Package ‘stepNorm’

November 21, 2016

Version 1.46.0
Date 2008-10-08
Title Stepwise normalization functions for cDNA microarrays
Author Yuanyuan Xiao <yxiao@itsa.ucsf.edu>, Yee Hwa (Jean) Yang <jean@biostat.ucsf.edu>
Depends R (>= 1.8.0), marray, methods
Imports marray, MASS, methods, stats
Maintainer Yuanyuan Xiao <yxiao@itsa.ucsf.edu>
Description Stepwise normalization functions for cDNA microarray data.
License LGPL
URL http://www.biostat.ucsf.edu/jean/
biocViews Microarray, TwoChannel, Preprocessing
NeedsCompilation no

R topics documented:
calcAIC .................................................. 2
calcBIC .................................................. 3
fit2DWithin ............................................. 4
fitWithin ................................................ 6
maCompPlate2 ........................................... 8
makeStepList .......................................... 10
marrayFit-class ...................................... 11
seqWithinNorm ......................................... 13
stepWithinNorm ....................................... 15
withinNorm ............................................ 18

Index 20
calcAIC

Extract AIC from a Fitted Model

Description
Computes the Akaike Information Criterion for a fitted parametric model.

Usage
calcAIC(fit, subset=TRUE, scale = 0, enp, loss.fun = square)

Arguments
- fit: fitted model; see details below
- scale: optional numeric specifying the scale parameter of the model; see scale in step.
- subset: A "logical" or "numeric" vector indicating the subset of points used to compute the fitted model.
- enp: equivalent number of parameters in the fitted model. If missing, the enp component from fit will be used.
- loss.fun: the loss function used to calculate deviance; default uses the squared deviations from the fitted values; one could also use, for example, absolute deviations (abs).

Details
The argument fit can be an object of class marrayFit, in which case the residuals component from the marrayFit object will be extracted to calculate the deviance; the user can also pass in a numeric vector, in which case it will be interpreted as the residuals and the user needs to specify the argument enp.

The criterion used is

\[ AIC = -2 \times \log L + k \times \text{enp}, \]

where \( L \) is the likelihood and enp the equivalent number of parameters of fit. For linear models (as in marrayFit), \(-2\log L\) is computed from the deviance.

\( k = 2 \) corresponds to the traditional AIC and is the penalty for the number of parameters.

Value
A numeric vector of length 4, giving
- Dev: the deviance of the fit.
- enp: the equivalent number of parameters of the fit.
- penalty: the penalty for number of parameters.
- Criterion: the Akaike Information Criterion for fit.

Author(s)
Yuanyuan Xiao, <yxiao@itsa.ucsf.edu>,
Jean Yee Hwa Yang, <jean@biostat.ucsf.edu>
calcBIC

Extract BIC from a Fitted Model

Description
Computes the Bayesian Information Criterion for a fitted parametric model.

Usage
calcBIC(fit, subset=TRUE, scale = 0, enp, loss.fun = square)

Arguments
- fit: fitted model; see details below
- subset: A "logical" or "numeric" vector indicating the subset of points used to compute the fitted model.
- scale: optional numeric specifying the scale parameter of the model; see scale in step.
- enp: equivalent number of parameters in the fitted model. If missing, the enp component from fit will be used.
- loss.fun: the loss function used to calculate deviance; the default uses the squared deviation from the fitted values; one could also use absolute deviations (abs).

Details
The argument fit can be an object of class marrayFit, in which case the residuals component from the marrayFit object will be extracted to calculate the deviance; the user can also pass in a numeric vector, in which case it will be interpreted as the residuals and the user needs to specify the argument enp.

The criterion used is

$$BIC = -2 \times \log L + k \times enp,$$

where $L$ is the likelihood and $enp$ the equivalent number of parameters of fit. For linear models (as in marrayFit), $-2 \log L$ is computed from the deviance.

$k = \log(n)$ corresponds to the BIC and is the penalty for the number of parameters.
**Value**

A numeric vector of length 4, giving

- **Dev** the deviance of the fit.
- **enp** the equivalent number of parameters of the fit.
- **penalty** the penalty for number of parameters.
- **Criterion** the Akaike Information Criterion for fit.

**Author(s)**

Yuanyuan Xiao, <yxiao@itsa.ucsf.edu>,
Jean Yee Hwa Yang, <jean@biostat.ucsf.edu>

**See Also**

AIC, deviance, calcAIC.

