Package 'NanoStringNorm'

September 11, 2020

Type Package

Title Normalize NanoString miRNA and mRNA Data

Version 1.2.1.1

Date 2017-12-11

Author Daryl M. Waggott

Maintainer ORPHANED

Depends R (>= 2.14.0), gdata (>= 2.8.2), vsn (>= 3.22.0), XML (>= 3.98-1.5)

Imports methods

Suggests googleVis (>= 0.2.14), lme4, RUnit (>= 0.4.26)

Description A set of tools for normalizing, diagnostics and visualization of NanoString nCounter data.

License GPL-2

LazyLoad yes

LazyData yes

NeedsCompilation no

Repository CRAN

Date/Publication 2020-09-11 11:11:11

R topics documented:

`NanoStringNorm-package` *A suite of functions for normalization, visualization and differential expression of NanoString miRNA and RNA expression levels*

Description

A suite of functions for the normalization, visualization and differential expression of NanoString miRNA and RNA expression level. The primary interface is through the function *'NanoString-Norm'*. The package enables existing preprocessing methods proposed by NanoString in addition to an extensible plug and play framework for further methods incorporation. Emphasis is put on visualization of normalization diagnostics and results in order to prevent inadvertent batch effects or confounding introduced through the specifics of study design.

Details

Package:	NanoStringNorm
Type:	Package
Version:	1.2.0
Date:	2017-08-17
License:	GPL2
LazyLoad:	yes

Author(s)

Author: Daryl M Waggott Maintainer: Daryl Waggott <daryl.waggott@oicr.on.ca>

`NanoString` *An rda file storing NanoString example datasets. Currently, a data.frame consisting of Nanostring Rat RNA expression values. The data were exported as a tab delimited file from the XLS results.*

Description

Contains mRNA data from an experiment of TCDD exposure in Rats. The data structure and normalization workflow is different for miRNA.

Usage

```
data(NanoString)
```

Format

A data.frame of 25 samples and 68 genes.

`NanoString.mRNA` *A data.frame consisting of Nanostring Rat RNA expression values. The data were exported as a tab delimited file from the XLS results.*

Description

Contains mRNA data from an experiment of TCDD exposure in Rats. The data structure and normalization workflow is different for miRNA.

Usage

```
data(NanoString)
```

Format

A data.frame of 25 samples and 68 genes.

`NanoStringNorm` *Methods for preprocessing NanoString nCounter data*

Description

This function can be used to normalize mRNA and miRNA expression data from the NanoString platform.

Usage

```
NanoStringNorm(
x,
anno = NA,
header = NA,
Probe.Correction.Factor = 'adjust',
CodeCount = 'none',
Background = 'none',
SampleContent = 'none',
OtherNorm = 'none',
CodeCount.summary.target = NA,
SampleContent.summary.target = NA,
round.values = FALSE,
is.log = FALSE,
take.log = FALSE,
return.matrix.of.endogenous.probes = FALSE,
```

```
traits = NA,
predict.conc = FALSE,
verbose = TRUE,
genes.to.fit,
genes.to.predict,
guess.cartridge = TRUE,
...
);
```

