flowStats

October 25, 2011

`BackGating` *Sample backgating results*

Description

A data frame containing the sub-populations of ITN dataset corresponding to the high-density areas on "FSC" and "SSC" channels. This dataset is yielded by `backGating` on channel CD3, CD8, and CD4 of the `ITN` sample data.

Usage

```
data(BackGating)
```

Source

Results from executing the following code:

data(ITN)

flowStats:::backGating(ITN, xy=c("FSC", "SSC"), channels=c("CD3", "CD8", "CD4"))

`ITN` *Sample flow cytometry data*

Description

A `flowSet` cotaining data from 15 patients.

Usage

```
data(ITN)
```

Format

A `flowSet` containing 15 `flowFrames`. There are 3 patient groups with 5 samples each.

Source

Immune Tolerance Network

autoGate *Automated gating of single populations in 2D*

Description

This function tries to fit a single norm2Filter based on a rough preselection of the data. This function is considered internal. Please use the API provided by lymphGate.

Usage

```
autoGate(x, ..., scale = 2.5)
```

Arguments

x	An object of class flowSet
...	Named arguments or a list of the ranges used for the initial rough preselection. This gets passed on to rectangleGate, see it's documentation for details.
scale	The scale parameter that gets passed on to norm2Filter.

Details

The flowSet is first filtered using a rectangleGate and the norm2Filter is subsequently fitted to the remaining subset.

Value

A list with items:

x	The filtered flowSet.
n2gate	The norm2Filter object.
n2gateResults	The filterResult after applying the norm2Filter on the flowSet.

Author(s)

Florian Hahne

See Also

lymphGate, norm2Filter

Examples

```
data(GvHD)
flowStats:::autoGate(GvHD[10:15], "FSC-H"=c(100,500), "SSC-H"=c(0, 400))
```

binByRef — *Bin a test data set using bins previously created by probability*

Description

The bins generated by probability binning a control data set can be applied to a test data set to perfom statistical comparisions by methods such as the Chi-squared test or the probability binning statistic.

Usage

```
binByRef(binRes, data)
```

Arguments

binRes	The result generated by calling teh probBin function on a control dataset.
data	An object of class flowFrame

Value

An enviroment containing the matrices for each bin of the test data set

Author(s)

Nishant Gopalakrishnan

See Also

plotBins, proBin

Examples

```
data(GvHD)
resCtrl<-proBin(GvHD[[1]],200)
resSample<-binByRef(resCtrl,GvHD[[2]])
ls(resSample)
```

calcPBChiSquare — *Probability binning metirc for comparing the probability binned*

Description

This function calculates the Probability binning metric proposed by Baggerly et al. The function utilizes the data binned using the proBin and binByRef functions.

Usage

```
calcPBChiSquare(ctrlRes,sampRes,ctrlCount,sampCount)
```

Arguments

ctrlRes	The result generated by calling the probBin function on a control dataset.
sampRes	The result generated by calling the byByRef function on a test sample dataset
ctrlCount	The number of events in the control sample
sampCount	The number of events in the test sample being compared

Value

A list containing the statistic, p.value, observed, expected counts and the residuals

Author(s)

Nishant Gopalakrishnan

See Also

proBin, calcPBChiSquare

Examples

```
data(GvHD)
# flow frame 1 is treated as  control dataset and used to generate bins
resCtrl<-proBin(GvHD[[1]][,c("FSC-H","SSC-H","Time")],200)
plotBins(resCtrl,GvHD[[1]],channels=c("FSC-H","SSC-H","Time"),title="Binned control data"
# Same bins are applied to flowFrame 16
resSample<-binByRef(resCtrl,GvHD[[16]][,c("FSC-H","SSC-H","Time")])
ctrlCount<-nrow(GvHD[[1]])
sampCount<-nrow(GvHD[[16]])
stat<-calcPBChiSquare(resCtrl,resSample,ctrlCount,sampCount)
```

---

calcPearsonChi *Pearsons chi-square statistic for comparing the probability binned*

Description

This function calculates the Pearsons chi-squared statistic for comparing data binned using the proBin and binByRef functions.Internally, the function utilizes the chisq.test function.

Usage

```
calcPearsonChi(ctrlRes,sampRes)
```

Arguments

ctrlRes	The result generated by calling the probBin function on a control dataset.
sampRes	The result generated by calling the byByRef function on a sample dataset

Value

A list containing the statistic, p.value, observed, expected counts and the residuals

Author(s)

Nishant Gopalakrishnan

See Also

proBin, calcPBChiSquare

Examples

```
data(GvHD)
# flow frame 1 is treated as  control dataset and used to generate bins
resCtrl<-proBin(GvHD[[1]][,c("FSC-H","SSC-H","Time")],200)
plotBins(resCtrl,GvHD[[1]],channels=c("FSC-H","SSC-H","Time"),title="Binned control data"
# Same bins are applied to flowFrame 16
resSample<-binByRef(resCtrl,GvHD[[16]][,c("FSC-H","SSC-H","Time")])
stat<-calcPearsonChi(resCtrl,resSample)
```

---

curvPeaks *Parse curv1Filter output*

Description

Parse the output of curv1Filter and find modes and midpoints of the high-density regions. This function is considered to be internal.

Usage

```
curvPeaks(x, dat, borderQuant = 0.01, n = 201, from, to, densities=NULL)
```

Arguments

x	A multipleFilterResult produced by a curv1Filter operation.
dat	The corresponding flowFrame.
borderQuant	A numeric in [0,1] giving the extreme quantiles for which high-density regions are ignored.
n, from, to	Arguments are passed on to density.
densities	The optional y values of the density estimate computed for the respective data.