**Examples**

```r
## load in swirl data
data(swirl)

## fit a model
fit <- fitWithin(fun="medfit")
## res is an object of class marrayFit
res <- fit(swirl[,1])

## calculate BIC
calcBIC(res)
## or could pass in the residual vector, but then argument "enp" needs to be specified
calcBIC(res$residual, enp=1)
```

---

**Description**

This function performs 2D location normalization on cDNA microarray. It operates on class marrayRaw or class marrayNorm. It allows the user to choose from a set of four basic normalization procedures.

**Usage**

```r
fit2DWithin(x1.fun = "maSpotRow", x2.fun = "maSpotCol", y.fun = "maM", subset=TRUE, fun = aov2Dfit, ...)
```
Arguments

x1.fun Name of accessor method for spot row coordinates, usually maSpotRow.
x2.fun Name of accessor method for spot column coordinates, usually maSpotCol.
y.fun Name of accessor method for spot statistics, usually the log-ratio maM.
subset A "logical" or "numeric" vector indicating the subset of points used to compute the normalization values.
fun Character string specifying the normalization procedures:
rlm2Dfit for robust linear regression using the rlm function
loess2Dfit for robust local regression using the loess function
aov2Dfit for linear regression using the lm function
spatialMedfit for spatial median normalization
... Misc arguments for fun

Details

The spot statistic named in y is regressed on spot row and column coordinates, using the function specified by the argument fun. Typically, rlm2Dfit and loess2Dfit, which treat row and column coordinates as numeric vectors, require a lot fewer parameters than aov2Dfit which specifies these two variables as categorical. spatialMedfit could yet fit the most complicated model, depending on size of the smoothing window specified; details see Wilson et al (2003).

Value

The function fit2DWithin returns a function (F) with bindings for x1.fun, x2.fun, y.fun, subset and fun. When the function F is evaluated with an object of class marrayNorm or marrayRaw, it carries out normalization and returns an object of class marrayFit that contains the normalization information as a list with the following components:

varfun : A character vector of names of predictor variables.
x : A numeric matrix of predictor variables.
y : A numeric matrix of responses.
residuals : A numeric matrix of normalized values (typically log ratios (M)).
fitted : A numeric matrix of the fitted values.
enp : The equivalent number of parameters; see loess.
df.residual : The residual degrees of freedom.
fun : A character string indicating the name of the function used for normalization.

Note that the residuals component stores the normalized ratios.

Author(s)

Yuanyuan Xiao, <yxiao@itsa.ucsf.edu>,
Jean Yee Hwa Yang, <jean@biostat.ucsf.edu>

References


See Also

fitWithin

Examples

```r
## use the swirl data as example
data(swirl)

## 2D rlm normalization
rlm2D <- fit2DWithin(fun="rlm2Dfit")
swirl1.rlm <- rlm2D(swirl[,1])
norm.M <- swirl1.rlm$residuals ## matrix of normalized ratios

## 2D loess normalization, default span=0.2
loess2D <- fit2DWithin(fun="loess2Dfit")
swirl1.loess <- loess2D(swirl[,1])
## 2D loess normalization, span=0.4
## Not run:
loess2D.1 <- fit2DWithin(fun="loess2Dfit", span=0.4)
swirl1.loess.1 <- loess2D.1(swirl[,1])
## End(Not run)

## 2D aov normalization
aov2D <- fit2DWithin(fun="aov2Dfit")
swirl1.aov <- aov2D(swirl[,1])

## 2D spatial median normalization, default window width=3
spatialMed2D <- fit2DWithin(fun="spatialMedfit")
swirl1.spatialMed <- spatialMed2D(swirl[,1])
## 2D loess normalization, window width=9
## Not run:
spatialMed2D.1 <- fit2DWithin(fun="spatialMedfit", width=9)
swirl1.spatialMed.1 <- spatialMed2D.1(swirl[,1])
## End(Not run)
```

fitWithin

Simple location normalization function for cDNA microarray data

Description

This function performs location normalization on cDNA microarray. It operates on class `marrayRaw` or class `marrayNorm`. It allows the user to choose from a set of three basic normalization procedures.

Usage

```r
fitWithin(x.fun = "maA", y.fun = "maM", z.fun = TRUE, subset=TRUE, fun = "medfit", ...)
```

Arguments

- `x.fun` Name of accessor method for spot intensity, usually `maA`.
- `y.fun` Name of accessor method for spot statistics, usually the log-ratio `maM`. 