Arguments

x	The data used for Normalization. This is typically the raw expression data as exported from an Excel spreadsheet. If *anno* is *NA* then the first three columns must be c('Code.Class', 'Name', 'Accession') and the remaining columns refer to the samples being analyzed. The rows should include all control and endogenous genes. For convenience you can use the Excel import functions *read.xls.RCC* to read directly from nCounter output files.
anno	Alternatively, *anno* can be used to specify the first three annotation columns of the expression data. If anno used then it is assumed that *x* does not contain these data. Anno allows flexible inclusion of alternative annotation data. The only requirement is that it includes the *Code.Class* and *Name* columns. Code.Class refers to gene classification i.e. Positive, Negative, Housekeeping or Endogenous gene. For some Code Sets you will need to manually specify Housekeeping genes.
header	The sample meta data at the top of the RCC excel worksheet. The data typically spans rows 4-18 and there should be the same number of columns as samples. This is imported automaically when you use the *read.xls.RCC*. Values in the header will be parsed for normalization diagnostics i.e. is expression associated with a specific cartridge.
Probe.Correction.Factor	An adjustment factor to be applied at the probe level prior to any normalization. Specify 'filter' if you would like to remove any flagged genes. The default to 'adjust' which uses the probe correction factors concatenated to the gene names. See details.
CodeCount	The method used to normalize for technical assay variation. The options are *none* and *sum* and *geo.mean* (geometric mean). The method adjusts each sample based on its relative value to all samples. The geometric mean may be less susceptible to extreme values. The CodeCount normalization is applied first and is considered the most fundamental Normalization step for technical variation.
Background	The method used to estimate the background count level. Background is calculated based on negative controls and the options are *none, mean, mean.2sd, max*. Mean is the least is the least conservative, while max and mean.2sd are the most robust to false positives. The calculated background is subtracted from each sample. Background is calculated after code count adjustment.
SampleContent	The method used to normalize for sample or RNA content i.e. pipetting fluctuations. The options are *none, housekeeping.sum, housekeeping.geo.mean, total.sum, low.cv.geo.mean, top.mean* and *top.geo.mean*. The Housekeeping options require a set of annotated genes. For RNA datasets the genes can be

specified by editing the Code.Class field to equal 'Housekeeping' or 'Control'. miRNA Code Sets generally already have annotated Housekeeping genes. Note that the Housekeeping genes on miRNA Code Sets are messenger RNAs and therefore undergo a different laboratory process, specifically no ligation. If using Housekeeping genes then 'housekeeping.geo.mean' is recommended. Alternatively, you can use the method *total.sum* which uses the sum of the all the genes or *top.geo.mean* which uses the geometric mean of the top 75 expressed. *low.cv.geo.mean* has recently been added to reduce the influence of outliers and normalizing to trait associated genes. This method simply selects genes that expressed with low coefficients of variation. Sample Content adjustment is applied after Code Count and Background correction.

`OtherNorm` Some additional normalization methods. Options include 'none', 'quantile' and 'zscore'. OtherNorm is applied after CodeCount, Background, SampleContent normalizations. This is probably the largest source of inter sample variation in most studies.

`CodeCount.summary.target`
The expected summary value for CodeCount positive control probes. By default this is calculated as the mean of summary values across all samples. This target value is then used to determine the normalization factor of each sample when a CodeCount method is applied. User-assignment of the target value produces sample-independent CodeCount normalization, allowing for new samples to be added without changing the normalization factors of existing samples.

`SampleContent.summary.target`
The expected summary value for SampleContent housekeeping control probes. By default this is calculated as the mean of summary values across all samples. This target value is then used to determine the normalization factor of each sample when a SampleContent method is applied. User-assignment of the target value produces sample-independent SampleContent normalization, allowing for new samples to be added without changing the normalization factors of existing samples. Note that by definition 'low.cv.geo.mean', 'top.mean', and 'top.geo.mean' methods are not compatible with sample-independent normalization.

`round.values` Should the values be rounded to the nearest absolute count or integer. This simplifies interpretation if taking the log by removing values 0-1 which result in negatives.

`is.log` Is the data already in logspace. This is recommended if applying NSN to PCR type microarray data. In this scenario the geometric log is altered and negative values are not flagged.

`take.log` Should a log2 transformation of the data be used. Taking the log may help distributional assumptions and consistency with PCR based methods for calculating fold change. Note, this is not a flag on if the log has already been taken.

`return.matrix.of.endogenous.probes`
If true a matrix of normalized Endogenous code counts is returned. Annotation columns and control probes are removed. This can be useful if you the output is being used directly for downstream analysis. By default it is FALSE and list of objects including descriptive and diagnostics are returned.

`traits` A vector or matrix of phenotypes, design or batch effect variables. For example tumour status, trx regimen, age at FFPE fixation, RNA quality scores and sample plate. Note, that nCounter design covariates such as 'cartridge' and 'date' can be collected from the header of the RCC Excel spreadsheet. At this time each trait should be binary and may only contain 1,2 or NA similar to the numeric coding of factors. T-tests p-values and Fold-Change are presented in terms of the '2' category. The results can be displayed using built in plotting functions.

