geom_text(aes(
      x, y,
      label=label,
      color=case),
      hjust=1,
      data=split.counts$totals)+
    scale_color_manual(
      values=binsegRcpp::case.colors,
      guide="none")+
    scale_size_manual(
      values=binsegRcpp::case.sizes,
      guide="none")
}
```

`get_complexity_empirical` *get complexity empirical*

Description

Get empirical split counts.

Usage

```
get_complexity_empirical(model.dt)
```

Arguments

model.dt	data.table from binseg_normal.

Value

data.table with one row per model size, and column splits with number of splits to check after computing that model size.

Author(s)

Toby Dylan Hocking

---

get_complexity_extreme *get complexity extreme*

Description

Compute best and worst case number of splits.

Usage

```
get_complexity_extreme(N.data)
```

Arguments

N.data	number of data to segment.

Value

data.table with one row per model size, and column splits with number of splits to check after computing that model size. Column case has values best (equal segment sizes, min splits to check) and worst (unequal segment sizes, max splits to check).

Author(s)

Toby Dylan Hocking

plot.binseg_normal *plot binseg normal*

Description

Plot loss values from binary segmentation.

Usage

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

Arguments

x	data.table from binseg_normal.
...	ignored.

Author(s)

Toby Dylan Hocking

plot.binseg_normal_cv *plot binseg normal cv*

Description

Plot loss values from binary segmentation.

Usage

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

Arguments

x	data.table from binseg_normal_cv.
...	ignored.

Author(s)

Toby Dylan Hocking

plot.complexity *plot complexity*

Description

Plot comparing empirical number of splits to best/worst case.

Usage

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

Arguments

x	data.table from get_complexity.
...	ignored.

Author(s)

Toby Dylan Hocking

print.binseg_normal *print binseg normal*

Description

Print method for binseg_normal.

Usage

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

Arguments

x	data.table from binseg_normal.
...	ignored.

Author(s)

Toby Dylan Hocking

print.binseg_normal_cv *print binseg normal cv*

Description

Print method for binseg_normal_cv.

Usage

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

Arguments

x	data.table from binseg_normal_cv.
...	ignored.

Author(s)

Toby Dylan Hocking

random_set_vec *random set vec*

Description

Random set assignment.

Usage

```
random_set_vec(N, props.vec)
```

Arguments

N	integer, size of output vector.
props.vec	numeric vector of set proportions (must sum to one), with set names.

Value

Random vector of N set names.

Author(s)

Toby Dylan Hocking

Examples

```
library(data.table)
library(ggplot2)
library(binsegRcpp)
tvt.props <- c(test=0.19, train=0.67, validation=0.14)
tvt.N <- 1234567L
system.time({
  tvt.vec <- random_set_vec(tvt.N, tvt.props)
})
table(tvt.vec, useNA="ifany")/tvt.N

random_set_vec(6L, c(train=2/3, test=1/3))
random_set_vec(5L, c(train=2/3, test=1/3))
random_set_vec(4L, c(train=2/3, test=1/3))
random_set_vec(3L, c(train=2/3, test=1/3))

test.rev <- function(N, prop.vec, expected.vec){
  result <- list()
  for(fun.name in c("identity", "rev")){
    fun <- get(fun.name)
    ctab <- table(random_set_vec(N, fun(prop.vec)))
    result[[fun.name]] <- ctab
  }
  result$same <- sapply(
    result, function(tab)identical(as.numeric(tab), expected.vec))
  result
}
test.rev(4L, c(test=1/3, train=2/3), c(1, 3))
table(random_set_vec(3L, c(test=0.5, train=0.5)))
table(random_set_vec(3L, c(train=0.5, test=0.5)))
test.rev(3L, c(test=0.4, train=0.6), c(1, 2))
test.rev(3L, c(test=0.49, train=0.51), c(1, 2))
test.rev(3L, c(test=0.6, train=0.4), c(2, 1))
## 2 is optimal after prob=2/3.
test.rev(2L, c(test=0.6, train=0.4), c(1, 1))
test.rev(2L, c(test=0.7, train=0.3), c(2))

