Package ‘hypergate’

February 6, 2020

Title Machine Learning of Hyperrectangular Gating Strategies for High-Dimensional Cytometry

Version 0.8.3

Description Given a high-dimensional dataset that typically represents a cytometry dataset, and a subset of the datapoints, this algorithm outputs an hyperrectangle so that datapoints within the hyperrectangle best correspond to the specified subset. In essence, this allows the conversion of clustering algorithms' outputs to gating strategies outputs.

Depends R (>= 3.5.0)

License GPL-3

Encoding UTF-8

LazyData false

Imports stats, grDevices, utils, graphics

Suggests knitr, rmarkdown, flowCore, sp, rgeos

VignetteBuilder knitr

RoxygenNote 6.1.1

NeedsCompilation no

Author Etienne Becht [cre, aut],
Samuel Granjeaud [ctb]

Maintainer Etienne Becht <etienne.becht@protonmail.com>

Repository CRAN

Date/Publication 2020-02-06 09:20:02 UTC

R topics documented:

`boolmat` *boolmat*

Description

Convert an expression matrix and a gating strategy to a boolean matrix (whether each event is gated out by each channel)

Usage

```
boolmat(gate, xp)
```

Arguments

`gate`	A return from hypergate
`xp`	Expression matrix as in the hypergate callxp=Samusik_01_subset$xp_src[,Samusik_01_subset$regular_c

Examples

```
data(Samusik_01_subset)
xp=Samusik_01_subset$xp_src
gate_vector=Samusik_01_subset$labels
hg=hypergate(xp=xp,gate_vector=gate_vector,level=23,delta_add=0.01)
head(boolmat(hg,xp))
```

channels_contributions *channels_contributions*

Description

Gives scores for the contribution of individual channels to a gating strategy

Usage

```
channels_contributions(gate, xp, gate_vector, level, beta = 1)
```

Arguments

gate	A return from hypergate
xp	Expression matrix as in the hypergate call
gate_vector	Categorical vector of length nrow(xp)
level	A level of gate_vector that identifies the population of interest
beta,	should be the same as for the hypergate object

Examples

```
data(Samusik_01_subset)
xp=Samusik_01_subset$xp_src[,Samusik_01_subset$regular_channels]
gate_vector=Samusik_01_subset$labels
hg=hypergate(xp=xp,gate_vector=gate_vector,level=23,delta_add=0)
contribs=channels_contributions(gate=hg,xp=xp,gate_vector=gate_vector,level=23,beta=1)
contribs
```

color_biplot_by_discrete *Colors a biplot according to a vector with discrete values*

Description

Colors a biplot according to a vector with discrete values

Usage

```
color_biplot_by_discrete(matrix, discrete_vector, ..., bty = "l",
  pch = 16, cex = 0.5, colors = NULL)
```

Arguments

matrix	a two columns matrix
discrete_vector	a vector of size nrow(matrix)
...	passed to plot
bty	passed to plot
pch	passed to plot
cex	passed to plot
colors	Palette to used named after the unique elements of discrete_vector. Generated from rainbow() if missing.

Examples

```
data(Samusik_01_subset)
levels=unique(sort(Samusik_01_subset$labels))
colors=setNames(colorRampPalette(palette())(length(levels)),sort(levels))
with(Samusik_01_subset,color_biplot_by_discrete(matrix=tsne,discrete_vector=labels,colors=colors))
```

---

contract *contract*

Description

Test (some) possible contractions of the hyperrectangle

Usage

```
contract(par = par, xp_pos = envir$xp_pos,
  state_pos = envir$state_pos, xp_neg = envir$xp_neg,
  state_neg = envir$state_neg, n = envir$n, TP = envir$TP,
  TN = envir$TN, beta = envir$beta2, envir = parent.frame())
```

Arguments

par	Current parametrization of the hyperrectangle
xp_pos	Expression matrix for positive events
state_pos	State vector of the positive events
xp_neg	Expression matrix for negative events
state_neg	State vector of the negative events
n	passed to f
TP	integer: current number of TP
TN	integer: current number of TN
beta	Passed from the top-level function
envir	Current environment of the optimization

contract.update *contract.update*

Description

Update the hyperrectangle to the best contraction move found

Usage

```
contract.update(contract_object, pars = envir$pars,
  active_channels = envir$active_channels, b_pos = envir$b_pos,
  b_neg = envir$b_neg, state_pos = envir$state_pos,
  state_neg = envir$state_neg, TN = envir$TN, TP = envir$TP,
  xp_pos = envir$xp_pos, xp_neg = envir$xp_neg,
  envir = parent.frame())
```