`predict.conc` Should the predicted concentration values be returned. Defaults to FALSE. The predicted concentrations are based on a fitted model between the observed counts and the expected concentration (fM). Counts are replaced by predicted concentrations from a model using each samples normalized data. In addition a final column is added which predicts the concentration using the mean of all samples.

`verbose` Output comments on run-status

`genes.to.fit` The set of genes used to generate the normalization parameters. You can specify a vector of code classes or gene names as indicated in the annotation. Alternatively, you can use descriptors such as 'all', 'controls', 'endogenous'. For most methods the model will be fit and applied to the endogenous genes. In some cases such as vsn you may want to vary this approach.

`genes.to.predict`
The set of genes that the parameters or model is applied to.

`guess.cartridge`
Should the cartridge be estimated based on the order of the samples.

`...` The ellipses are included to allow flexible use of parameters that are required by OtherNorm methods. For example the use of *strata or calib* parameters documented in the vsn package.

Details

The code is based on the NanoString analysis guidelines (see references). The function allows normalization of both mRNA and miRNA NanoString expression data. The order of the methods is fixed but the use of the method *none* and multiple iterations of the functions allows flexibility.

Note. Poorly assayed samples could negatively influence the normalization of the remaining data. Prior to normalization check that the binding density is not less than .04 and the number of total counts/FOV is not much less than 1500.

In the newer RCC format (Spring 2012) the probe correction factors are concatenated to the gene name via a pipe symbol. By default these are parsed and corrected for. In the older RCC format the 'Name' column of the RCC worksheet sometimes flags certain probes with the message '(+++ See Message Below)'. If this is the case a 'Readme' document including probe level adjustment factors should have been supplied by your Microrray center. This file must be edited into a tabular file and specified in the Probe.Correction.Factor argument. The function will fail with an error if warnings are detected and no probe levels correction factor is supplied. Upon correction any warning messages will be stripped from the raw data file. This background is probe-specific and all background subtraction should be completed prior to normalization. The number of counts to be subtracted for a given probe is determined by multiplying a correction factor by the number of counts observed for the 128fM positive control in the same lane.

Value

By default the function returns a list of objects including pre-processed counts, specified model, sample and gene summaries. The sample summary holds the calculated normalization factors for evaluation of problem samples. The gene summary includes means, coefficient of variation and differential expression statistics for any traits.

Note

Future updates to include more informative diagnostic plots, trait specific normalization and replicate merging.

Author(s)

Daryl M. Waggott

References

See NanoString website for PDFs on analysis guidelines: https://www.nanostring.com/support/product-support/support-documentation

The NanoString assay is described in the paper: Fortina, P. & Surrey, S. Digital mRNA profiling. Nature biotechnology 26, 293-294 (2008).

