vsn

Class to contain result of a vsn fit

Description

Class to contain result of a vsn fit

Creating Objects

new("vsn") vsn2(x) with x being an ExpressionSet.

Slots

coefficients: A 3D array of size (number of strata) x (number of columns of the data matrix) x 2 containing fitted normalization parameters (see vignette).

strata: A factor of length 0 or n. If its length is n, then its levels correspond to different normalization strata (see vignette).

mu: A numeric vector of length n with the fitted parameters $\hat{\mu}_k, k = 1, \ldots, n.$

sigsq: A numeric scalar, $\hat{\sigma}^2.$

hx: A numeric matrix with 0 or n rows. If the number of rows is n, then hx contains the transformed data matrix.

lbfgsb: An integer scalar containing the return code from the L-BFGS-B optimizer.

Methods

[ Subset

dim Get dimensions of data matrix.

nrow Get number of rows of data matrix.

ncol Get number of columns of data matrix.

show Print a summary of the object

exprs Accessor to slot hx.

coef, coefficients Accessors to slot coefficients.

Author(s)

Wolfgang Huber http://www.ebi.ac.uk/huber
See Also

vsn2

Examples

data("kidney")
v = vsn2(kidney)
show(v)
dim(v)
v[1:10, ]

vsnInput

Class to contain input data and parameters for vsn functions

Description

Class to contain input data and parameters for vsn functions

Creating Objects

new("vsnInput")

Slots

x: A numeric matrix with the input data.
reference: An object of vsn, typically this would have been obtained from a previous fit to a set of reference arrays (data).
strata: A factor of length 0 or n. If its length is n, then its levels correspond to different normalization strata (see vsn2).
ordered: Logical scalar; are the rows reordered so that the strata are contiguous.
lts.quantile: Numeric scalar, seevsn2.
subsample: Integer scalar, seevsn2.
verbose: Logical scalar, seevsn2.
pstart: A 3D array of size (number of strata) x (number of columns of the data matrix) x 2 containing start parameters.
cvg.niter: Integer scalar, seevsn2.
cvg.eps: Numeric scalar, seevsn2.

Methods

[ Subset
dim Get dimensions of data matrix.
nrow Get number of rows of data matrix.
ncol Get number of columns of data matrix.
show Print a summary of the object
**getIntensityMatrix**

**Author(s)**

Wolfgang Huber [http://www.ebi.ac.uk/huber](http://www.ebi.ac.uk/huber)

**See Also**

vsn2

---

**getIntensityMatrix**  
*Extract a matrix of microarray intensities from various kinds of objects*

**Description**

Extract a matrix of microarray intensities from various kinds of objects. This function is called by the function vsn, and is normally not called by the user.

**Usage**

```r
getIntensityMatrix (intensities, verbose)
```

**Arguments**

- `intensities` Object of class `ExpressionSet` with raw intensity values from a microarray experiment. Alternatively, this may be a `matrix`, a `data.frame` with all numeric columns or an `marrayRaw` object.
- `verbose` Logical. If TRUE, some messages are printed.

**Value**

A numeric matrix.

**Author(s)**

Wolfgang Huber [http://www.ebi.ac.uk/huber](http://www.ebi.ac.uk/huber)

**See Also**

vsn

**Examples**

```r
## see the vignette
```
justvsn

Wrapper functions for vsn

Description

justvsn returns the vsn-normalised intensities in an object generally of the same class as its first argument (see the man page of predict for details). It preserves the metadata. It is equivalent to calling

\[
\begin{align*}
  \text{fit} & = \text{vsn2}(x, \ldots) \\
  \text{nx} & = \text{predict}(\text{fit}, \text{newdata}=x, \text{useDataInFit} = \text{TRUE})
\end{align*}
\]

vsnrma is a wrapper around vsn2 and rma

Usage

justvsn(x, ...)  

vsnrma(x, ...)

Arguments

x  

For justvsn, any kind of object for which vsn2 methods exist. For vsnrma, an AffyBatch.

...  

Further arguments that get passed on to vsn2.

Details

vsnrma does probe-wise background correction and between-array normalization by calling vsn2 on the perfect match (PM) values only. Probeset summaries are calculated with the medianpolish algorithm of rma.

Author(s)

Wolfgang Huber http://www.ebi.ac.uk/huber

See Also

vsn2

Examples

```r
##--------------------------------------------------
## use "vsn2" to produce a "vsn" object
##--------------------------------------------------
library(vsn)
data("kidney")
fit = vsn2(kidney)
nkid = predict(fit, newdata=kidney)

##--------------------------------------------------
## justvsn on ExpressionSet
##--------------------------------------------------
nkid2 = justvsn(kidney)
stopifnot(identical(exprs(nkid), exprs(nkid2)))
```
kidney

Intensity data for 1 cDNA slide with two adjacent tissue samples from a nephrectomy (kidney)

Description

Intensity data for 1 cDNA slide with two adjacent tissue samples from a nephrectomy (kidney)

Usage

data(kidney)

Format

kidney is an ExpressionSet containing the data from one cDNA chip. The 8704x2 matrix exprs(kidney) contains the spot intensities for the red (635 nm) and green color channels (532 nm) respectively. For each spot, a background estimate from a surrounding region was subtracted.