**fitWithin**

z.fun Name of accessor method for spot statistic used to stratify the data, usually a layout parameter, e.g. `maPrintTip` or `maCompPlate`. If `z` is not a character, e.g. `NULL`, the data are not stratified.

subset A "logical" or "numeric" vector indicating the subset of points used to compute the normalization values.

fun Character string specifying the normalization procedure:

- `medfit` for global median location normalization
- `rlmfit` for global intensity or A-dependent location normalization using the `rlm` function
- `loessfit` for global intensity or A-dependent location normalization using the `loess` function

... Miscs arguments to be passed in `fun`

Details

Normalization is typically performed on the expression ratios of cDNA microarray data, using the function specified by argument `fun`. Currently, this function is to be chosen from: `medfit` (median), `rlmfit` (rlm) and `loessfit` (loess). When `z.fun` is provided as a character string, for example, `maPrintTip`, the normalization procedure is operated within each print-tip of the slide.

Value

The function `fitWithin` returns a function(`F`) with bindings for `x.fun`, `y.fun`, `z.fun`, `subset` and `fun`. When the function `F` is evaluated with an object of class `marrayNorm` or `marrayRaw`, it carries out normalization and returns an object of class `marrayFit` that contains the normalization information as a list with the following list components:

- `varfun` : A character vector of names of predictor variables.
- `x` : A numeric matrix of predictor variables.
- `y` : A numeric matrix of responses.
- `residuals` : A numeric matrix of normalized values (typically log ratios (`M`)).
- `fitted` : A numeric matrix of the fitted values.
- `enp` : The equivalent number of parameters; see `loess`.
- `df.residual` : The residual degrees of freedom.
- `fun` : A character string indicating the name of the function used for normalization.

Note that the `residuals` component stores the normalized ratios.

Author(s)

Yuanyuan Xiao, <yxiao@itsa.ucsf.edu>,
Jean Yee Hwa Yang, <jean@biostat.ucsf.edu>

References

See Also

fit2DWithin

Examples

```r
## using the swirl data as example
data(swirl)

## median normalization
med <- fitWithin(fun="medfit")
swirl1.med <- med(swirl[,1])
norm.M <- swirl1.med$residuals ## matrix of normalized ratios

## rlm normalization
rlmF <- fitWithin(fun="rlmfit")
swirl1.rlm <- rlmF(swirl[,1])

## loess normalization, default span=0.4
loessF <- fitWithin(fun="loessfit")
swirl1.loess <- loessF(swirl[,1])

## loess normalization, span=0.2
loessF.1 <- fitWithin(fun="loessfit", span=0.2)
swirl1.loess.1 <- loessF.1(swirl[,1])

## within-printtip loess normalization
loessP <- fitWithin(z.fun="maPrintTip", fun="loessfit")
swirl1.loessP <- loessP(swirl[,1])
```

---

**maCompPlate2**  
*Generate plate IDs*

**Description**

This function is a modification of the `maCompPlate` function in the `marray` library. It generates plate IDs from the dimensions of the grid and spot matrices. Unlike the `maCompPlate` function, the number of spots is not necessarily a multiple of the number of wells on a plate, therefore this function allows empty spots on the slide.

**Usage**

```r
maCompPlate2(no.plates = NULL, n = 384)
```

**Arguments**

- `no.plates`  
  object of class "numeric", number of plates used specified by the user. If a number is not specified, then it is assumed that there are no empty spots on the slide.

- `n`  
  object of class "numeric", number of wells in each plate, usually 384 or 96.
maCompPlate2

Details

This function can be used to handle three cases: 1) the number of spots is a multiple of the number
of wells on a plate (usually 96 or 384); 2) the number of spots is not a multiple of the number
of wells on a plate, and several of spots on the slide are therefore left empty. In this case, the user
needs to specify the number of total plates used; plate IDs of empty spots will be NAs; 3) the number
of spots is not a multiple of the number of wells on a plate, but all spots on the slide are spotted,
therefore there is one plate not fully used. In this case, the user does not need to specify the number
of total plates (as this will not be an integer), the function assumes no empty spots on the slide
automatically. See Examples below.

Value

The function maCompPlate2 returns a function with bindings for no.plates and n, which when
receiving a object of marrayRaw, marrayNorm or marrayLayout class, it returns a vector of plate
IDs (factor).