Arguments

contract_object	output of the contract function
pars	Current parametrization of the hyperrectangle
active_channels	vector of currently-used parameters
b_pos	boolean matrix of positive events
b_neg	boolean matrix of negative events
state_pos	State vector of the positive events
state_neg	State vector of the negative events
TN	integer: current number of TN
TP	integer: current number of TP
xp_pos	Expression matrix for positive events
xp_neg	Expression matrix for negative events
envir	Current environment of the optimization

coreloop *coreloop*

Description

Core optimization loop of hypergate

Usage

```
coreloop(par, hg.env = hg.env$hg.env)
```

Arguments

`par`	Current parametrization of the hyperrectangle
`hg.env`	Environment where the main execution of hypergate takes place

---

`en.locator` *Wrapper to locator that plots segments on the fly*

Description

Wrapper to locator that plots segments on the fly

Usage

```
en.locator()
```

`expand` *expand*

Description

Test (some) possible expansions of the hyperrectangle

Usage

```
expand(FN = envir$FN, FNTN_matrix = envir$FNTN_matrix, TP = envir$TP,
  TN = envir$TN, n = envir$n, beta = envir$beta2,
  envir = parent.frame())
```

Arguments

`FN`	integer: current number of FP
`FNTN_matrix`	Boolean matrix of dim (FN, FN + TN), where Mij is TRUE if and only if expanding to include the ith FN in the gate would lead to the inclusion of the jth column event
`TP`	integer: current number of TP
`TN`	integer: current number of TN
`n`	passed to f
`beta`	Passed from the top-level function
`envir`	Coreloop environment

expand.update *expand.update*

Description

Update the hyperrectangle to the best expansion move found

Usage

```
expand.update(expand.object, pars = envir$pars, xp_pos = envir$xp_pos,
  xp_neg = envir$xp_neg, state_pos = envir$state_pos,
  state_neg = envir$state_neg, b_pos = envir$b_pos,
  b_neg = envir$b_neg, n = envir$n, TP = envir$TP, TN = envir$TN,
  envir = parent.frame())
```

Arguments

`expand.object`	output of the expand function
`pars`	Current parametrization of the hyperrectangle
`xp_pos`	Expression matrix for positive events
`xp_neg`	Expression matrix for negative events
`state_pos`	State vector of the positive events
`state_neg`	State vector of the negative events
`b_pos`	boolean matrix of positive events
`b_neg`	boolean matrix of negative events
`n`	passed to f
`TP`	integer: current number of TP
`TN`	integer: current number of TN
`envir`	Current environment of the optimization

f *f*

Description

Computes the F_beta score given an intenger number of True Positives (TP), True Negatives (TN). It is optimized for speed and n is thus not the total number of events

Usage

```
f(TP, TN, n, beta2 = 1)
```

Arguments

`TP`	Number of true positive events
`TN`	Number of true negative events
`n`	beta^2*(TP+FN)+TN+FP
`beta2`	squared-beta to weight precision (low beta) or recall (high beta) more

---

`fill_FNTN_matrix` *fill_FNTN_matrix*

Description

fill_FNTN_matrix Used for assessing whether an expansion move is possible

Usage

```
fill_FNTN_matrix(xp_FN, xp_TN, B_FN, B_TN, par)
```

Arguments

`xp_FN`	Expression matrix of False Negative events
`xp_TN`	Expression matrix of True Negative events
`B_FN`	Boolean matrix of FN events
`B_TN`	Boolean matrix of TN events
`par`	Current hyper-rectangle parametrization

`FNTN_matrix.recycle` *FNTN_matrix.recycle*

Description

Recycle an expansion matrix

Usage

```
FNTN_matrix.recycle(FNTN_matrix, B_FN_old, B_TN_old, B_FN_new, B_TN_new,
  xp_FN, xp_TN, par)
```

Arguments

FNTN_matrix	Expansion matrix to recycle
B_FN_old	Boolean matrix of FN events before the last expansion
B_TN_old	Boolean matrix of TN events before the last expansion
B_FN_new	Boolean matrix of FN events after the last expansion
B_TN_new	Boolean matrix of TN events after the last expansion
xp_FN	Expression matrix of False Negative events
xp_TN	Expression matrix of True Negative events
par	Current hyper-rectangle parametrization

F_beta *F_beta*

Description

Compute a F_beta score comparing two boolean vectors

Usage

```
F_beta(pred, truth, beta = 1)
```

Arguments

pred	boolean vector of predicted values
truth	boolean vector of true values
beta	Weighting of yield as compared to precision. Increase beta so that the optimization favors yield, or decrease to favor purity.