Value

A list with items

peaks	x and y locations of the modes of the regions in the density estimates.
regions	the left and right margins of the regions.
midpoints	the mean of regions.
regPoints	x and y locations of the outline of the significant density regions.
densFuns	an approximation function of the density estimate

Author(s)

Florian Hahne

See Also

landmarkMatrix

Examples

```
data(GvHD)
tmp <- filter(GvHD[[10]], curv1Filter("FSC-H"))
res <-  flowStats:::curvPeaks(tmp, exprs(GvHD[[10]])[, "FSC-H"])
```

---

density1d *Find most likely separation between positive and negative populations*

Description

The function tries to find a reasonable split point between the two hypothetical cell populations "positive" and "negative". This function is considered internal, please use the API provided by rangeGate.

Usage

```
density1d(x, stain, alpha = "min", sd = 2, plot = FALSE, borderQuant =
0.1, absolute = TRUE, inBetween = FALSE, refLine=NULL, ...)
```

Arguments

x	A flowSet or flowFrame.
stain	A character scalar giving the flow parameter for which to compute the separation.
alpha	A tuning parameter that controls the location of the split point between the two populations. This has to be a numeric in the range [0,1], where values closer to 0 will shift the split point closer to the negative population and values closer to 1 will shift towards the positive population. Additionally, the value of alpha can be "min", in which case the split point will be selected as the area of lowest local density between the two populations.
sd	For the case where there is only a single population, the algorithm falls back to esitmating the mode of this population and a robust measure of the variance of it distribution. The sd tuning parameter controls how far away from the mode the split point is set.
plot	Create a plot of the results of the computation.
borderQuant	Usualy the instrument is set up in a way that the positive population is somewhere on the high end of the measurement range and the negative population is on the low end. This parameter allows to disregard populations with mean values in the extreme quantiles of the data range. It's value should be in the range [0,1].

absolute	Logical controling whether to classify a population (positive or negative) relative to the theoretical measurment range of the instrument or the actual range of the data. This can be set to TRUE if the alignment of the measurment range is not optimal and the bulk of the data is on one end of the theoretical range.
inBetween	Force the algorithm to put the separator in between two peaks. If there are more than two peaks, this argument is ignored.
refLine	Either NULL or a numeric of lenth 1. If NULL, this parameter is ignored. When it is set to a numeric, the minor sub-population (if any) below this reference line will be igored while determining the separator between positive and negative.
...	Further arguments.

Details

The algorithm first tries to identify high density regions in the data. If the input is a flowSet, density regions will be computed on the collapsed data, hence it should have been normalized before (see warpSet for one possible normalization technique). The high density regions are then clasified as positive and negative populations, based on their mean value in the theoretical (or absolute if argument absolute=TRUE) measurement range. In case there are only two high-density regions the lower one is usually clasified as the negative populations, however the heuristics in the algorithm will force the classification towards a positive population if the mean value is already very high. The absolute and borderQuant arguments can be used to control this behaviour. The split point between populations will be drawn at the value of mimimum local density between the two populations, or, if the alpha argument is used, somewhere between the two populations where the value of alpha forces the point to be closer to the negative (0 - 0.5) or closer to the positive population (0.5 - 1).

If there is only a single high-density region, the algorithm will fall back to estimating the mode of the distribution (hubers) and a robust measure of it's variance and, in combination with the sd argument, set the split point somewhere in the right or left tail, depending on the classification of the region.

For more than two populations, the algorithm will still classify each population into positive and negative and compute the split point between those clusteres, similar to the two population case.

Value

A numeric indicating the split point between positive and negative populations.

Author(s)

Florian Hahne

See Also

warpSet, rangeGate

Examples

```
data(GvHD)
dat <- GvHD[pData(GvHD)$Patient==10]
dat <- transform(dat, "FL4-H"=asinh(`FL4-H`), "FL3-H"=asinh(`FL3-H`))
d <- flowStats:::density1d(dat, "FL4-H", plot=TRUE)
if(require(flowViz))
densityplot(~`FL4-H`, dat, refline=d)
```

```
## tweaking the location
flowStats:::density1d(dat, "FL4-H", plot=TRUE, alpha=0.8)

## only a single population
flowStats:::density1d(dat, "FL3-H", plot=TRUE)
flowStats:::density1d(dat, "FL3-H", plot=TRUE, sd=2)
```

`flowStats-package` *Statistical methods for flow cytometry data analysis*

Description

Functions, methods and classes implementing algorithmns for statistical analysis of flow cytometry data. This involves mostly data normalization and automated gating.

Details

Package:	flowStats
Type:	Package
Version:	1.0
License:	Artistic-2.0
Lazyload:	yes

Author(s)

Florian Hahne

Maintainer: Florian Hahne <fhahne@fhcrc.org>

`gaussNorm` *Per-channel normalization based on landmark registration*

Description

This funciton normalizes a set of flow cytometry data samples by identifying and aligning the high density regions (landmarks or peaks) for each channel. The data of each channel is shifted in such a way that the identified high density regions are moved to fixed locations called base landmarks.

Usage

```
gaussNorm (flowset, channel.names, max.lms=2, base.lms=NULL,
peak.density.thr=0.05, peak.distance.thr=0.05, debug=FALSE, fname='')
```

Arguments

flowset	A flowSet.
channel.names	A character vector of flow parameters in flowset to be normalized.
max.lms	A numeric vector of the maximum number of base landmarks to be used for normalizing each channel. If it has only one value that will be used as the maximum number of base landmarks for all the channels.
base.lms	A list of vector for each channel that contains the base landmarks for normalizing that channel. If not specified the base landmarks are computed from the set of extracted landmarks.
peak.density.thr	The peaks with density value less than "peak.density.thr times maximum peak density" are discarded.
peak.distance.thr	The sequences of peaks that are located closer than "peak.distance.thr times range of data" are identified. Then for each sequence only one peak (the one with the highest intensity value) is used as a landmark. In other words no two landmarks are located closer than "peak.distance.thr times range of data" to each other.
debug	Logical. Forces the function to draw before and after normalization plots for each sample. The plot of the i-th sample is stored in paste(fname, i) file.
fname	The pre- and post- normalization plots of the i-th sample is stored in paste(fname, i) file if debug is set to TRUE. If default value is used the plots are drawn on separate X11 windows for each sample. In this case, the function waits for a user input to draw the plots for the next sample.