Author(s)

Yuanyuan Xiao

See Also

maCompPlate, marrayLayout.

Examples

####### case 1: no empty spots on the slide, full plates used
L<-new("marrayLayout", maNgr=4, maNgc=4, maNsr=22, maNsc=24)
### "compPlate" is a function
compPlate <- maCompPlate2(n=384)
plate <- compPlate(L)
table(plate)
### can also use:
plate<-maCompPlate(L,384)
table(plate)

####### case 2: with empty spots on the slide, full plates used
L<-new("marrayLayout", maNgr=4, maNgc=4, maNsr=22, maNsc=26)
### "compPlate" is a function
compPlate <- maCompPlate2(no.plates=22, n=384)
plate <- compPlate(L)
table(plate)
### empty spots are NAs
unique(plate)

####### case 3: no empty spots on the slide, one plate not full
L<-new("marrayLayout", maNgr=4, maNgc=4, maNsr=22, maNsc=26)
### argument no.plates not specified, the function assumes no empty spots
compPlate <- maCompPlate2(n=384)
plate <- compPlate(L)
### 23 full plates (384), the 24th not full (304)
table(plate)
### no NAs, no empty spots
unique(plate)
makeStepList

Construction of a stepwise normalization list

Description

This function provides a user friendly way to construct a list for input to the function stepWithinNorm. The list indicates intended biases for correction and models for stepwise normalization.

Usage

makeStepList(A = c("median", "rlm", "loess"), PT = c("median", "rlm", "loess"), PL = c("median", "rlm", "loess"), Spatial2D = c("rlm2D", "loess2D", "aov2D", "spatialMedian"))

Arguments

A

A character string specifying the normalization models for the adjustment of intensity or A bias:

- median: global median location normalization
- rlm: global intensity or A-dependent robust linear normalization using the rlm function
- loess: global intensity or A-dependent robust nonlinear normalization using the loess function

The user can specify any of these three choices and the selected model will be compared based on goodness fit and model parsimony; If the correction of the $A$ bias is not desired, the user can set A = NULL.

PT

A character string specifying the normalization models for the adjustment of print-tip or PT bias:

- median: within-print-tip-group median normalization
- rlm: within-print-tip-group robust linear normalization using the rlm function
- loess: within-print-tip-group robust nonlinear normalization using the loess function
- none: no normalization for the $PT$ bias

If the correction of the SPTS bias is not desired, the user can set PT = NULL.

PL

A character string specifying the normalization models for the adjustment of well-plate or PL bias:

- median: within-well-plate median normalization
- rlm: within-well-plate robust linear normalization using the rlm function
- loess: within-well-plate robust nonlinear normalization using the loess function
- none: no normalization for the $PL$ bias

If the correction of the SPLS bias is not desired, the user can set PL = NULL.

Spatial2D

A character string specifying the normalization models for the adjustment of spatial 2D bias:

- none: no normalization for the spatial 2D bias
- aov2D: spatial bivariate location normalization using ANOVA
marrayFit-class

Class "marrayFit", storing parameters and results of post-normalization cDNA microarray data

Description

A simple list-based class for the storage of parameters and results of normalization of cDNA microarray data.

Details

This function provides a user friendly way to specify the parameter wf.loc for the main stepwise normalization function stepWithinNorm; see examples for details.

Value

An object of class "list" for input to the stepWithinNorm function.

Author(s)

Yuanyuan Xiao, <yxiao@itsa.ucsf.edu>,
Jean Yee Hwa Yang, <jean@biostat.ucsf.edu>

See Also

stepWithinNorm.

Examples

# Examples use swirl dataset, for description type ? swirl
data(swirl)

# To use the default parameters, which adjusts A, PT, PL and Spatial2D
# biases sequentially and compares all models available, simple type
wf.loc <- makeStepList()

# To apply loess for the A bias, and to omit the Spatial2D step
wf.loc <- makeStepList(A="loess", Spatial2D=NULL)

# To compare only rlm and loess in the A bias step, and other biases as default
wf.loc <- makeStepList(A=c("rlm","loess"))

# input to the stepWithinNorm function
## Not run:
step.swirl1 <- stepWithinNorm(swirl[,1],wf.loc=wf.loc)
## End(Not run)
Creating Objects from the Class

Objects can be created by calls of the form `new('marrayFit', fit)` where `fit` is a list. Objects of `marrayFit` in the StepNorm package are typically created by functions `fitWithin` and `fit2DWithin`.