Details

The chip was produced in 2001 by Holger Sueltmann at the Division of Molecular Genome Analysis at the German Cancer Research Center in Heidelberg.

References


See Also

vsn

Examples

data(kidney)
plot(exprs(kidney), pch=".", log="xy")
abline(a=0,b=1,col="blue")
lymphoma

Intensity data for 8 cDNA slides with CLL and DLBL samples from the Alizadeh et al. paper in Nature 2000

Description

8 cDNA chips from Alizadeh lymphoma paper

Usage

data(lymphoma)

Format

lymphoma is an ExpressionSet containing the data from 8 chips from the lymphoma data set by Alizadeh et al. (see references). Each chip represents two samples: on color channel 1 (CH1, Cy3, green) the common reference sample, and on color channel 2 (CH2, Cy5, red) the various disease samples. See pData(lymphoma). The 9216x16 matrix exprs(lymphoma) contains the background-subtracted spot intensities (CH1I-CH1B and CH2I-CH2B, respectively).

Details

The chip intensity files were downloaded from the Stanford microarray database. Starting from the link below, this was done by following the links Published Data -> Alizadeh AA, et al. (2000) Nature 403(6769):503-11 -> Data in SMD -> Display Data, and selecting the following 8 slides:

lc7b019
lc7b047
lc7b048
lc7b056
lc7b057
lc7b058
lc7b069
lc7b070

Then, the script makedata.R from the scripts subdirectory of this package was run to generate the R data object.

Source

http://genome-www5.stanford.edu/MicroArray/SMD

References


See Also

vsn
Examples

data(lymphoma)
lymphoma
pData(lymphoma)

meanSdPlot(x, ranks = TRUE, xlab = ifelse(ranks, "rank(mean)", "mean"), ylab = "sd", pch = ".", plot = TRUE, 
...)

Arguments

x An object of class matrix, ExpressionSet, or vsn.
ranks Logical, indicating whether the x-axis (means) should be plotted on the original scale (FALSE) or on the rank scale (TRUE). The latter distributes the data more evenly along the x-axis and allows a better visual assessment of the standard deviation as a function of the mean.
xlab Character, label for the x-axis.
ylab Character, label for the y-axis.
pch Plot symbol.
plot Logical. If TRUE (default), a plot is produced. Calling the function with plot=FALSE can be useful if only its return value is of interest.
...

Details

Standard deviation and mean are calculated row-wise from the expression matrix (in) x. The scatterplot of these versus each other allows to visually verify whether there is a dependence of the standard deviation (or variance) on the mean. The red dots depict the running median estimator (window-width 10%). If there is no variance-mean dependence, then the line formed by the red dots should be approximately horizontal.

Value

A named list with two components, containing x-positions and values of the running median estimator (the red dots in the plot). The methods can also have a side effect, which is to create a plot on the active graphics device.
normalize.AffyBatch.vsn

Author(s)
Wolfgang Huber http://www.ebi.ac.uk/huber

See Also
vsn

Examples

data(kidney)
log.na = function(x) log(ifelse(x>0, x, NA))
exprs(kidney) = log.na(exprs(kidney))
meanSdPlot(kidney)

## ...try this out with non-logged data, the lymphoma data, your data...

normalize.AffyBatch.vsn

Wrapper for vsn to be used as a normalization method with expresso

Description

Wrapper for vsn2 to be used as a normalization method with the expresso function of the package affy. The expresso function is deprecated, consider using justvsn instead. The normalize.AffyBatch.vsn can still be useful on its own, as it provides some additional control of the normalization process (fitting on subsets, alternate transform parameters).

Usage

normalize.AffyBatch.vsn(
  abatch,
  reference,
  strata = NULL,
  subsample = if (nrow(exprs(abatch))>30000L) 30000L else 0L,
  subset,
  log2scale = TRUE,
  log2asymp=FALSE,
  ...
)

Arguments

abatch An object of type AffyBatch.
reference Optional, a 'vsn' object from a previous fit. If this argument is specified, the data in 'x' are normalized "towards" an existing set of reference arrays whose parameters are stored in the object 'reference'. If this argument is not specified, then the data in 'x' are normalized "among themselves". See vsn2 for details.
strata The 'strata' functionality is not supported, the parameter is ignored.
subsample Is passed on to vsn2.
normalize.AffyBatch.vsn

subset

This allows the specification of a subset of expression measurements to be used for the vsn fit. The transformation with the parameters of this fit is then, however, applied to the whole dataset. This is useful for excluding expression measurements that are known to be differentially expressed or control probes that may not match the vsn model, thus avoiding that they influence the normalization process. This operates at the level of probesets, not probes. Both 'subset' and 'subsample' can be used together.

log2scale

If TRUE, this will perform a global affine transform on the data to put them on a similar scale as the original non-transformed data. Many users prefer this. Fold-change estimates are not affected by this transform. In some situations, however, it may be helpful to turn this off, e.g., when comparing independently normalized subsets of the data.

log2asymp

If TRUE, this will perform a global affine transform on the data to make the generalized log (asinh) transform be asymptotically identical to a log base 2 transform. Some people find this helpful. Only one of 'log2scale' or 'log2asymp' can be set to TRUE. Fold-change estimates are not affected by this transform.