Details

Normalization is archived in three phases: (i) identifying high-density regions (landmarks) for each flowFrame in the flowSet for a single channel; (ii) computing the best matching between the landmarks and a set of fixed reference landmarks for each channel called base landmarks; (iii) manipulating the data of each channel in such a way that each landmark is moved to its matching base landmark. Please note that this normalization is on a channel-by-channel basis. Multiple channels are normalized in a loop.

Value

A list with items flowset: normalized flowSet. confidence: a confidence measure of the normalization procedure.

Author(s)

Alireza Hadj Khodabakhshi

Examples

```
data(ITN)
dat <- transform(ITN, "CD4"=asinh(CD4), "CD3"=asinh(CD3), "CD8"=asinh(CD8))
lg <- lymphGate(dat, channels=c("CD3", "SSC"),
preselection="CD4",scale=1.5)
dat <- Subset(dat, lg$n2gate)
```

```
datr <- gaussNorm(dat, "CD8")$flowset
if(require(flowViz)){
d1 <- densityplot(~CD8, dat, main="original", filter=curv1Filter("CD8"))
d2 <- densityplot(~CD8, datr, main="normalized", filter=curv1Filter("CD8"))
plot(d1, split=c(1,1,2,1))
plot(d2, split=c(2,1,2,1), newpage=FALSE)
}
```

---

gpaSet *Multi-dimensional normalization of flow cytometry data*

Description

This function performs a multi-dimensional normalization of flow cytometry data (flowSets) using a generalized Procrustes analysis (GPA) method.

Usage

```
gpaSet(x, params, register="backgating", bgChannels=NULL,
       bg=NULL,  rotation.only=TRUE,
       downweight.missingFeatures=FALSE, thres.sigma=2.5,
       show.workflow=FALSE,
       ask=names(dev.cur())!="pdf")
```

Arguments

x	A flowSet.
params	A character vector of length 2 describing the channels of interest.
register	A character indicating the method to be used for identifying features. Available method only includes "backgating" at the point.
bgChannels	A character vector indicating the channels used for backgating. If NULL, backGating will find the appropriate backgating channels.
bg	A data frame as the returning value of the backGating function. If not NULL, gpaSet will skip the backGating process and use the given data frame to extract potential features.
rotation.only	Logical for coarsing a reflection matrix to a rotation matrix.
downweight.missingFeatures	Logical. If TRUE, the missing features, labeled as bogus features, are down-weighted to zero. See details.
thres.sigma	A numerical value indicating the threshold of where to cut the tree, e.g., as resulting from diana, into several clusters. It is default to 2.5 sigma of the distribution of the heights of the cluster points.
show.workflow	Logical. If TRUE, the workflow of gpaSet will be displayed.
ask	Logical. If TRUE, the display operates in interactive mode.

Details

Normalization is achieved by first identifying features for each `flowFrame` in the `flowSet` for designated channels using backgating, subsequently labeling features, and finally aligning the features to a reference feature in the sense of minimizing the Frobenus norm of

$$||sFQ - \bar{F}||,$$

where s is a scalar, Q a rotational matrix, F the matrix of features, and $\bar{F}$ the reference feature. Both s and Q are solved by using singular value decomposition (SVD).

Note that if feature F_{ij} is missing, it is given a bogus value as $\bar{F}_{ij}$.

If `downweight.missingFeatures` is TRUE, the cost function becomes

$$||sW_0FQ - W_0\bar{F}||,$$

where the weighting function W_0 is zero if the corresponding feature is bogus.

Value

The normalized `flowSet` with "GPA" attribute.

Author(s)

C. J. Wong <cwon2@fhcrc.org>

References

in progress

Examples

```
## Example 1: calling up gpaSet directly
data(ITN)
data(BackGating)

tl <- transformList(colnames(ITN)[3:7], asinh, transformationId="asinh")
dat <- transform(ITN, tl)

xy = c("FSC", "SSC")
bgChannels = c("CD8", "CD4", "CD3")
## bg <- flowStats:::backGating(dat, xy=xy, channels=bgChannels)
## using pre-generated backgating results: BackGating
s <- gpaSet(dat, params=xy, bgChannels=bgChannels, bg=BackGating)

if(require(flowViz)) {
   d1 <- densityplot(~., s, channels=c("FSC", "SSC"),
                     layout=c(2,1), main="After GPA using bg")
   d2 <- xyplot(FSC ~ SSC, as(s, "flowFrame"),
                channels=c("FSC", "SSC"), main="All flowFrames")
   plot(d1)
   plot(d2)
}

## view "GPA" attribute
attr(s, "GPA")
```

```
## Not run:
## Example 2: using work flow and normalization objects
data(ITN)
ITN <- ITN[1:8, ]
wf <- workFlow(ITN)
tl <- transformList(colnames(ITN)[3:7], asinh, transformationId="asinh")
add(wf, tl)
x <- Data(wf[["asinh"]])
## normalize 'FSC' and 'SSC' channels
norm <- normalization(normFun=function(x, parameters, ...)
        gpaSet(x, parameters, ...),
        parameters = c("FSC", "SSC"),
arguments=list(bgChannels=c("CD8", "CD3"),
                       register="backgating"),
normalizationId="Procrustes")

add(wf, norm2, parent="asinh")
s <- Data(wf[["Procrustes"]])
if(require(flowViz)) {
   d1 <- densityplot(~., s, channels=c("FSC", "SSC"),
                     layout=c(2,1), main="After GPA using bg")
   d2 <- xyplot(FSC ~ SSC, as(s, "flowFrame"),
                channels=c("FSC", "SSC"), main="All flowFrames")
   plot(d1)
   plot(d2)
}

## End(Not run) ## end of dontrun
```

`iProcrustes` *Procrustes analysis. Using singular value decomposition (SVD) to*

Description

Based on generalized Procrustes analysis, this function determines a linear transformation (rotation/reflection and scalling) of the points in matrix `x` to align them to their reference points in matrix `xbar`. The alignemnt is carried out by minimizing the distance between the points in `x` and `xbar`.