Examples

```
### 1 Normalize mRNA and output a matrix of normalized counts #################

# load the NanoString.mRNA dataset
data(NanoString);

# alternatively you can import the dataset
# path.to.xls.file <- system.file("extdata", "RCC_files", "RCCCollector1_rat_tcdd.xls",
#  package = "NanoStringNorm");
# NanoString.mRNA <- read.xls.RCC(x = path.to.xls.file, sheet = 1);

# specifiy housekeeping genes in annotation
NanoString.mRNA[NanoString.mRNA$Name %in%
c('Eef1a1','Gapdh','Hprt1','Ppia','Sdha'),'Code.Class'] <- 'Housekeeping';

# normalize
NanoString.mRNA.norm <- NanoStringNorm(
x = NanoString.mRNA,
anno = NA,
CodeCount = 'geo.mean',
Background = 'mean',
SampleContent = 'housekeeping.geo.mean',
round.values = TRUE,
take.log = TRUE,
return.matrix.of.endogenous.probes = TRUE
);
```

```
### 2 include a trait for differential expression and batch effect evaluation  #
###  A list of diagnostic objects is output.                                   #

# Define a traits based on strain
sample.names <- names(NanoString.mRNA)[-c(1:3)];
strain1 <- rep(1, times = (ncol(NanoString.mRNA)-3));
strain1[grepl('HW',sample.names)] <- 2;
strain2 <- rep(1, times = (ncol(NanoString.mRNA)-3));
strain2[grepl('WW',sample.names)] <- 2;
strain3 <- rep(1, times = (ncol(NanoString.mRNA)-3));
strain3[grepl('LE',sample.names)] <- 2;
trait.strain <- data.frame(
row.names = sample.names,
strain1 = strain1,
strain2 = strain2,
strain3 = strain3
);

# Split the input into separate annotation and count input  ###################

NanoString.mRNA.anno <- NanoString.mRNA[,c(1:3)];
NanoString.mRNA.data <- NanoString.mRNA[,-c(1:3)];

# Normalize
NanoString.mRNA.norm <- NanoStringNorm(
x = NanoString.mRNA.data,
anno = NanoString.mRNA.anno,
CodeCount = 'sum',
Background = 'mean.2sd',
SampleContent = 'top.geo.mean',
round.values = TRUE,
take.log = TRUE,
traits = trait.strain,
return.matrix.of.endogenous.probes = FALSE
);

### 3 plot results ############################################################

# Plot all the plots as PDF report.  See help on Plot.NanoStringNorm for examples
pdf('NanoStringNorm_Example_Plots_All.pdf');
Plot.NanoStringNorm(
x = NanoString.mRNA.norm,
label.best.guess = TRUE,
plot.type = 'all'
);
dev.off();
```

---

norm.comp *norm.comp*

Description

Compare normalizaton methods using signal to noise CV and replicates ICC.

Usage

```
norm.comp(
x,
anno,
replicates = NULL,
CodeCount.methods = c('none', 'sum', 'geo.mean'),
Background.methods = c('none','mean', 'mean.2sd','max'),
SampleContent.methods = c('none','housekeeping.sum', 'housekeeping.geo.mean',
'total.sum','top.mean', 'top.geo.mean', 'low.cv.geo.mean'),
OtherNorm.methods = c('none','quantile','zscore', 'rank.normal', 'vsn'),
histogram = FALSE,
verbose = TRUE,
icc.method = "mixed")
```

Arguments

x	The data used for Normalization. This is typically the raw expression data as exported from an Excel spreadsheet. If anno is NA then the first three columns must be *c('Code.Class', 'Name', 'Accession')* and the remaining columns refer to the samples being analyzed. The rows should include all control and endogenous genes.
anno	Alternatively, anno can be used to specify the first three annotation columns of the expression data. If anno used then it assumed that 'x' does not contain these data. Anno allows flexible inclusion of alternative annotation data. The only requirement is that it includes the 'Code.Class' and 'Name' columns. Code.Class refers to gene classification i.e. Positive, Negative, Housekeeping or Endogenous gene.
replicates	A vector of IDs indicating what samples are replicates. Replicate samples need to have the same ID.
CodeCount.methods	vector of methods to compare
Background.methods	vector of methods to compare
SampleContent.methods	vector of methods to compare
OtherNorm.methods	vector of methods to compare
histogram	logical if histogram is to be output
verbose	logical if logging should be output
icc.method	if replicates are included then the intra-class correlation can be calculated via mixed models or anova. Anova is fast but not appropriate in unbalanced situations.

Details

A data.frame is returned with a single row for each combination of methods. For each method the coefficient of variation is reported for positive controls, housekeeping and endogenous genes. If replicates are included then intra-class correlation (ICC) values are reported. One would expect that minimizing variation in controls and replicates is a reasonable estimate of normalization performance. Ideally, this should be performed on a pilot study. Future extensions will include comparing methods via cross validation on trait differences in order to maximize information content.