## visualize the likelihood as a function of the proportion of
## success.
test.prop <- seq(0, 1, by=0.01)
prob.dt.list <- list()
n.total <- 2
for(n.test in 0:n.total){
  prob.dt.list[[paste(n.test)]] <- data.table(
    n.test,
    test.prop,
    prob=dbinom(n.test, n.total, test.prop))
}
prob.dt <- do.call(rbind, prob.dt.list)
thresh.dt <- data.table(thresh=(1:2)/3)
gg <- ggplot()+
```

```
  geom_vline(aes(xintercept=thresh), data=thresh.dt)+
  geom_line(aes(
    test.prop, prob, color=n.test, group=n.test),
    data=prob.dt)
if(requireNamespace("directlabels")){
  directlabels::direct.label(gg, "last.polygons")
}else{
  gg
}

## visualize the binomial likelihood as a function of number of
## successes, for a given probability of success.
n.total <- 43
n.success <- 0:n.total
p.success <- 0.6
lik.dt <- data.table(
  n.success,
  prob=dbinom(n.success, n.total, p.success))
ggplot()+
  geom_point(aes(
    n.success, prob),
    data=lik.dt)+
  geom_vline(xintercept=(n.total+1)*p.success)

## visualize the multinomial likelihood as a function of number of
## successes, for a given probability of success.
n.total <- 43
prob.vec <- c(train=0.6, validation=0.3, test=0.1)
train.dt <- data.table(train=0:n.total)
grid.dt <- train.dt[, data.table(
  validation=0:(n.total-train)), by=train]
grid.dt[, prob := dmultinom(
  c(train, validation, n.total-train-validation),
  n.total,
  prob.vec),
  by=.(train, validation)]

train.bound <- (n.total+1)*prob.vec[["train"]]
validation.bound <- (n.total+1)*prob.vec[["validation"]]
guess.dt <- data.table(
  train=floor(train.bound),
  validation=floor(validation.bound))
max.dt <- grid.dt[which.max(prob)]#same
max.dt[, test := n.total-train-validation]

ggplot()+
  geom_tile(aes(
    train, validation, fill=prob),
    data=grid.dt)+
  scale_fill_gradient(low="white", high="red")+
  theme_bw()+
  geom_vline(
    xintercept=train.bound)+
```

```
  geom_hline(
    yintercept=validation.bound)+
  geom_point(aes(
    train, validation),
    shape=1,
    data=guess.dt)+
  coord_equal()

## visualize what happens when we start obs.seq variable above at 1
## or 0. starting at 0 is problematic e.g. 99% train/1% test with
## N=2 observations should return 2 train/0 test (and does when
## obs.seq starts with 1, but does NOT when obs.seq starts with 0).
random_set_vec(2L, c(train=0.99, test=0.01))
obs.dt.list <- list()
cum.dt.list <- list()
for(tvt.N in 2:4){
  obs.dt.list[[paste(tvt.N)]] <- data.table(tvt.N, rbind(
    data.table(start=0, obs=seq(0, tvt.N, l=tvt.N)),
    data.table(start=1, obs=seq(1, tvt.N, l=tvt.N))))
  not.round <- data.table(
    set=c("train", "test"),
    cum.thresh=tvt.N*c((tvt.N-2)/(tvt.N-1), 1))
  cum.dt.list[[paste(tvt.N)]] <- data.table(tvt.N, rbind(
    data.table(round=FALSE, not.round),
    not.round[, .(round=TRUE, set, cum.thresh=round(cum.thresh))]))
}
cum.dt <- do.call(rbind, cum.dt.list)
obs.dt <- do.call(rbind, obs.dt.list)
ggplot()+
  theme_bw()+
  theme(panel.spacing=grid::unit(0, "lines"))+
  facet_grid(tvt.N ~ .)+
  geom_point(aes(
    obs, start),
    data=obs.dt)+
  geom_vline(aes(
    xintercept=cum.thresh, color=round, linetype=round),
    data=cum.dt)
```

Index