Further parameters for vsn2.

Details

Please refer to the Details and References sections of the man page for vsn2 for more details about this method.

Important note: after calling vsn2, the function normalize.AffyBatch.vsn exponentiates the data (base 2). This is done in order to make the behavior of this function similar to the other normalization methods in affy. That packages uses the convention of taking the logarithm to base in subsequent analysis steps (e.g. in medpolish).

Value

An object of class AffyBatch. The vsn object returned, which can be used as reference for subsequent fits, is provided by description(abatch)$preprocessing$vsnReference.

Author(s)

D. P. Kreil http://bioinf.boku.ac.at/, Wolfgang Huber http://www.ebi.ac.uk/huber

See Also

vsn2

Examples

## Please see vignette.
sagmbSimulateData  
Simulate data and assess vsn’s parameter estimation

Description

Functions to validate and assess the performance of vsn through simulation of data.

Usage

sagmbSimulateData(n=8064, d=2, de=0, up=0.5, nrstrata=1, miss=0, log2scale=FALSE)
sagmbAssess(h1, sim)

Arguments

n  Numeric. Number of probes (rows).

d  Numeric. Number of arrays (columns).

de  Numeric. Fraction of differentially expressed genes.

up  Numeric. Fraction of up-regulated genes among the differentially expressed genes.

nrstrata  Numeric. Number of probe strata.

miss  Numeric. Fraction of data points that is randomly sampled and set to NA.

log2scale  Logical. If TRUE, glog on base 2 is used, if FALSE, (the default), then base e.

h1  Matrix. Calibrated and transformed data, according, e.g., to vsn

sim  List. The output of a previous call to sagmbSimulateData, see Value

Details

Please see the vignette.

Value

For sagmbSimulateData, a list with four components: hy, an n x d matrix with the true (=simulated) calibrated, transformed data; y, an n x d matrix with the simulated uncalibrated raw data - this is intended to be fed into vsn: is.de, a logical vector of length n, specifying which probes are simulated to be differentially expressed. strata, a factor of length n.

For sagmbSimulateData, a number: the root mean squared difference between true and estimated transformed data.

Author(s)

Wolfgang Huber http://www.ebi.ac.uk/huber

References

scalingFactorTransformation

See Also

vsn

Examples

sim <- sagmbSimulateData(nrstrata=4)
y <- vsn(sim$y, strata=sim$strata)
res <- sagmbAssess(exprs(y), sim)
res

Description

The transformation that is applied to the scaling parameter of the vsn model

Usage

scalingFactorTransformation(b)

Arguments

b

Real vector.

Value

A real vector of same length as b, with transformation \( f \) applied (see vignette Likelihood Calculations for vsn).

Author(s)

Wolfgang Huber http://www.ebi.ac.uk/huber

Examples

b = seq(-3, 2, length=20)
fb = scalingFactorTransformation(b)
if(interactive())
   plot(b, fb, type="b", pch=16)
The main function of the package is \texttt{vsn2}. Interesting for its applications are also \texttt{predict} and the wrapper function \texttt{justvsn}.

\texttt{vsn2} can be applied to objects of class \texttt{ExpressionSet}, \texttt{NChannelSet}, \texttt{AffyBatch} (from the \texttt{affy} package) and \texttt{RGList} (from the \texttt{limma} package), \texttt{matrix} and \texttt{vector}. It returns an object of class \texttt{vsn}, which contains the results of fitting the \texttt{vsn} model to the data.

The most common use case is that you will want to construct a new data object with the \texttt{vsn}-normalized data whose class is the same as that of the input data and which preserves the metadata. This can be achieved by

\begin{verbatim}
fit = vsn2(x, ...)
x = predict(fit, newdata=x)
\end{verbatim}

To simplify this, there exists also a simple wrapper \texttt{justvsn}.

Author(s)

Wolfgang Huber

\textbf{vsn} \hfill \textit{Variance stabilization and calibration for microarray data.}

\textbf{Description}

Robust estimation of variance-stabilizing and calibrating transformations for microarray data. This function has been superseded by \texttt{vsn2}. The function \texttt{vsn} remains in the package for backward compatibility, but for new projects, please use \texttt{vsn2}.