Usage

```
iProcrustes(x, xbar, rotation.only=TRUE, scalling=TRUE, translate=FALSE)
```

Arguments

`x`	A numerical matrix to be align to points in `xbar`, the second arguement. The columns represents the coordinates of the points. The matrices `x` and `xbar` must have the same dimensions.
`xbar`	A numerical, reference matrix to which points in matrix `x` are to be aligned.
`rotation.only`	Logical. When `rotaion.only` is TRUE, it allows the function to lose reflection component of the linear transformation. Although it might not give

the best-fitting aligenment, when dealing with flow cytometry data alignment, a non-reflection transformation is prefered. When `rotaion.only` is FALSE, it allows the function to retain the reflection component.

`scalling` Logical. When `scalling` is FALSE, it allows the function to calculate the linear transformation without a scalling factor. That is, the returning scalling factor is set to 1.

`translate` Logical. Set `translate` to FALSE when the points in matrices x and xbar are already centralized prior to applying this function. When `translate` is TRUE, it allows the function to translate the centroid the points in matrix `x` to that of points in `xbar`.

Details

Suppose the points in matrix X and $\bar{X}$ are centralized (meaning their centroids are at the origin). The linear transformation of X for aligning X to its reference matrix $\bar{X}$., i.e., min $||sXQ - \bar{X}||_F$, is given by:

$$Q = VU^T,$$

and

$$s = trace(\bar{X}^T XQ)/trace(X^T X),$$

where V and U are the sigular value vectors of $\bar{X}^T X$ (that is, $\bar{X}^T X = U\Sigma V^T$), and s is the scalling factor.

Value

A list of the linear tranformation with items

`Q` An orthogonal, rotation/reflection matrix.

`scal` A scalling factor

.

`T` (optional) A translation vector used to shift the centroid of the points in matrix `x` to the origin. Returned when `translate` is TRUE.

`T.xbar` (optional) Centered `xbar` (that is, the centroid of the points in `xbar` is translated to the origin). Returned when `translate` is TRUE.

Note that the return values of this function do not include the transformed matrix $scal * x * Q$ or $scal * (x - IT) * Q$, where T is the translation vector and I is an $n - by - 1$ vector with elements 1.

Author(s)

C. J. Wong <cwon2@fhcrc.org>

See Also

`gpaSet`

Examples

```
## Example 1
x <- matrix(runif(20), nrow=10, ncol=2)+ 1.4
s <- matrix(c(cos(60), -sin(60), sin(60), cos(60)),
            nrow=2, ncol=2, byrow=TRUE)
xbar <- 2.2 *(x %*% s) - 0.1

lt <- iProcrustes(x, xbar, translate=TRUE) ## return linear transformation
lt

## showing result
I <- matrix(1, nrow=nrow(x), ncol=1)
tx <- x - I %*% lt$T
## get the transformed matrix xnew
xnew <- lt$scal * (tx %*% lt$Q)

if (require(lattice)) {
   xyplot(V1 ~ V2,
          do.call(make.groups, lapply(list(x=x, xbar=xbar, T.xbar=lt$T.xbar,
                  xnew=xnew),as.data.frame)),
          group=which, aspect=c(0.7), pch=c(1,3,2,4), col.symbol="black",
  main=("Align the points in x to xbar"),
          key=list(points=list(pch=c(1,3,2,4), col="black"), space="right",
                   text=list(c("x", "xbar", "T.xbar", "xnew"))))
}

## Example 2. centralized x and xbar prior to using iProcrustes
x <- matrix(runif(10), nrow=5, ncol=2)
s <- matrix(c(cos(60), -sin(60), sin(60), cos(60)),
            nrow=2, ncol=2, byrow=TRUE)
xbar <- 1.2 *(x %*% s) - 2
I <- matrix(1, nrow=nrow(x), ncol=1)
x <- x-(I %*% colMeans(x)) ## shift the centroid of points in x to the origin
xbar <- xbar - (I %*% colMeans(xbar)) ## shift centroid to the origin
lt <- iProcrustes(x, xbar, translate=FALSE) ## return linear transformation
## only return the rotation/reflection matrix and scalling factor
lt

xnew=lt$scal *(x %*% lt$Q) ## transformed matrix aligned to centralized xbar
if (require(lattice)) {
    xyplot(V1 ~ V2,
           do.call(make.groups, lapply(list(x=x,xbar=xbar,
                   xnew=xnew), as.data.frame)),
           group=which, auto.key=list(space="right"))
}
```

`idFeaturesByBackgating` *(Internal use only) Identify features of flow cytometry data using*

Description

Identify and labeling significant features using divisive clustering method such as `diana`.