Examples

```
data(Samusik_01_subset)
truth=c(rep(TRUE,40),rep(FALSE,60))
pred=rep(c(TRUE,FALSE),50)
table(pred,truth) ##40% purity, 50% yield
#' F_beta(pred=pred,truth=truth,beta=2) ##Closer to yield
F_beta(pred=pred,truth=truth,beta=1.5) ##Closer to yield
F_beta(pred=pred,truth=truth,beta=1) ##Harmonic mean
F_beta(pred=pred,truth=truth,beta=0.75) ##Closer to purity
F_beta(pred=pred,truth=truth,beta=0.5) ##Closer to purity
```

`gate_from_biplot` *gate_from_biplot*

Description

From a biplot let the user interactively draw polygons to create a "Gate" vector

Usage

```
gate_from_biplot(matrix, x_axis, y_axis, ..., bty = "l", pch = 16,
  cex = 0.5, sample = NULL)
```

Arguments

`matrix`	A matrix
`x_axis`	character, colname of matrix used for x-axis in the biplot
`y_axis`	character, colname of matrix used for y-axis in the biplot
`...`	passed to plot
`bty`	passed to plot
`pch`	passed to plot
`cex`	passed to plot
`sample`	Used to downsample the data in case there are too many events to plot quickly

Details

Data will be displayed as a bi-plot according to user-specified x_axis and y_axis arguments, then a call to locator() is made. The user can draw a polygon around parts of the plot that need gating. When done, 'right-click' or 'escape' (depending on the IDE) escapes locator() and closes the polygon. Then the user can press "n" to draw another polygon (that will define a new population), "c" to cancell and draw the last polygon again, or "s" to exit. When exiting, events that do not fall within any polygon are assigned NA, the others are assigned an integer value corresponding to the last polygon they lie into.

Value

A named vector of length nrow(matrix) and names rownames(matrix). Ungated events are set to NA

Examples

```
if(interactive()){
    ##See the details section to see how this function works
    gate_from_biplot(matrix=Samusik_01_subset$tsne,x_axis="tSNE1",y_axis="tSNE2")
}
```

hgate_info *hgate_info*

Description

Extract information about a hypergate return: the channels of the phenotype, the sign of the channels, the sign of the comparison, the thresholds. The function could also compute the Fscores if the xp, gate_vector and level parameters are given.

Usage

```
hgate_info(hgate, xp, gate_vector, level, beta = 1)
```

Arguments

hgate	A hypergate object (produced by hypergate())
xp	The expression matrix from which the 'hgate' parameter originates, needed for Fscore computation
gate_vector	Categorical data from which the 'hgate' parameter originates, needed for Fscore computation
level	Level of gate_vector identifying the population of interest, needed for Fscore computation
beta	Beta to weight purity (low beta) or yield (high beta) more, needed for Fscore computation

Value

A data.frame with channel, sign, comp and threshold columns, and optionnally deltaF (score deterioration when parameter is ignored),Fscore1d (F_value when using only this parameter) and Fscore (F score when all parameters up to this one are included). Fscores are computed if xp, gate_vector and level are passed to the function.

See Also

hg_pheno, hg_rule

Examples

```
data(Samusik_01_subset)
xp=Samusik_01_subset$xp_src[,Samusik_01_subset$regular_channels]
gate_vector=Samusik_01_subset$labels
hg=hypergate(xp=xp,gate_vector=gate_vector,level=23,delta_add=0.01)
hgate_info(hgate=hg)
hgate_pheno(hgate=hg)
hgate_rule(hgate=hg)
```

hgate_pheno *hgate_pheno*

Description

Build a human readable phenotype, i.e. a combination of channels and sign (+ or -) from a hypergate return.

Usage

```
hgate_pheno(hgate, collapse = ", ")
```

Arguments

`hgate`	A hypergate object (produced by hypergate())
`collapse`	A character string to separate the markers.

Value

A string representing the phenotype.

See Also

`hg_rule`, `hg_info`

Examples

```
## See hgate_info
```

hgate_rule *hgate_rule*

Description

Build a human readable rule i.e. a combination of channels, sign of comparison and threshold.

Usage

```
hgate_rule(hgate, collapse = ", ", digits = 2)
```

Arguments

`hgate`	A hypergate object (produced by hypergate())
`collapse`	A character string to separate the markers.
`digits`	An integer that specifies the decimal part when rounding.

Value

A data.frame with channel, sign, comp and threshold columns

See Also

`hg_pheno`, `hg_rule`

Examples

```
## See hgate_info
```

`hgate_sample` *hgate_sample*

Description

Downsample the data in order to fasten the computation and reduce the memory usage.