\textbf{Usage}

\begin{verbatim}
vsn(intensities, lts.quantile = 0.5, verbose = interactive(), niter = 10, cvg.check = NULL, describe.preprocessing = TRUE, subsample, pstart, strata)
\end{verbatim}
Arguments

intensities An object that contains intensity values from a microarray experiment. See `getIntensityMatrix` for details. The intensities are assumed to be the raw scanner data, summarized over the spots by an image analysis program, and possibly "background subtracted". The intensities must not be logarithmically or otherwise transformed, and not thresholded or "floored". NAs are not accepted. See details.

lts.quantile Numeric. The quantile that is used for the resistant least trimmed sum of squares regression. Allowed values are between 0.5 and 1. A value of 1 corresponds to ordinary least sum of squares regression.

verbose Logical. If TRUE, some messages are printed.

niter Integer. The number of iterations to be used in the least trimmed sum of squares regression.

cvg.check List. If non-NULL, this allows finer control of the iterative least trimmed sum of squares regression. See details.

pstart Array. If not missing, user can specify start values for the iterative parameter estimation algorithm. See `vsnh` for details.

describe.preprocessing Logical. If TRUE, calibration and transformation parameters, plus some other information are stored in the `preprocessing` slot of the returned object. See details.

subsample Integer. If specified, the model parameters are estimated from a subsample of the data only, the transformation is then applied to all data. This can be useful for performance reasons.

strata Integer vector. Its length must be the same as `nrow(intensities)`. This parameter allows for the calibration and error model parameters to be stratified within each array, e.g to take into account probe sequence properties, print-tip or plate effects. If `strata` is not specified, one pair of parameters is fitted for every sample (i.e. for every column of `intensities`). If `strata` is specified, a pair of parameters is fitted for every stratum within every sample. The strata are coded for by the different integer values. The integer vector `strata` can be obtained from a factor `fac` through `as.integer(fac)`, from a character vector `str` through `as.integer(factor(fac))`.

Details

Overview: The function calibrates for sample-to-sample variations through shifting and scaling, and transforms the intensities to a scale where the variance is approximately independent of the mean intensity. The variance stabilizing transformation is equivalent to the natural logarithm in the high-intensity range, and to a linear transformation in the low-intensity range. In an intermediate range, the arsinh function interpolates smoothly between the two. For details on the transformation, please see the help page for `vsnh`. The parameters are estimated through a robust variant of maximum likelihood. This assumes that for the majority of genes the expression levels are not much different across the samples, i.e., that only a minority of genes (less than a fraction 1–lts.quantile) is differentially expressed.

Even if most genes on an array are differentially expressed, it may still be possible to use the estimator: if a set of non-differentially expressed genes is known, e.g. because they are external controls or reliable 'house-keeping genes', the transformation parameters can be fitted with `vsn` from the data of these genes, then the transformation can be applied to all data with `vsnh`.
Format: The format of the matrix of intensities is as follows: for the two-color printed array technology, each row corresponds to one spot, and the columns to the different arrays and wavelengths (usually red and green, but could be any number). For example, if there are 10 arrays, the matrix would have 20 columns, columns 1...10 containing the green intensities, and 11...20 the red ones. In fact, the ordering of the columns does not matter to vsn, but it is your responsibility to keep track of it for subsequent analyses. For one-color arrays, each row corresponds to a probe, and each column to an array.

Performance: This function is slow. That is due to the nested iteration loops of the numerical optimization of the likelihood function and the heuristic that identifies the non-outlying data points in the least trimmed squares regression. For large arrays with many tens of thousands of probes, you may want to consider random subsetting: that is, only use a subset of the e.g. 10-20,000 rows of the data matrix intensities to fit the parameters, then apply the transformation to all the data, using vsnh. An example for this can be seen in the function normalize.AffyBatch.vsn, whose code you can inspect by typing normalize.AffyBatch.vsn on the R command line.

Iteration control: By default, if cvg.check is NULL, the function will run the fixed number niter of iterations in the least trimmed sum of squares regression. More fine-grained control can be obtained by passing a list with elements eps and n. If the maximum change between transformed data values is smaller than eps for n subsequent iterations, then the iteration terminates.

Estimated transformation parameters: If describe.preprocessing is TRUE, the transformation parameters are returned in the preprocessing slot of the experimentData slot of the resulting ExpressionSet object, in the form of a list with three elements

- vsnParams: the parameter array (see vsnh for details)
- vsnParamsIter: an array with dimensions c(dim(vsnParams, niter)) that contains the parameter trajectory during the iterative fit process (see also vsnPlotPar).
- vsnTrimSelection: a logical vector that for each row of the intensities matrix reports whether it was below (TRUE) or above (FALSE) the trimming threshold.

If intensities has class ExpressionSet, and its experimentData slot has class MIAME, then this list is appended to any existing entries in the preprocessing slot. Otherwise, the experimentData object and its preprocessing slot are created.

Value

An object of class ExpressionSet. Differences between the columns of the transformed intensities are "generalized log-ratios", which are shrinkage estimators of the natural logarithm of the fold change. For the transformation parameters, please see the Details.