Usage

```
idFeaturesByBackgating(bg, nDim, thres.sigma=2.5, lambda=0.1,
                       reference.method="median",
                       plot.workflow=FALSE, ask=names(dev.cur())!="pdf")
```

Arguments

bg	A data frame containing subpopulations on channels of interests. Must be a returning result from flowStats:::backGating
nDim	An integer indicating the length of channels of interest.
thres.sigma	An numerical value indicating the threshold at which to cut tree, e.g., as resulting from 'diana', into several clusters.
lambda	A numerical value indicating the percentage of the potential features that is used as a threshold for deciding outlier clusters. The default value is 0.1.
reference.method	A character vector indicating the method for computing the reference features. If median, the reference feature is defined by the medain of eac cluster of features. Valid methods include median and mean only.
plot.workflow	Logical. If TURE, display the workflow of feature identification.
ask	Logical. If TRUE, the display operates in interactive mode.

Details

Using the resulting data frame from backGating as potential features, the algorithm follows four major steps: (i) centering the potential features, which yields the returning value TransMatrix, (ii) using diana to compute a clustering of the potential features, (iii) cutting the tree into several clusters, and (iv) accessing outliers and rendering the final registered features with labels.

In step three, the threshold for cutting the tree is computed by

$$sd * thres.sigma,$$

where sd is the standard deviation of the distribution of the height between entities computed by diana.

A cluster is determined as an outlier if the number of its members is less than the median of the numbers of all clusters' members times 'lambda'.

Value

register	A list containing registered features for each sample.

Author(s)

Chao-Jen Wong

See Also

diana, BackGating, gpaSet.

Examples

```
## Not run:
data(ITN)
wf <- workFlow(ITN)
tl <- transformList(colnames(ITN)[3:7], asinh, transformationId="asinh")
dat <- trnasformList(ITN, tl)
bg <- backGating(dat, xy=c("FSC", "SSC"), channels="CD3")

## End(Not run)

data(BackGating)
results <- flowStats:::idFeaturesByBackgating(bg=BackGating, nDim=2,
                        plot.workflow=TRUE, ask=TRUE)
```

---

landmarkMatrix *Compute and cluster high density regions in 1D*

Description

This functions first identifies high-density regions for each `flowFrame` in a `flowSet` and subsequently tries to cluster these regions, yielding the landmarks matrix that needs to be supplied to `landmarkreg`. The function is considered to be internal.

Usage

```
landmarkMatrix(data, fres, parm, border=0.05, peakNr=NULL, densities =
NULL, n = 201, indices=FALSE)
```

Arguments

`data`	A `flowSet`.
`fres`	A list of `filterResultList` objects generated by a filtering opration using a `curv1Filter`. Each list item represents the results for one of the flow parameters in `parm`.
`parm`	Character scalar of flow paramater to compute landmarks for.
`border`	A numeric in `[0,1]`. Ignore all high-density regions with mean values in the extreme percentiles of the data range.
`peakNr`	Force a fixed number of peaks.
`densities`	An optional matrix of y values of the density estimates for the `flowSet`. If this is not present, density estimates will be calculated by the function.
`n`	Number of bins used for the density estimation.
`indices`	Return matrix of population indices instead of landmark locations. These indices can be used to point into the populations identified by the curv1Filter.

Details

In order to normalize the data using the `landmarkreg` function in the `fda`, a set of landmarks has to be computed for each `flowFrame` in a `flowSet`. The number of lansmarks has to be the same for each frame. This function identifies high-density regions in each frame, computes a simple clustering and returns a matrix of landmark locations. Missing landmarks of individual frames are substituted by the mean landmark location of the respective cluster.

Value

A matrix of landmark locations. Columns are landmarks and rows are `flowFrames`.

Author(s)

Florian Hahne

See Also

`landmarkreg`,`warpSet`

Examples

```
data(GvHD)
tmp <- list("FSC-H"=filter(GvHD[1:3], curv1Filter("FSC-H")))
res <-  flowStats:::landmarkMatrix(GvHD[1:3], tmp, "FSC-H")
```

---

lymphFilter-class *Automated gating of elliptical cell populations in 2D.*

Description

Cell populations of roughly elliptical shape in two-dimensional projections are of huge interest in many flow cytometry applications. This function identifies a single such population, potentially from a mixture of multiple populations.

Usage

```
lymphGate(x, channels, preselection=NULL, scale=2.5,  bwFac=1.3,
          filterId="defaultLymphGate", evaluate=TRUE, plot=FALSE, ...)

lymphFilter(channels, preselection=as.character(NULL),
            scale=2.5, bwFac=1.3, filterId="defaultLymphFilter")
```

Arguments

`x`	An object of class `flowSet`.
`channels`	A character vector of length 2 of valid flow parameters in `x`.
`preselection`	Either `NULL`, in which case this boils down to fitting a regular `norm2Filter`, a character scalar giving one of the flow parameters in `x`, or a named list of numerics specifying the initial rough preselection. The latter gets passed on to `rectangleGate`, see it's documentation for details.
`scale`	The `scaleFactor` parameter that gets passed on to `norm2Filter`.
`bwFac`	The bandwidth factor that gets passed on to `curv1Filter`.
`filterId`	A character used as filterId.
`evaluate`	A logical indicating wether the filter should be resolved (computation of the `filterResult` and the subset).
`plot`	Logical. Produce plots of filter results
`...`	Additional arguments.

Details

This algorithm does not apply real mixture modelling, however it is able to identify a single elliptical cell population from a mixture of multiple such populations. The idea is to first define a rough rectangular preselection and, in a second step, fit a bivariate normal distribution to this subset only.

Depending on the value of `preselection`, the initial rough selection is either

NULL: No preselection at all

character scalar Preselection based on cells that are positive for a single marker only. This allows for back-gating, for instances by selecting `CD4+` T-cells and using this information to back-gate lymphocytes in `FSC` and `SSC`. Positive cells are identified using a `curv1Filter`.

a named list of numerics: Preselection by a rectangular gate. The items of the list have to be numerics of length one giving the gate boundaries in the respective dimensions.