Usage

```
hgate_sample(gate_vector, level, size = 1000, method = "prop")
```

Arguments

`gate_vector`	A Categorical vector whose length equals the number of rows of the matrix to sample (nrow(xp))
`level`	A level of gate_vector so that gate_vector == level will produce a boolean vector identifying events of interest
`size`	An integer specifying the maximum number of events of interest to retain. If the count of events of interest is lower than `size`, than `size` will be set to that count.
`method`	A string specifying the method to balance the count of events. `"prop"` means proportionnality: if events of interest are sampled in a 1/10 ratio, then all others events are sampled by the same ratio. `"10x"` means a balance of 10 between the count events of interest and the count all others events. `"ceil"` means a uniform sampling no more than the specified size for each level of the gate_vector. `level` is unused in that method.

Value

A logical vector with TRUE correspond to the events being sampled, ie kept to further analysis

Note

No replacement is applied. If there are less events in one group or the alternate than the algorithm requires, then all available events are returned. NA values in gate_vector are not sampled, ie ignored.

Examples

```
# Standard procedure with downsampling
data(Samusik_01_subset)
xp <- Samusik_01_subset$xp_src[,Samusik_01_subset$regular_channels]
gate_vector <- Samusik_01_subset$labels
sampled <- hgate_sample(gate_vector, level=8, 100)
table(sampled)
table(gate_vector[sampled])
xp_sampled <- xp[sampled, ]
gate_vector_sampled <- gate_vector[sampled]
hg <- hypergate(xp_sampled, gate_vector_sampled, level=8, delta_add=0.01)
# cluster 8 consists in 122 events
table(gate_vector)
# Downsampling
table(gate_vector[hgate_sample(gate_vector, level=8, 100)])
# Downsampling reduces the alternate events
table(gate_vector[hgate_sample(gate_vector, level=8, 100, "10x")])
# Downsampling is limited to the maximum number of events of interest
table(gate_vector[hgate_sample(gate_vector, level=8, 150)])
# Downsampling is limited to the maximum number of events of interest, and
# the alternate events are downsampled to a total of 10 times
table(gate_vector[hgate_sample(gate_vector, level=8, 150, "10x")])
# More details about sampling
# Convert -1 to NA, NA are not sampled
gate_vector[gate_vector==-1] = NA
gate_vector = factor(gate_vector)
table(gate_vector, useNA = "alw")
#
# target size = 100 whereas initial freq is 122 for pop 8
smp.prop = hgate_sample(gate_vector, level = 8, size = 100, method = "prop")
smp.10x  = hgate_sample(gate_vector, level = 8, size = 100, method = "10x")
smp.ceil = hgate_sample(gate_vector, size = 10, method = "ceil")
table(smp.prop)
table(smp.10x)
table(smp.ceil)
rbind(raw = table(gate_vector),
      prop = table(gate_vector[smp.prop]),
      `10x` = table(gate_vector[smp.10x]),
      ceil = table(gate_vector[smp.ceil]))
#
# target size = 30 whereas initial freq is 25 for pop 14
smp.prop = hgate_sample(gate_vector, level = 14, size = 30, method = "prop")
smp.10x  = hgate_sample(gate_vector, level = 14, size = 30, method = "10x")
table(smp.prop)
table(smp.10x)
rbind(raw = table(gate_vector),
      prop = table(gate_vector[smp.prop]),
      `10x` = table(gate_vector[smp.10x]))
# prop returns original data, because target size ids larger than initial freq
# 10x  returns sampled data according to initial freq, such as the total amount
# of other events equals 10x initial freq of pop 14
```

hypergate *hypergate*

Description

Finds a hyperrectangle gating around a population of interest

Usage

```
hypergate(xp, gate_vector, level, delta_add = 0, beta = 1,
  verbose = FALSE)
```

Arguments

xp	an Expression matrix
gate_vector	A Categorical vector of length nrow(xp)
level	A level of gate_vector so that gate_vector == level will produce a boolean vector identifying events of interest
delta_add	If the increase in F after an optimization loop is lower than delta_add, the optimization will stop (may save computation time)
beta	Purity / Yield trade-off
verbose	Boolean. Whether to print information about the optimization status.