Author(s)

Wolfgang Huber http://www.ebi.ac.uk/huber

References

Variance stabilization applied to microarray data calibration and to the quantification of differential expression, Wolfgang Huber, Anja von Heydebreck, Holger Sueltmann, Annemarie Poustka, Martin Vingron; Bioinformatics (2002) 18 Suppl.1 S96-S104.

vsn2

Fit the vsn model

vsn2 fits the vsn model to the data in x and returns a vsn object with the fit parameters and the transformed data matrix. The data are, typically, feature intensity readings from a microarray, but this function may also be useful for other kinds of intensity data that obey an additive-multiplicative error model. To obtain an object of the same class as x, containing the normalised data and the same metadata as x, use

```r
fit = vsn2(x, ...)
mx = predict(fit, newdata=x)
```

or the wrapper justvsn. Please see the vignette Introduction to vsn for a description on how to use vsn2 for different use cases.

Usage

vsnMatrix(x,
reference,
strata,
lts.quantile = 0.9,

data(kidney)
log.na = function(x) log(ifelse(x>0, x, NA))

if(interactive()) {
x11(width=9, height=4.5)
par(mfrow=c(1,2))
}
plot(log.na(exprs(kidney)), pch=".", main="log-log")

vsnkid = vsn(kidney)  ## transform and calibrate
plot(exprs(vsnkid), pch=".", main="h-h")

if (interactive()) {
x11(width=9, height=4)
par(mfrow=c(1,3))
}
meanSdPlot(vsnkid)
vsnPlotPar(vsnkid, "factors")
vsnPlotPar(vsnkid, "offsets")

## this should always hold true
params = preproc(description(vsnkid))$vsnParams
stopifnot(all(vsnh(exprs(kidney), params) == exprs(vsnkid)))

Examples

data(kidney)
log.na = function(x) log(ifelse(x>0, x, NA))

if(interactive()) {
x11(width=9, height=4.5)
par(mfrow=c(1,2))
}
plot(log.na(exprs(kidney)), pch=".", main="log-log")

vsnkid = vsn(kidney)  ## transform and calibrate
plot(exprs(vsnkid), pch=".", main="h-h")

if (interactive()) {
x11(width=9, height=4)
par(mfrow=c(1,3))
}
meanSdPlot(vsnkid)
vsnPlotPar(vsnkid, "factors")
vsnPlotPar(vsnkid, "offsets")

## this should always hold true
params = preproc(description(vsnkid))$vsnParams
stopifnot(all(vsnh(exprs(kidney), params) == exprs(vsnkid)))

See Also

vsnh, vsnPlotPar, ExpressionSet-class, MIAME-class, normalize.AffyBatch.vsn

Description

vsn2 fits the vsn model to the data in x and returns a vsn object with the fit parameters and the transformed data matrix. The data are, typically, feature intensity readings from a microarray, but this function may also be useful for other kinds of intensity data that obey an additive-multiplicative error model. To obtain an object of the same class as x, containing the normalised data and the same metadata as x, use

```r
fit = vsn2(x, ...)
mx = predict(fit, newdata=x)
```

or the wrapper justvsn. Please see the vignette Introduction to vsn for a description on how to use vsn2 for different use cases.
subsample = 0L,
verbose = interactive(),
returnData = TRUE,
pstart,
minDataPointsPerStratum = 42L,
optimpar = list(),
defaultpar = list(factr=5e7, pgtol=2e-4, maxit=60000L,
trace=0L, cvg.niter=7L, cvg.eps=0))

## S4 method for signature 'ExpressionSet':
## S4 method for signature 'AffyBatch':
## S4 method for signature 'matrix':
## S4 method for signature 'NChannelSet':
## S4 method for signature 'RGList':

Arguments

x
An object containing the data to which the model is to be fitted.

reference
Optional, a vsn object from a previous fit. If this argument is specified, the data in x are normalized “towards” an existing set of reference arrays whose parameters are stored in the object reference. If this argument is not specified, then the data in x are normalized “among themselves”. See Details for a more precise explanation.

strata
Optional, a factor or integer whose length is nrow(x). Can be used for stratified normalization (i.e. separate offsets a and factors b for each level of strata). If missing, all rows of x are assumed to come from one stratum. If strata is an integer, its values must cover the range 1...n, where n is the number of strata.

lts.quantile
Numeric of length 1. The quantile that is used for the resistant least trimmed sum of squares regression. Allowed values are between 0.5 and 1. A value of 1 corresponds to ordinary least sum of squares regression.

subsample
Integer of length 1. If specified, the model parameters are estimated from a subsample of the data of size subsample only, yet the fitted transformation is then applied to all data. For large datasets, this can substantially reduce the CPU time and memory consumption at a negligible loss of precision.

backgroundsubtract
Logical of length 1: should local background estimates be subtracted before fitting vsn?

foreground, background
Aligned character vectors of the same length, naming the channels of x that should be used as foreground and background values.
verbose
Logical. If TRUE, some messages are printed.

returnData
Logical. If TRUE, the transformed data are returned in a slot of the resulting 
$vsn$ object. Setting this option to FALSE allows saving memory if the data are 
not needed.

pstart
Optional, a three-dimensional numeric array that specifies start values for the 
iterative parameter estimation algorithm. If not specified, the function tries 
to guess useful start values. The first dimension corresponds to the levels of 
strata, the second dimension to the columns of $x$ and the third dimension 
must be 2, corresponding to offsets and factors.

minDataPointsPerStratum
The minimum number of data points per stratum.

optimpar
Optional, a list with parameters for the likelihood optimisation algorithm. De-
fault parameters are taken from defaultpar. See details.

defaultpar
The default parameters for the likelihood optimisation algorithm. Values in 
optimpar take precedence over those in defaultpar. The purpose of this 
argument is to expose the default values in this manual page - it is not intended 
to be changed, please use optimpar for that.