The `lymphFilter` class and constructor provide the means to treat `lymphGates` as regular `flowCore` filters.

Value

A list with items

`x` The filtered `flowSet`.

`n2gate` The `norm2Filter` object.

`n2gateResults` The `filterResult` after applying the `norm2Filter` on the `flowSet`.

for the `lymphGate` function. Note that `x` and `n2gateResults` are `NULL` when `eval=FALSE`.

The `lymphFilter` constructor returns and object of class `lymphFilter`, which can be used as a regular `flowCore` fitler.

Extends

Class `parameterFilter`, directly.

Class `concreteFilter`, by class "parameterFilter", distance 2.

Class `filter`, by class "parameterFilter", distance 3.

Slots

See `Arguments` section for details.

`preselection`**:** Object of class `character`, the name of the flow parameter used for preselection.

`rectDef`**:** Object of class `list`, the initial rectangular selection.

`scale`**:** Object of class `numeric`.

`bwFac`**:** Object of class `numeric`.

`parameters`**:** Object of class `parameters`, the flow parameters to operate on.

`filterId`**:** Object of class `"character"`, the filter identifier.

Objects from the Class

Objects can be created by calls of the form `new("lymphFilter", parameters, ...)` or using the constructor `lymphFilter`. The constructor is the recommended way of object instantiation.

Methods

%in% `signature(x = "flowFrame", table = "lymphFilter")`: the work horse for doing the actual filtering. Internally, this simply calls the `lympghGate` function.

Author(s)

Florian Hahne

See Also

`norm2Filter`, `curv1Filter`

Examples

```
data(GvHD)
dat <- GvHD[pData(GvHD)$Patient==10]
dat <- transform(dat, "FL4-H"=asinh(`FL4-H`))
lg <- lymphGate(dat, channels=c("FSC-H", "SSC-H"), preselection="FL4-H",scale=1.5)

if(require(flowViz))
xyplot(`SSC-H`~`FSC-H`, dat, filter=lg$n2gate)

## This is using the abstract lymphFilter class instead
lf <- lymphFilter(channels=c("FSC-H", "SSC-H"), preselection="FL4-H")
filter(dat, lf)
```

---

normQA *Normalization quality assessment*

Description

Create QA plots for a flow cytometry normalization process.

Usage

```
normQA(data, morph = c("^fsc", "^ssc"), channels, odat = NULL, ask = names(dev.c
```

Arguments

`data`	a normalized `flowSet`.
`morph`	A character vector of channel names to use for the backgating into the morphological channels.
`channels`	The channels for which to create plots. Defaults to all normalized channels.
`odat`	The original data set, always needed if there are no warping functions available.
`ask`	Ask before creating a new plot.
`grouping`	A grouping variable in `data`'s `phenoData` slot.
`tag.outliers`	Logical. Add sample name to outliers in the plots.
`peaksOnly`	Logical. Only use data when a peak was detected in a particular sample. If set to `FALSE`, a average peak location is estimated.

Details

This function assumes that the necessary information has been added as attributes to `data` during the normalization procedure. Depending on the available information, a set of QA plots is generated. Available plots are:

Amount of peak adjustment

Warping functions

Landmark classification confidence

Backgating of peak events in morphological channels

Value

This function is called for its side effect of generating plots.

Author(s)

Florian Hahne

`plotBins` *Plots the probability bins overlaid with flowFrame data*

Description

This function is useful in visualizing the differences between the binned control and sample datasets. The bins generated from the control dataset are overlaid with the sample dataset. An optional argument residuals can be used to shade each bin based on a calculated statistical measure of difference between the number of events in each bin.

Usage

```
plotBins(binRes,data,channels,title,residuals,shadeFactor)
```

Arguments

`binRes`	The result generated by calling the `probBin` function on a control dataset.
`data`	An object of class `flowFrame` sample(dataset)
`channels`	The flow parameters to be plotted.In cases where more than two parameters are binned from the control set, the `plotBins` function plots the projections of the hyperplanes in 2 dimensions)
`title`	Optional title for the plot generated
`residuals`	A vector of length equal to the number of bins generated that can be used to shade each bin. The residuals from the `calcPearsonChi` function or the `calcPBChiSquare` function can be used to highlight the bins that are different between control and sample datasets
`shadeFactor`	Optional argument between 0 and 1 that controls the intensity of the shading of bins

Author(s)

Nishant Gopalakrishnan

See Also

proBin, calcPearsonChi, calcPBChiSquare

Examples

```
data(GvHD)
# flow frame 1 is treated as  control dataset and used to generate bins
resCtrl<-proBin(GvHD[[1]],200,channels=c("FSC-H","SSC-H"))
plotBins(resCtrl,GvHD[[1]],channels=c("FSC-H","SSC-H"),title="Binned control data")
# Same bins are applied to flowFrame 16
resSample<-binByRef(resCtrl,GvHD[[16]])
stat<-calcPearsonChi(resCtrl,resSample)
dev.new()
plotBins(resCtrl,data=GvHD[[16]],channels=c("FSC-H","SSC-H","Time"),title="Comparision 1
residuals=stat$residuals[2,],shadeFactor=0.7)
```

---

proBin *Probability binning - a metric for evaluating multivariate differences*

Description

This function divides the flowframe events into bins such that each bin contains the same number of events. The number of events falling into each bin can then be compared across the control and test samples using statistical methods such as the Chi-squared test.

Usage

```
proBin(m, minEvents=500,channels=NULL)
```

Arguments

m	An object of class flowFrame
minEvents	The minEvents The minimum number of events in each bin. (i.e. the termination criterion for the probability binning algorithm)
channels	A character vector for the Flourescence channels on which probability binning is to be performed. Defaults is NULL, in which case, all flourescence channels are used for probability binning.(Time information, if provided in the flowFrame is discarded)

Details

The flowSet is first filtered using a rectangleGate and the norm2Filter is subsequently fitted to the remaining subset.