Author(s)

Daryl M. Waggott

Examples

```
## Not run:

# load the NanoString.mRNA dataset
data(NanoString);


# specifiy housekeeping genes in annotation
NanoString.mRNA[NanoString.mRNA$Name %in% c('Eef1a1','Gapdh','Hprt1','Ppia','Sdha'),
'Code.Class'] <- 'Housekeeping';

NanoString.mRNA <- NanoString.mRNA[1:50,];


# strain x experimental condition i.e. replicate.
# this is only a small subset of the original data used for the plot
biological.replicates <- c("HW_1.5_0","HW_1.5_0","HW_1.5_0","HW_1.5_100","HW_1.5_100",
"HW_1.5_100","HW_6_100","HW_6_100","HW_3_100","HW_3_100","HW_3_100","HW_3_100","LE_19_0",
"LE_19_0","LE_19_0","LE_96_0","LE_96_0","LE_96_0","HW_10_100","HW_10_100","HW_10_100",
"HW_10_100","HW_6_100","HW_6_100","HW_96_0");

if (requireNamespace("lme4")) {

norm.comp.results <- norm.comp(
x = NanoString.mRNA,
replicates = biological.replicates,
CodeCount.methods = 'none',
Background.methods = 'none',
SampleContent.methods = c('none','housekeeping.sum', 'housekeeping.geo.mean',
'top.mean', 'top.geo.mean'),
OtherNorm.methods = 'none',
verbose = FALSE
);

}


## End(Not run)
```

nsn *Methods for preprocessing NanoString nCounter data.*

Description

This function can be used to normalize mRNA and miRNA expression data from the NanoString platform. This is just a convenience wrapper of the NanoStrinNorm function

Usage

```
nsn(
x,
...
);
```

Arguments

x	The data used for Normalization. This is typically the raw expression data as exported from an Excel spreadsheet. If *anno* is *NA* then the first three columns must be c('Code.Class', 'Name', 'Accession') and the remaining columns refer to the samples being analyzed. The rows should include all control and endogenous genes. For convenience you can use the Excel import functions *read.xls.RCC* to read directly from nCounter output files.
...	Other NanoStringNorm arguments

Author(s)

Daryl M. Waggott

other.normalization *other.normalization*

Description

This function can be used to normalize mRNA and miRNA expression data from the NanoString platform.

Usage

```
other.normalization(
 x,
 anno,
 OtherNorm = 'none',
 verbose = TRUE,
 genes.to.fit,
 genes.to.predict,
 ...);
```

Arguments

x	The data used for Normalization. This is typically the raw expression data as exported from an Excel spreadsheet. If anno is NA then the first three columns must be *c('Code.Class', 'Name', 'Accession')* and the remaining columns refer to the samples being analyzed. The rows should include all control and endogenous genes.
anno	Alternatively, anno can be used to specify the first three annotation columns of the expression data. If anno used then it assumed that 'x' does not contain these data. Anno allows flexible inclusion of alternative annotation data. The only requirement is that it includes the 'Code.Class' and 'Name' columns. Code.Class refers to gene classification i.e. Positive, Negative, Housekeeping or Endogenous gene.
OtherNorm	Some additional normalization methods. Options **currently** include *'none', 'zscore', 'rank.normal'* and *'vsn'*. OtherNorm is applied after CodeCount, Background, SampleContent normalizations, therefore you may want to set them to 'none'.
verbose	Enable run-time status messages
genes.to.fit	The set of genes used to generate the normalization parameters. You can specify a vector of code classes or gene names as indicated in the annotation. Alternatively, you can use descriptors such as 'all', 'controls', 'endogenous'. For most methods the model will be fit and applied to the endogenous genes. vsn defaults to 'all' genes.
genes.to.predict	The set of genes that the parameters or model is applied to.
...	The ellipses are included to allow flexible use of parameters that are required by sub normalization methods. For example the use of *'strata'* or *'calib'* parameters documented in the vsn package.