... Arguments that get passed on to $vsnMatrix$.

Value
An object of class $vsn$.

Note on overall scale and location of the glog transformation

The data are returned on a $glog$ scale to base 2. More precisely, the transformed data are subject 
to the transformation $glog2(f(b)*x+a) + c$, where the function $glog2(u) = log2(u+sqrt(u*u+1)) = 
asinh(u)/log(2)$ is called the generalised logarithm, the offset $a$ and the scaling parameter $b$ are the 
fitted model parameters (see references), and $f(x)=exp(x)$ is a parameter transformation that allows 
ensuring positivity of the factor in front of $x$ while using an unconstrained optimisation over $b$ [4]. 
Different parameters $a$ and $b$ are fit for each array, and, if applicable, for each stratum. The overall 
offset $c$ is computed from the $b$'s such that for large $x$ the transformation approximately corresponds 
to the $log2$ function. This is done separately for each stratum, but with the same value across arrays. 
More precisely, if the element $b[s,i]$ of the array $b$ is the scaling parameter for the $s$-th stratum 
and the $i$-th array, then $c[s]$ is computed as $log2(2*f(mean(b[,i])))$. The offset $c$ is 
inconsequential for all differential expression calculations, but many users like to see the data in a 
range that they are familiar with.

Specific behaviour of the different methods

$vsn2$ methods exist for ExpressionSet, NChannelSet, AffyBatch (from the affy pack-
age), RGList (from the limma package), matrix and numeric. If $x$ is an NChannelSet, 
then vsn2 is applied to the matrix that is obtained by horizontally concatenating the color chan-
nels. Optionally, available background estimates can be subtracted before. If $x$ is an RGList, 
it is converted into an NChannelSet using a copy of Martin Morgan’s code for RGList to 
NChannelSet coercion, then the NChannelSet method is called.

Standalone versus reference normalisation

If the reference argument is not specified, then the model parameters $\mu_k$ and $\sigma$ are fit from the 
data in $x$. This is the mode of operation described in [1] and that was the only option in versions 
1.X of this package. If reference is specified, the model parameters $\mu_k$ and $\sigma$ are taken from 
it. This allows for ’incremental’ normalization [4].
Convergence of the iterative likelihood optimisation

L-BFGS-B uses three termination criteria:

1. \((f_k - f_{k+1}) / \max(|f_k|, |f_{k+1}|, 1) \leq \text{factr} \times \text{epsmch}\) where 
   \text{epsmch} is the machine precision.

2. \(|\text{gradient}| < \text{pgtol}\)

3. \(\text{iterations} > \text{maxit}\)

These are set by the elements \text{factr}, \text{pgtol} and \text{maxit} of \text{optimpar}. The remaining elements are

\text{trace} An integer between 0 and 6, indicating the verbosity level of L-BFGS-B, higher values create more output.

\text{cvg.niter} The number of iterations to be used in the least trimmed sum of squares regression.

\text{cvg.eps} Numeric. A convergence threshold for the least trimmed sum of squares regression.

Author(s)

Wolfgang Huber http://www.ebi.ac.uk/huber

References


See Also

\text{justvsn, predict}

Examples

data("kidney")

fit = vsn2(kidney) # fit
nkid = predict(fit, newdata=kidney) # apply fit

plot(exprs(nkid), pch=".")
abline(a=0, b=1, col="red")
Apply the vsn transformation to data

Description

Apply the vsn transformation to data.

Usage

```r
## S4 method for signature 'vsn':
predict(object, newdata, strata=object@strata, log2scale=TRUE, useDataInFit=FALSE)
```

Arguments

- `object`: An object of class `vsn` that contains transformation parameters and strata information, typically this is the result of a previous call to `vsn2`.
- `newdata`: Object of class `ExpressionSet`, `NChannelSet`, `AffyBatch` (from the `affy` package), `RGList` (from the `limma` package), `matrix` or `numeric`, with the data to which the fit is to be applied to.
- `strata`: Optional, a factor or integer that aligns with the rows of `newdata`; see the `strata` argument of `vsn2`.
- `log2scale`: If TRUE, the data are returned on the glog scale to base 2, and an overall offset is added (see Value section of the `vsn2` manual page). If FALSE, the data are returned on the glog scale to base e, and no offset is added.
- `useDataInFit`: If TRUE, then no transformation is attempted and the data stored in `object` is transferred appropriately into resulting object, which otherwise preserves the class and metadata of `newdata`. This option exists to increase performance in constructs like
  ```r
  fit = vsn2(x, ...)
  nx = predict(fit, newdata=x)
  ```
  and is used, for example, in the `justvsn` function.