Value

A list with items:

`table` A `data.frame` that stores information regarding each node of the tree generated during each stage of the probability binning algorithm. Each row in the table represents a node, the first row representing the original `flowFrame` matrix.

The dataIndx column provides indexes for retrieving the matrices during each stage of the binning process from the enviroment `data` .

The parent field indicates the row number in the table that holds the parent information for the corresponding node.

The left and right columns indicates the row numbers in the table that stores information regarding the children of that particular node. The leaf nodes that hold the binned data can be identified by the nodes with the left of right values of zero(ie. no children nodes)

The visited column is used internally by the algorithm to check if a particular node has been visited during the computation process.

`data` An enviroment in which the matrices generated during each stage of the probability binning process is stored. The matrices stored at the leaf nodes represent the binned events obtained after the stop criterion of minEvents has been achieved. These can be identified by the corresponding dataIndx fields provided by the rows in the table with the left or right column values of zero.

`limits` A list containing the the boundaries of each hyperplane generated during probability binning

`splitPars` A `data.frame` containing two columns splitCol - indicates the column number of the `flowFrame` , the split was performed.

splitMed - The median value which was used as the threshold for splitting the `flowFrame`

The splitCol and splitMed parameters are utilized by the plotBins and shadeBins functions in visualizing the differences between control and test sample cases.

Author(s)

Nishant Gopalakrishnan

See Also

`plotBins`, `binByRef`

Examples

```
data(GvHD)
res<-proBin(GvHD[[1]],200,channels=c("FSC-H","SSC-H","FL1-H","FL4-H"))
```

quadrantGate *Automated quad gating*

Description

This function tries to find the most likely separation of two-dimensional flow cytometry in four quadrants.

Usage

```
quadrantGate(x, stains, alpha=c("min", "min"), sd=c(2, 2), plot=FALSE,
    filterId="defaultQuadGate", refLine.1=NULL, refLine.2=NULL, ...)
```

Arguments

x	A flowSet or flowFrame.
stains	A character vector of length two giving the two flow parameters for which the quad gate is to be computed.
alpha, sd	Tuning factors to control the computation of the gate boundaries. See rangeGate for details.
plot	Logical. Produce plots of intermediate results.
filterId	Character, the name assigned to the resulting filter.
refLine.1	Either NULL or a numeric of lenth 1. If NULL, this parameter is ignored. When it is set to a numeric, the minor sub-population (if any) below this reference line in the first stain channel will be igored while determining the separator between positive and negative.
refLine.2	Either NULL or a numeric of lenth 1. If NULL, this parameter is ignored. When it is set to a numeric, the minor sub-population (if any) below this reference line in the second stain channel will be igored while determining the separator between positive and negative.
...	Additional arguments

Details

The most likely separation between postitive and negative stains for two-dimensional data is computed based on density estimates. Essentially, the gate parameters are first fitted separately for the two parameters and later combined. See the documentation for rangeGate for details. There is a certain amount of heuristics involved in this process. The algorithm can be slightly tweaked using the alpha and sd arguments. Their values will be recycled for the two dimensions unless explicitely given as vectors of length 2.

Value

An object of class quadGate.

Author(s)

Florian Hahne

See Also

quadGate, rangeGate

Examples

```
data(GvHD)
dat <- GvHD[pData(GvHD)$Patient==10]
dat <- transform(dat, "FL4-H"=asinh(`FL4-H`), "FL2-H"=asinh(`FL2-H`))
qg <- quadrantGate(dat, c("FL2-H", "FL4-H"))
qg

if(require(flowViz))
xyplot(`FL2-H`~`FL4-H`, dat, filter=qg)

qg <- quadrantGate(dat, c("FL2-H", "FL4-H"), alpha=c(0.1, 0.9), plot=TRUE)
qg
split(dat, qg)
```

---

rangeGate *Find most likely separation between positive and negative populations*

Description

The function tries to find a reasonable split point between the two hypothetical cell populations "positive" and "negative".

Usage

```
rangeGate(x, stain, alpha="min", sd=2, plot=FALSE, borderQuant=0.1,
absolute=TRUE, filterId="defaultRectangleGate", positive=TRUE,
refLine=NULL, ...)

rangeFilter(stain, alpha="min", sd=2, borderQuant=0.1,
filterId="defaultRangeFilter")
```

Arguments

x	A flowSet or flowFrame.
stain	A character scalar giving the flow parameter for which to compute the separation.
alpha	A tuning parameter that controls the location of the split point between the two populations. This has to be a numeric in the range [0,1], where values closer to 0 will shift the split point closer to the negative population and values closer to 1 will shift towards the positive population. Additionally, the value of alpha can be "min", in which case the split point will be selected as the area of lowest local density between the two populations.
sd	For the case where there is only a single population, the algorithm falls back to esitmating the mode of this population and a robust measure of the variance of it distribution. The sd tuning parameter controls how far away from the mode the split point is set.

plot	Create a plot of the results of the computation.
borderQuant	Usualy the instrument is set up in a way that the positive population is somewhere on the high end of the measurement range and the negative population is on the low end. This parameter allows to disregard populations with mean values in the extreme quantiles of the data range. It's value should be in the range [0,1].
absolute	Logical controling whether to classify a population (positive or negative) relative to the theoretical measurment range of the instrument or the actual range of the data. This can be set to TRUE if the alignment of the measurment range is not optimal and the bulk of the data is on one end of the theoretical range.
filterId	Character, the name assigned to the resulting filter.
positive	Create a range gate that includes the positive (TRUE) or the negative (FALSE) population.
refLine	Either NULL or a numeric of lenth 1. If NULL, this parameter is ignored. When it is set to a numeric, the minor sub-population (if any) below this reference line will be igored while determining the separator between positive and negative.
...	Further arguments.