Details

The methods used for OtherNorm are designed to be extensible to alternate and evolving NanoString pre-processing analysis. These can be combined with standard CodeCount, Background, SampleContent methods (i.e. positive, negative and housekeeping controls).

zscore is simply scaling each sample to have a mean 0 and standard deviation of 1. Similarly, rank.normal uses the quantiles or ranks to transform to the normal distribution with mean 0 and standard deviation of 1. Both methods are helpful when comparing multiple datasets or platforms during meta or joint analysis due to the abstraction of effects sizes or expression scale.

quantile normalization is based on pegging each sample to an empirically derived distribution. The distribution is simply the mean expression across all samples for each gene rank.

vsn normalization is well documented and additional parameterization can be read about via it's package documentation. One interesting feature is that the affine transformation can be turned off leaving just the generalized log2 (glog2) transformation *calib = 'none'*. One can thereby use the existing NanoString methods to shift and scale the count data and then apply glog2 transformation to help stabilize the variance.

Note

Future updates to include more informative diagnostic plots, trait specific normalization and replicate merging.

Author(s)

Daryl M. Waggott

References

See NanoString website for PDFs on analysis guidelines: https://www.nanostring.com/support/product-support/support-documentation

The NanoString assay is described in the paper: Fortina, P. & Surrey, S. Digital mRNA profiling. Nature biotechnology 26, 293-294 (2008).

Examples

```
# load the NanoString.mRNA dataset
data(NanoString);

# specify housekeeping genes in annotation
NanoString.mRNA[NanoString.mRNA$Name %in%
c('Eef1a1','Gapdh','Hprt1','Ppia','Sdha'),'Code.Class'] <- 'Housekeeping';

# z-value transformation. scale each sample to have a mean 0 and sd
# by default all the other normalization methods are 'none'
# you cannot apply a log because there are negative values
# good for meta-analysis and cross platform comparison abstraction of effect size
NanoString.mRNA.norm <- NanoStringNorm(
x = NanoString.mRNA,
OtherNorm = 'zscore',
return.matrix.of.endogenous.probes = TRUE
);


# inverse normal transformation. use quantiles to transform each sample to
# the normal distribution
NanoString.mRNA.norm <- NanoStringNorm(
x = NanoString.mRNA,
OtherNorm = 'rank.normal',
return.matrix.of.endogenous.probes = TRUE
);

# quantile normalization.  create an empirical distribution based on the
# median gene counts at the same rank across sample.  then transform each
# sample to the empirical distribution.
NanoString.mRNA.norm <- NanoStringNorm(
x = NanoString.mRNA,
OtherNorm = 'quantile',
return.matrix.of.endogenous.probes = FALSE
```

```
);

if(require(vsn)) { # only run if vsn installed
# vsn.  apply a variance stabilizing normalization.
# fit and predict the model using 'all' genes
# i.e. 'controls' and 'endogenous', this is the default
# note this is just a wrapper for the vsn package.
# you could even add strata for the controls vs the
#  endogenous to review systematic differences

NanoString.mRNA.norm <- NanoStringNorm(
x = NanoString.mRNA,
OtherNorm = 'vsn',
return.matrix.of.endogenous.probes = FALSE,
genes.to.fit = 'all',
genes.to.predict = 'all'
);

# vsn. this time generate the parameters (fit the model) on the 'controls' and apply
# (predict) on the endogenous
# alternatively you may want to use the endogenous probes for both fitting and predicting.
NanoString.mRNA.norm <- NanoStringNorm(
x = NanoString.mRNA,
OtherNorm = 'vsn',
return.matrix.of.endogenous.probes = FALSE,
genes.to.fit = 'controls',
genes.to.predict = 'endogenous'
);

# vsn. apply standard NanoString normalization strategies as an alternative to the vsn
# affine transformation.
# this effectively applies the glog2 variance stabilizing transformation to the
# adjusted counts
NanoString.mRNA.norm <- NanoStringNorm(
x = NanoString.mRNA,
CodeCount = 'sum',
Background = 'mean',
SampleContent = 'top.geo.mean',
OtherNorm = 'vsn',
return.matrix.of.endogenous.probes = FALSE,
genes.to.fit = 'endogenous',
genes.to.predict = 'endogenous',
calib = 'none'
);
} # end vsn
```