Value

An object typically of the same class as `newdata`. There are two exceptions: if `newdata` is an `RGList`, the return value is an `NChannelSet`, and if `newdata` is numeric, the return value is a `matrix` with 1 column.

Author(s)

Wolfgang Huber [http://www.ebi.ac.uk/huber](http://www.ebi.ac.uk/huber)

Examples

data("kidney")
```r
## nb: for random subsampling, the 'subsample' argument of vsn
## provides an easier way to do this
fit = vsn2(kidney[sample(nrow(kidney), 500), ])
tn = predict(fit, newdata=exprs(kidney))
```
logLik-methods  Calculate the log likelihood and its gradient for the vsn model

Description

logLik calculates the log likelihood and its gradient for the vsn model. plotVsnLogLik makes a false color plot for a 2D section of the likelihood landscape.

Usage

## S4 method for signature 'vsnInput':
logLik(object, p, mu=numeric(0), sigsq=as.numeric(NA))

plotVsnLogLik(object, p, whichp=1:2, expand=1, ngrid=31L, fun=logLik, ...)

Arguments

- **object**  A vsnInput object.
- **p**  For plotVsnLogLik, a vector or a 3D array with the point in parameter space around which to plot the likelihood. For logLik, a matrix whose columns are the set of parameters at which the likelihoods are to be evaluated.
- **mu**  Numeric vector of length 0 or nrow(object). If the length is 0, there is no reference and sigsq must be NA (the default value). See vsn2.
- **sigsq**  Numeric scalar.
- **whichp**  Numeric vector of length 2, with the indices of those two parameters in p along which the section is to be taken.
- **expand**  Numeric vector of length 1 or 2 with expansion factors for the plot range. The range is auto-calculated using a heuristic, but manual adjustment can be useful; see example.
- **ngrid**  Integer scalar, the grid size.
- **fun**  Function to use for log-likelihood calculation. This parameter is exposed only for testing purposes.
- **...**  Arguments that get passed on to fun, use this for mu and sigsq.

Details

logLik is an R interface to the likelihood computations in vsn (which are done in C).

Value

For logLik, a numeric matrix of size nrow(p)+1 by ncol(p). Its columns correspond to the columns of p. Its first row are the likelihood values, its rows 2...nrow(p)+1 contain the gradients. If mu and sigsq are specified, the ordinary negative log likelihood is calculated using these parameters as given. If they are not specified, the profile negative log likelihood is calculated.

For plotVsnLogLik, a dataframe with the 2D grid coordinates and log likelihood values.

Author(s)

Wolfgang Huber http://www.ebi.ac.uk/huber
vsnPlotPar

See Also
vsn2

Examples

data("kidney")

v = new("vsnInput", x=exprs(kidney),
   pstart=array(as.numeric(NA), dim=c(1, ncol(kidney), 2)))

fit = vsn2(kidney)
print(coef(fit))

p = sapply(seq(-1, 1, length=31), function(f) coef(fit)+c(0,0,f,0))

ll = logLik(v, p)

plot(p[3, ], ll[1, ], type="l", xlab=expression(b[1]), ylab=expression(-log(L)))
abline(v=coef(fit)[3], col="red")

plotVsnLogLik(v, coef(fit), whichp=c(1,3), expand=0.2)

vsnPlotPar

Plot trajectories of calibration and transformation parameters for a vsn fit

Description
Plot trajectories of calibration and transformation parameters for a vsn fit

Usage
vsnPlotPar(x, what, xlab="iter", ylab=what, ...)

Arguments
x An object of class ExpressionSet-class which has been created by the function vsn.
what Character, should either be "factors" or "offsets".
xlab Character, label for the x-axis.
ylab Character, label for the y-axis.
... Further arguments that get passed to plot.default.

Details
The plot that is created by this function may help in assessing whether the parameter estimation in vsn was sufficiently converged.

Value
The function is called for its side effect, creating a plot on the active graphics device.
Author(s)
Wolfgang Huber http://www.ebi.ac.uk/huber

See Also
vsn

Examples
## see example for vsn

---

vsnh

A function that transforms a matrix of microarray intensities.

Description
A function that transforms a matrix of microarray intensities. This function works in conjunction with vsn. vsn and vsnh have been superseded by vsn2 and the predict method for vsn objects. The functions vsn and vsnh remain in the package for backward compatibility, but for new projects, please use vsn2 and predict.

Usage
vsnh(y, p, strata)

Arguments
y
A numeric matrix containing intensity values from an array experiment. It may contain NA values.

p
An array with the transformation parameters. If strata is specified, it must be a 3d array, dim(p)[1] must be greater than or equal to the maximum of strata, dim(p)[2] must be ncol(y), and dim(p)[3] must be 2. If strata is missing, then the first dimension may be omitted. NA values are not allowed. See Details.

strata
Integer vector of length nrow(y). See vsn for details.