Details

The algorithm first tries to identify high density regions in the data. If the input is a flowSet, density regions will be computed on the collapsed data, hence it should have been normalized before (see warpSet for one possible normalization technique). The high density regions are then clasified as positive and negative populations, based on their mean value in the theoretical (or absolute if argument absolute=TRUE) measurement range. In case there are only two high-density regions the lower one is usually clasified as the negative populations, however the heuristics in the algorithm will force the classification towards a positive population if the mean value is already very high. The absolute and borderQuant arguments can be used to control this behaviour. The split point between populations will be drawn at the value of mimimum local density between the two populations, or, if the alpha argument is used, somewhere between the two populations where the value of alpha forces the point to be closer to the negative (0 - 0.5) or closer to the positive population (0.5 - 1).

If there is only a single high-density region, the algorithm will fall back to estimating the mode of the distribution (hubers) and a robust measure of it's variance and, in combination with the sd argument, set the split point somewhere in the right or left tail, depending on the classification of the region.

For more than two populations, the algorithm will still classify each population into positive and negative and compute the split point between those clusteres, similar to the two population case.

The rangeFilter class and constructor provide the means to treat rangeGate as regular flowCore filters.

Value

A range gate, more explicitely an object of class rectangleGate.

Methods

%in% signature(x = "flowFrame", table = "rangeFilter"): the work horse for doing the actual filtering. Internally, this simply calls the rangeGate function.

Author(s)

Florian Hahne, Kyongryun Lee

See Also

warpSet, rangeGate, rectangleGate

Examples

```
data(GvHD)
dat <- GvHD[pData(GvHD)$Patient==10]
dat <- transform(dat, "FL4-H"=asinh(`FL4-H`), "FL3-H"=asinh(`FL3-H`))
rg <- rangeGate(dat, "FL4-H", plot=TRUE)
rg
split(dat, rg)

## Test rangeGate when settting refLine=0; it does not do anything since
## there is no sub-population below zero.
rangeGate(dat, "FL4-H", plot=FALSE, refLine=0)

rf <- rangeFilter("FL4-H")
filter(dat, rf)
```

---

warpSet *Normalization based on landmark registration*

Description

This function will perform a normalization of flow cytometry data based on warping functions computed on high-density region landmarks for individual flow channels.

Usage

```
warpSet(x, stains, grouping = NULL, monwrd = TRUE, subsample=NULL,
        peakNr=NULL, clipRange=0.01, nbreaks=11, fres, bwFac=1.3,
        warpFuns=FALSE, ...)
```

Arguments

x	A flowSet.
stains	A character vector of flow parameters in x to be normalized.
grouping	A character indicating one of the phenotypic variables in the phenoData slot of x used as a grouping factor. The within-group and between-group variance is computed and a warning is issued in case the latter is bigger than the former, indicating the likely removal of signal by the normalization procedure.
monwrd	Logical. Compute strictly monotone warping functions. This gets directly passed on to landmarkreg.
subsample	Numeric. Reduce the number of events in each flowSet by sub sampling for all density estimation steps and the calculation of the warping functions. This can increase computation time for large data sets, however it might reduce the accuracy of the density estimates. To be used with care.

peakNr	Numeric scalar. Force a fixed number of peaks to use for the normalization.
clipRange	Only use peaks within a clipped data range. Essentially, the number indicates the percent of clipping on both sides of the data range, e.g. `min(x) - 0.01 * diff(range(x))`.
nbreaks	The number of spline sections used to approximate the data. Higher values produce more accurate results, however this comes with the cost of increaseqd computing times. For most data, the default setting is good enough.
fres	A named list of `filterResultList` objects. This can be used to speed up the process since the `curv1Filter` step can take quite some time.
bwFac	Numeric of lenght 1 used to set the bandwidth factor by `curv1Filter` for smoothing of the density estimate.
warpFuns	Logical indcating whether to return the normalized `flowSet` or a list of warping functions.
...	Further arguments that are passed on to `landmarkreg`.

Details

Normalization is achived by first identifying high-density regions (landmarks) for each `flowFrame` in the `flowSet` for a single channel and subsequently by computing warping functions for each `flowFrame` that best align these landmarks. This is based on the algorithm implemented in the `landmarkreg` function in the `fda` package. An intermediate step classifies the high-density regions, see `landmarkMatrix` for details.

Please note that this normalization is on a channel-by-channel basis. Multiple channels are normalized in a loop.

Value

The normalized `flowSet` if `warpFuns` is `FALSE`, otherwise a list of warping functions. Additional inforamtion is attached as the `warping` attribute to the `flowSet` in form of a list.

Note

We currently use a patched fda version.

Author(s)

Florian Hahne

References

J.O. Ramsay and B.W. Silverman: Applied Functional Data Analysis, Springer 2002

See Also

`curv1Filter`, `landmarkMatrix`

Examples

```
data(ITN)
dat <- transform(ITN, "CD4"=asinh(CD4), "CD3"=asinh(CD3), "CD8"=asinh(CD8))
lg <- lymphGate(dat, channels=c("CD3", "SSC"),
preselection="CD4",scale=1.5)
dat <- Subset(dat, lg$n2gate)
datr <- warpSet(dat, "CD8", grouping="GroupID", monwrd=TRUE)
if(require(flowViz)){
d1 <- densityplot(~CD8, dat, main="original", filter=curv1Filter("CD8"))
d2 <- densityplot(~CD8, datr, main="normalized", filter=curv1Filter("CD8"))
plot(d1, split=c(1,1,2,1))
plot(d2, split=c(2,1,2,1), newpage=FALSE)
}
```

Index