List Components

This class contains no slots, but objects should contain the following list components:

- `varfun` : A character vector of names of predictor variables.
- `x` : A numeric matrix of predictor variables.
- `y` : A numeric matrix of responses.
- `residuals` : A numeric matrix of normalized values (typically log ratios ($M$)).
- `fitted` : A numeric matrix of the fitted values.
- `enp` : The equivalent number of parameters; see `loess`.
- `df.residual` : The residual degrees of freedom.
- `fun` : A character string indicating the name of the function used for normalization.

Methods

This class inherits directly from class `list` so any operation appropriate for lists will work on objects of this class.

Author(s)

Yuanyuan Xiao, <yxiao@itsa.ucsf.edu>,
Jean Yee Hwa Yang, <jean@biostat.ucsf.edu>

See Also

`fitWithin`, `fit2DWithin`.

Examples

```r
## load in swirl data
data(swirl)

## median normalization for the first slide of the swirl data
medWithin <- fitWithin(fun="medfit")
## medFit is an object of class marrayFit
medFit <- medWithin(swirl[,1])
## normalized ratios is stored in:
norm.M <- medFit$residuals
```
**seqWithinNorm**

**Sequential within-slide normalization function**

**Description**

This function conducts cDNA microarray normalization in a sequential fashion. In a two-color cDNA array setting, within-slide normalization calibrates signals from the two channels to remove non-biological variation introduced by various processing steps.

**Usage**

```
seqWithinNorm(marraySet, y = "maM", subset = TRUE, loss.fun = square,
A = c("loess", "rlm", "median", "none"),
PT = c("median", "rlm", "loess", "none"),
PL = c("median", "rlm", "loess", "none"),
Spatial2D = c("none", "aov2D", "rlm2D", "loess2D", "spatialMedian"),
criterion = c("BIC", "AIC"))
```

**Arguments**

- **marraySet**: Object of class `marrayRaw` or class `marrayNorm`, containing intensity data for the batch of arrays to be normalized.
- **y**: Name of accessor method for spot statistics, usually the log-ratio `maM`.
- **subset**: A "logical" or "numeric" vector indicating the subset of points used to compute the normalization values.
- **loss.fun**: The loss function used in calculating deviance, the default uses squared sum of residuals; for absolute sum of residuals, use `abs`.
- **A**: A character string specifying the normalization model for the adjustment of intensity or A bias:
  - `loess`: global intensity or A-dependent robust nonlinear normalization using the `loess` function
  - `rlm`: global intensity or A-dependent robust linear normalization using the `rlm` function
  - `median`: global median location normalization
  - `none`: no normalization for the A bias
  If not specified, `loess` normalization will be applied.
- **PT**: A character string specifying the normalization model for the adjustment of print-tip or PT bias:
  - `median`: within-print-tip-group median normalization
  - `rlm`: within-print-tip-group robust linear normalization using the `rlm` function
  - `loess`: within-print-tip-group robust nonlinear normalization using the `loess` function
  - `none`: no normalization for the PT bias
  If not specified, median normalization within print-tip will be applied.
- **PL**: A character string specifying the normalization model for the adjustment of well-plate or PL bias:
  - `median`: within-well-plate median normalization
**seqWithinNorm**

- **rlm**: within-well-plate robust linear normalization using the `rlm` function
- **loess**: within-well-plate robust nonlinear normalization using the `loess` function
- **none**: no normalization for the \(PL\) bias

If not specified, median normalization within well-plate will be applied.

**Spatial2D**

A character string specifying the normalization model for the adjustment of spatial 2D bias:
- **none**: no normalization for the spatial 2D bias
- **aov2D**: spatial bivariate location normalization using ANOVA
- **rlm2D**: spatial bivariate location normalization using the `rlm` function
- **loess2D**: spatial bivariate location normalization using the `loess` function
- **spatialMedian**: spatial location normalization using a spatial median approach (see Wilson et al. (2003) in reference)

If not specified, no normalization will be carried out in this step.

**criterion**

Character string specifying the criterion:
- **AIC**: the AIC criterion is used; see `calcAIC`
- **BIC**: the BIC criterion is used; see `calcBIC`

If no specification, BIC is used. Note that here we don’t use the criterion to choose normalization model in each step. Criterion is calculated solely for information purpose.