Details
The transformation is:
vsnh(y, p, s)[k, i] = asinh( p[s[k], i, 1] + p[s[k], i, 2] * y[k, i] ) - log(2*p[s[1], 1, 2])
where k=1:nrow(y) counts over the probes, i=1:ncol(y) counts over the samples, p[s[k], i, 1] is the calibration offset for stratum s[k] in sample i, p[s[k], i, 2] is the calibration factor for stratum s[k] in sample i, and s[k] is the stratum of the k-th probe.

The constant offset - log(2*p[s[1], 1, 2]) is there to make sure that for large y, vsnh(y) for the first stratum on the first chip is approximately the same as log(y). This has no effect on the generalized log-ratios (glog-ratios), which are differences between transformed intensities, but some users are more comfortable with the absolute values that are obtained this way, since they are more comparable to the log scale.
Value

A numeric matrix of the same size as y, with the transformed data.

Author(s)

Wolfgang Huber http://www.ebi.ac.uk/huber

References

Variance stabilization applied to microarray data calibration and to the quantification of differential expression, Wolfgang Huber, Anja von Heydebreck, Holger Sueltmann, Annemarie Poustka, Martin Vingron; Bioinformatics (2002) 18 Suppl.1 S96-S104.


See Also

vsn

Examples

data(kidney)
y = exprs(kidney)
p = array(c(-0.2, -0.1, 0.1, 0.2, 0.0026, 0.0028, 0.0030, 0.0032), dim=c(2,2,2))
strata = sample(1:2, nrow(y), replace=TRUE)
res1 = vsnh(exprs(kidney), p, strata)
res2 = asinh(p[strata,,1] + p[strata,,2] * y) - log(2*p[strata,1,2])
stopifnot(max(abs(res1 - res2)) < 1e-10)
Index

*Topic classes
  vsn, 1
  vsnInput, 2
*Topic datagen
  sagmbSimulateData, 10
*Topic datasets
  kidney, 5
  lymphoma, 6
*Topic hplot
  meanSdPlot, 7
  vsnPlotPar, 21
*Topic manip
  getIntensityMatrix, 3
*Topic methods
  meanSdPlot, 7
*Topic package
  vsn-package, 12
*Topic robust
  vsn, 12

[,vsn-method(vsn), 1
[,vsnInput-method(vsnInput), 2

AffyBatch, 4, 8, 9, 12, 17, 19

class:vsn (vsn), 1
class:vsnInput (vsnInput), 2
coeff, vsn-method(vsn), 1
coefficients, vsn-method(vsn), 1
coerce, RGList, NChannelSet-method (vsn2), 15
data.frame, 3
dim, vsn-method(vsn), 1
dim, vsnInput-method(vsnInput), 2

ExpressionSet, 1, 3, 5–7, 12, 14, 17, 19
ExpressionSet-class, 15, 21
exprs, vsn-method(vsn), 1

getIntensityMatrix, 3, 13

justvsn, 4, 8, 12, 15, 18, 19
kidney, 5

list, 14
logLik, vsnInput-method
  (logLik-methods), 20
logLik-methods, 20
lymphoma, 6

marrayRaw, 3
matrix, 3, 7
meanSdPlot, 7
meanSdPlot, ExpressionSet-method (meanSdPlot), 7
meanSdPlot, matrix-method (meanSdPlot), 7
meanSdPlot, vsn-method (meanSdPlot), 7
meanSdPlot-methods(meanSdPlot), 7
medpolish, 9
MIAME, 14
MIAME-class, 15

NChannelSet, 12, 17, 19
ncol, vsn-method(vsn), 1
ncol, vsnInput-method(vsnInput), 2
normalize.AffyBatch.vsn, 8, 14, 15
nrow, vsn-method(vsn), 1
nrow, vsnInput-method(vsnInput), 2

plotVsnLogLik (logLik-methods), 20
predict, 4, 18, 22
predict, vsn-method(vsn2trsf), 19

RGList, 12, 17, 19
rma, 4

sagmbAssess (sagmbSimulateData), 10
sagmbSimulateData, 10
scalingFactorTransformation, 11
show, vsn-method(vsn), 1
show, vsnInput-method(vsnInput), 2

vsn, 1, 2, 3, 5–8, 10, 11, 12, 12, 15–17, 19,
  21–23
vsn-class (vsn), 1
vsn-package, 12
vsn2, 2–4, 8, 9, 12, 15, 19–22
vsn2, AffyBatch-method (vsn2), 15
vsn2, ExpressionSet-method (vsn2), 15
vsn2, matrix-method (vsn2), 15
vsn2, NChannelSet-method (vsn2), 15
vsn2, numeric-method (vsn2), 15
vsn2, RGList-method (vsn2), 15
vsn2-methods (vsn2), 15
vsn2trsf, 19
vsnh, 13–15, 22
vsnInput, 2, 20
vsnInput-class (vsnInput), 2
vsnMatrix (vsn2), 15
vsnPlotPar, 14, 15, 21
vsnrma (justvsn), 4