See Also

channels_contributions for ranking parameters within the output, reoptimize_strategy for reoptimizing a output on a subset of the markers, plot_gating_strategy for plotting an output, subset_matrix_hg to apply the output to another input matrix, boolmat to obtain a boolean matrix stating which events are filtered out because of which markers

Examples

```
data(Samusik_01_subset)
xp=Samusik_01_subset$xp_src[,Samusik_01_subset$regular_channels]
gate_vector=Samusik_01_subset$labels
hg=hypergate(xp=xp,gate_vector=gate_vector,level=23,delta_add=0.01)
```

`plot_gating_strategy` *plot_gating_strategy*

Description

Plot a hypergate return

Usage

```
plot_gating_strategy(gate, xp, gate_vector, level, cex = 0.5,
  highlight = "black", path = "./", ...)
```

Arguments

`gate`	A hypergate object (produced by hypergate())
`xp`	The expression matrix from which the 'gate' parameter originates
`gate_vector`	Categorical data from which the 'gate' parameter originates
`level`	Level of gate_vector identifying the population of interest
`cex`	size of dots
`highlight`	color of the positive population when plotting
`path`	Where png files will be produced
`...`	passed to png

Examples

```
data(Samusik_01_subset)
xp=Samusik_01_subset$xp_src[,Samusik_01_subset$regular_channels]
gate_vector=Samusik_01_subset$labels
hg=hypergate(xp=xp,gate_vector=gate_vector,level=23,delta_add=0.01)
par(mfrow=c(1,ceiling(length(hg$active_channels)/2)))
plot_gating_strategy(gate=hg,xp=xp,gate_vector=gate_vector,level=23,highlight="red")
```

`polygon.clean` *Remove self intersection in polygons*

Description

Remove self intersection in polygons

Usage

```
polygon.clean(poly)
```

Arguments

`poly`	a polygon (list with two components x and y which are equal-length numerical vectors)

Value

A polygon without overlapping edges and new vertices corresponding to non-inner points of intersection

---

reoptimize_strategy *reoptimize_strategy*

Description

Optimize a gating strategy given a manual selection of channels

Usage

```
reoptimize_strategy(gate, channels_subset, xp, gate_vector, level,
  beta = 1, verbose = FALSE)
```

Arguments

`gate`	A return from hypergate
`channels_subset`	Character vector identifying the channels that will be retained (others are ignored). The form is e.g. c("CD4_min","CD8_max")
`xp`	Expression matrix as in the hypergate call
`gate_vector`	Categorical vector as in the hypergate call
`level`	Level of gate_vector identifying the population of interest
`beta`	Yield / purity trade-off
`verbose`	Whether to print information about optimization status

Examples

```
data(Samusik_01_subset)
xp=Samusik_01_subset$xp_src[,Samusik_01_subset$regular_channels]
gate_vector=Samusik_01_subset$labels
hg=hypergate(xp=xp,gate_vector=gate_vector,level=23,delta_add=0)
contribs=channels_contributions(gate=hg,xp=xp,gate_vector=gate_vector,level=23,beta=1)
significant_channels=names(contribs)[contribs>=0.01]
hg_reoptimized=reoptimize_strategy(gate=hg,channels_subset=significant_channels,xp,gate_vector,23)
```

Samusik_01_subset *2000 events randomly sampled from the 'Samusik_01' dataset*

Description

2000 events randomly sampled from the 'Samusik_01' dataset

Usage

```
data(Samusik_01_subset)
```

Format

list with three elements: xp_src (expression matrix), labels (manual gates of the events) and tsne (a tSNE projection of the dataset)

References

https://flowrepository.org/id/FR-FCM-ZZPH

subset_matrix_hg *subset_matrix_hg*

Description

Returns a boolean vector whose TRUE elements correspond to events inside the hyperrectangle

Usage

```
subset_matrix_hg(gate, xp)
```

Arguments

gate	a return from hypergate
xp	Expression matrix used for gate

Examples

```
data(Samusik_01_subset)
xp=Samusik_01_subset$xp_src[,Samusik_01_subset$regular_channels]
gate_vector=Samusik_01_subset$labels
hg=hypergate(xp=xp,gate_vector=gate_vector,level=23,delta_add=0.01)
gating_state=subset_matrix_hg(hg,xp)
gating_state=ifelse(gating_state,"Gated in","Gated out")
target=ifelse(gate_vector==23,"Target events","Others")
table(gating_state,target)
```

`update_gate` *Updates a gate vector*

Description

Updates a gate vector

Usage

```
update_gate(xp, polygon, gate_vector = rep(0, nrow(xp)), value = 1)
```

Arguments

`xp`	A two colums matrix
`polygon`	A list with two components x and y of equal lenghts and numeric values
`gate_vector`	a vector of length nrow(xp) with integer values
`value`	The number that will be assigned to gate_vector, corresponding to points that lie in the polygon

Value

The updated gate_vector

Index