**Details**

Typical systematic non-biological variations of a two-color cDNA microarray include the dependence of ratio measurements (\(M\)) on intensity (\(A\)), print-tip IDs (\(PT\)), plate IDs (\(PL\)) and spatial heterogeneity of the slide (Spatial 2D). The sequential normalization procedure in `seqWithinNorm` normalizes a slide in a sequential fashion: \(A \rightarrow PT \rightarrow PL \rightarrow Spatial2D\). In each step one kind of variation is targeted for correction, and the user chooses the normalization method as desired. We calculate the AIC/BIC criterion along the normalization steps, but they are not used for selection of models.

**Value**

An object of class "list":
- **normdata**: an object of class `marrayNorm`, containing the normalized intensity data.
- **res**: a list of the sequential normalization result for each slide within the marray dataset. Each list component is also a list containing the name of the biases, deviance, equivalent number of parameters, AIC/BIC value for a certain slide.

**Author(s)**

- Yuanyuan Xiao, <yxiao@itsa.ucsf.edu>
- Jean Yee Hwa Yang, <jean@biostat.ucsf.edu>

**References**

stepWithinNorm

### Description

This function conducts cDNA microarray normalization in a stepwise fashion. In a two-color cDNA array setting, within-slide normalization calibrates signals from the two channels to remove non-biological variation introduced by various processing steps.

### Usage

```r
stepWithinNorm(marraySet, subset=TRUE, wf.loc, criterion = c("BIC", "AIC"), loss.fun = square)
```

### Arguments

- **marraySet**: Object of class `marrayRaw` or class `marrayNorm`, containing intensity data for the batch of arrays to be normalized.
- **subset**: A "logical" or "numeric" vector indicating the subset of points used to compute the normalization values.
- **wf.loc**: Object of class `list`, each component is a step for the removal of a particular systematic variation. Typically each step is also a list of several candidate models of different complexity, the best model will be chosen by the criterion specified. For a user friendly way of constructing such a list, consult the function `makeStepList`. If missing, the default procedure will be used, which we consider appropriate for most slides. See details for how to specify the list and how it is used.

### Examples

```r
# Examples use swirl dataset, for description type ? swirl
data(swirl)

# Apply sequential normalization for the first slide
# default: loess(A)->median(PT)->median(PL)-> none (Spatial2D)
res.swirl1 <- seqWithinNorm(swirl[,1])

# normalized data
norm.swirl <- res.swirl1[[1]]

# sequential normalization information
step.swirl <- res.swirl1[[2]]

# median(A)->median(PT)->median(PL)->none(Spatial2D)
res.swirl <- seqWithinNorm(swirl[,1], A="median",PT="median",PL="median",Spatial2D="none")
```
criterion: Character string specifying the criterion used for the selection of the best normalization procedure in each step. This argument can be specified using the first letter of each method; if no specification is made, the default is BIC:

- **AIC:** the AIC criterion is used
- **BIC:** the BIC criterion is used.

loss.fun: loss function; default set at using residual sum of squares.

**Details**

Typical systematic non-biological variations of a two-color cDNA microarray include the dependence of ratio measurements (M) on intensity (A), print-tip IDs (PT), plate IDs (PL) and spatial heterogeneity of the slide (SP). The stepwise normalization procedure normalizes a slide in a stepwise fashion. In each step one kind of variation is targeted for correction. Within each step, various candidate models are assessed for their adequacy with respect to the observed data. The assessment is made based on a common model selection criterion, AIC (see `calcAIC`) or BIC (see `calcBIC`), and the best model is then chosen for the specified step.

The argument `wf.loc` is a list of steps. Each step is also a list of models. The user uses the function `fitWithin` or `fit2DWithin` to specify a model. Below is a table of how to do so:

<table>
<thead>
<tr>
<th>systematic variation</th>
<th>model</th>
<th>function</th>
</tr>
</thead>
<tbody>
<tr>
<td>intensity (A)</td>
<td>median</td>
<td><code>fitWithin(fun=&quot;medfit&quot;)</code></td>
</tr>
<tr>
<td>A</td>
<td>robust linear</td>
<td><code>fitWithin(fun=&quot;rlmfit&quot;)</code></td>
</tr>
<tr>
<td>A</td>
<td>robust nonlinear</td>
<td><code>fitWithin(fun=&quot;loessfit&quot;)</code></td>
</tr>
<tr>
<td>print-tip (PT)</td>
<td>median</td>
<td><code>fitWithin(z.fun=&quot;maPrintTip&quot;, fun=&quot;medfit&quot;)</code></td>
</tr>
<tr>
<td>PT</td>
<td>robust linear</td>
<td><code>fitWithin(z.fun=&quot;maPrintTip&quot;, fun=&quot;rlmfit&quot;)</code></td>
</tr>
<tr>
<td>PT</td>
<td>robust nonlinear</td>
<td><code>fitWithin(z.fun=&quot;maPrintTip&quot;,fun=&quot;loessfit&quot;)</code></td>
</tr>
<tr>
<td>plate (PL)</td>
<td>median</td>
<td><code>fitWithin(z.fun=&quot;maCompPlate&quot;, fun=&quot;medfit&quot;)</code></td>
</tr>
<tr>
<td>PL</td>
<td>robust linear</td>
<td><code>fitWithin(z.fun=&quot;maCompPlate&quot;, fun=&quot;rlmfit&quot;)</code></td>
</tr>
<tr>
<td>PL</td>
<td>robust nonlinear</td>
<td><code>fitWithin(z.fun=&quot;maCompPlate&quot;,fun=&quot;loessfit&quot;)</code></td>
</tr>
<tr>
<td>spatial (SP)</td>
<td>robust linear</td>
<td><code>fit2DWithin(fun=&quot;rlm2Dfit&quot;)</code></td>
</tr>
<tr>
<td>SP</td>
<td>robust nonlinear(span=0.2)</td>
<td><code>fit2DWithin(fun=&quot;loess2Dfit&quot;, span=0.2)</code></td>
</tr>
<tr>
<td>SP</td>
<td>anova</td>
<td><code>fit2DWithin(fun=&quot;aov2Dfit&quot;)</code></td>
</tr>
<tr>
<td>SP</td>
<td>spatial median (11X11)</td>
<td><code>fit2DWithin(fun=&quot;spatialMedfit&quot;, width=11)</code></td>
</tr>
</tbody>
</table>

If the `wf.loc` is not specified by the user, the default procedure conducts normalization in four steps: A -> PT -> PL -> SP and models are as described in the table above. The user can choose not to follow such a procedure by passing in a different list, however we advocate normalizing the intensity (A) variation first as it is usually the source of most variation in most slides. The list can be easier specified using the function `makeStepList` by inputing models as character strings, see `makeStepList` for details.

**Value**

An object of class "list":

- `normdata`: an object of class `marrayNorm`, containing the normalized intensity data.
- `res`: a dataframe of the stepwise normalization result, containing the name of the model chosen for each step, deviance, equivalent number of parameters, AIC/BIC value.
stepWithinNorm

Author(s)
Yuanyuan Xiao, <yxiao@itsa.ucsf.edu>,
Jean Yee Hwa Yang, <jean@biostat.ucsf.edu>

References

See Also
seqWithinNorm, withinNorm, fitWithin, fit2DWithin, calcAIC, calcBIC.

Examples
# Examples use swirl dataset, for description type ? swirl
data(swirl)

# Apply stepwise normalization for the first slide
res.swirl1 <- stepWithinNorm(swirl[,1])

# normalized data
norm.swirl <- res.swirl1[[1]]

# stepwise procedure
step.swirl <- res.swirl1[[2]]

# using a stepwise procedure different than the default
# corrects intensity (A) and print-tip (PT), this can be
# carried out in two ways:
# 1)
steps <- list(
    wholeChipA = list(med = fitWithin(fun="medfit"),
                     rlm = fitWithin(fun="rlmfit"),
                     loess = fitWithin(fun="loessfit")),
    printTipA = list(med = fitWithin(z.fun="maPrintTip", fun="medfit"),
                     rlm = fitWithin(z.fun="maPrintTip", fun="rlmfit"),
                     loess = fitWithin(z.fun="maPrintTip", fun="loessfit")))

#2)
steps <- makeStepList(PL=NULL, Spatial2D=NULL)
## Not run:
res.swirl <- stepWithinNorm(swirl[,1], wf.loc=steps)
## End(Not run)

# using AIC criterion for the first slide
## Not run:
res.swirl <- stepWithinNorm(swirl[,1], criterion="A")
## End(Not run)
withinNorm

Within-slide normalization function for cDNA spotted microarrays

Description

This function is a wrapper function around fitWithin and fit2DWithin. It allows the user to choose from a set of thirteen basic location normalization procedures. The function operates on an object of class marrayRaw or marrayNorm and returns an object of class marrayNorm.

Usage

withinNorm(marraySet, y = "maM", subset = TRUE, norm = c("none", "median", "rlm", "loess", "medianPrintTip", "rlmPrintTip", "loessPrintTip", "medianPlate", "rlmPlate", "loessPlate", "aov2D", "rlm2D", "loess2D", "spatialMedian"), ...)

Arguments

marraySet Object of class marrayRaw or class marrayNorm, containing intensity data for the batch of arrays to be normalized.
y Name of accessor method for spot statistics, usually the log-ratio maM.
subset A "logical" or "numeric" vector indicating the subset of points used to compute the normalization values.
norm A character string specifying the normalization procedures:
  none: no normalization
  median: global median location normalization
  rlm: global intensity or A-dependent robust linear normalization using the rlm function
  loess: global intensity or A-dependent robust nonlinear normalization using the loess function
  medianPrintTip: within-print-tip-group median normalization
  rlmPrintTip: within-print-tip-group intensity or A-dependent robust linear normalization using the rlm function
  loessPrintTip: within-print-tip-group intensity or A-dependent robust nonlinear normalization using the loess function
  medianPlate: within-well-plate-group median normalization
  rlmPlate: within-well-plate-group intensity or A-dependent robust linear normalization using the rlm function
  loessPlate: within-well-plate-group intensity or A-dependent robust nonlinear normalization using the loess function
  aov2D: spatial bivariate location normalization using ANOVA
  rlm2D: spatial bivariate location normalization using the rlm function
  loess2D: spatial bivariate location normalization using the loess function
  spatialMedian: spatial location normalization using a spatial median approach (see Wilson et al. (2003) in reference)
... Misc arguments for the specified norm function
**withinNorm**

**Details**

The function withinNorm dispatches to the function fitWithin or fit2DWithin with specified arguments according to the choice of norm. For instance, when norm="loess" for global intensity dependent robust nonlinear normalization, withinNorm calls fitWithin(fun="loess") with the default span parameter set at 0.4. If a different span is preferred, it should be input by span=0.2 through the argument ... in the withinNorm function (see example below). For more details see fitWithin, fit2DWithin and individual fitting functions such as loessfit.

**Value**

An object of class marrayNorm, containing the normalized intensity data.

**Author(s)**

Yuanyuan Xiao, <yxiao@itsa.ucsf.edu>,
Jean Yee Hwa Yang, <jean@biostat.ucsf.edu>

**References**


**See Also**

seqWithinNorm, stepWithinNorm, fitWithin, fit2DWithin, loessfit, rlmfit.

**Examples**

```r
# Examples use swirl dataset, for description type ? swirl
data(swirl)

# Apply loess normalization for the first slide, span=0.4
## Not run:
res.swirl1 <- withinNorm(swirl[,1], norm="loess")
## End(Not run)

# Apply loess normalization for the first slide, span=0.2
## Not run:
res.swirl1 <- withinNorm(swirl[,1], norm="loess", span=0.2)
## End(Not run)
```
Index

*Topic classes
  marrayFit-class, 11

*Topic manip
  calcAIC, 2
  calcBIC, 3
  maCompPlate2, 8

*Topic models
  fit2DWithin, 4
  fitWithin, 6
  makeStepList, 10
  seqWithinNorm, 13
  stepWithinNorm, 15
  withinNorm, 15, 17, 18

abs, 2, 3, 13
AIC, 3, 4

calcAIC, 2, 4, 14–17
calcBIC, 3, 3, 14–17
deviance, 3, 4

factor, 9
fit2DWithin, 4, 8, 12, 15–17, 19
fitWithin, 6, 6, 12, 15–17, 19

list, 15
lm, 5
loess, 5, 7, 10–14, 18
loessfit, 19

maCompPlate, 7–9
maCompPlate2, 8
makeStepList, 10, 15, 16
maPrintTip, 7
marrayFit, 2, 3, 5, 7
marrayFit(marrayFit-class), 11
marrayFit-class, 11
marrayLayout, 9
marrayNorm, 4–7, 9, 13–16, 18, 19
marrayRaw, 4–7, 9, 13, 15, 18

rlm, 5, 7, 10, 11, 13, 14, 18
rlmfit, 19