Plot.NanoStringNorm *Plot.NanoStringNorm*

Description

Do some diagnostic and descriptive plots

Usage

```
Plot.NanoStringNorm(
x,
plot.type = 'RNA.estimates',
samples = NA,
genes = NA,
label.best.guess = TRUE,
label.ids = list(),
label.as.legend = TRUE,
label.n = 10,
title = TRUE,
col = NA
);
```

Arguments

x	An object of class NanoStringNorm i.e. the list not matrix output of NanoStringNorm
plot.type	A vector of the different plots that you are interested in. Currently the following are implemented: normalization factors 'norm.factors', coefficient of variance 'cv' and mean vs sd 'mean.sd', 'volcano', 'missing', 'RNA.estimates', 'batch.effects' and 'positive.controls'. To plot all available plots you can use 'all'.
samples	List of samples to use. Not yet implemented.
genes	List of genes to use. Not yet implemented.
label.best.guess	Labels will automatically be added for samples and genes that are either outliers (~3sd from mean) or the most significant. The default is TRUE.
label.ids	Explicitly label points. To use this feature supply a list with with two elements named genes and samples. By default this is set to NA. Note that you can add manual and auto labels at the same time.
title	Should the plot titles be included. By default they are included but this can be turned off if generating plots for publication or presentation.
label.as.legend	Should the points be labelled using a legend or directly. The legend tends to be more clear when there are number of overlapping positions. Only volcano plots implemented currently.
label.n	How many points to be chosen for labelling. Only volcano plots implemented currently.
col	Alternate colours for plotting. Input should be a vector of two rgb values. The default colours are orange and green.

Value

The output is single or series of plots. Plots focus on either genes or samples. Due to variation in sample quality it is a good idea to review data in order to determine outliers. Outliers could effect normalization of other samples and bias results of downstream statistical analysis. Systematic batch effects in terms of cartridge, sample type or covariates should also be evaluated to minimize confounding results.

The mean vs standard deviation plot for each gene is a good indication of the noisiness of controls genes. Also, candidate housekeeping genes can be identified by having a high mean and very low variance.

The coefficient of variation density plot shows the change in signal to noise pre and post normalization.

The volcano plots show the relationship between p-value and fold-change for traits supplied in the normalization step. These can be phenotype's of interest or study design based for evaluating batch effects. Orange dots have a fold-change greater than 2 and are sized proportional to the mean expression.

The proportion missing plot highlights samples that have elevated levels of missing. Elevated missing could be indicative of a low RNA, inhibition or tissue type.

The rna content plot compares estimates of the total amount of RNA. Any inconsistency between housekeeping and global RNA content estimates should be evaluated for assay issues. In the case of miRNA's this could be due to the housekeeping genes undergoing different processing (no ligation).

The batch effect plot looks for differences in sample summary statistics among groups, as defined by traits supplied during normalization. Green dots are sized proportionally to their p-value and are less that 0.05. A cartridge variable is automatically added to the checked traits. If no header is supplied then the cartridge is assumed to be determined by column position in the input dataset. That is, sample 1-12 from cartridge 1, 2-24 from cartridge 2 etc. Note that the mean, sd and missing are post normalization and therefore may show an opposite trend than expected due to overcompensating normalization factors. To check the raw data normalize using 'none' for each option. Ideally, technical or study design traits should have orange dots and be near the line. It can be expected that biological traits such as tumour status have different mean, sd and missing but they should not be correlated with assay related controls or normalization factors (positive, negative, housekeeping controls). It's a good idea to double check any significant associations.

The normalization factors plot looks at the calculated normalization parameters for each sample. By default samples are labeled if one of the normalization factors is greater or less than 100% from the mean. The raw data should be investigated for these samples as they could be contributing unnecessary noise to the downstream analysis. Remember that the normalization factors are ratios of all samples vs a specifcic sample therefore high normalization factors reflect low code counts. If a sample had a high positive control normalization factor this would show up as low intercept on the positive control plots. The trend will also be opposite for the batch effects plots which use the summary of the counts not the ratio based normalization factor. Normalization factors outside 100% difference from the mean could reflect noise inducing over or under correction.

The positive control plots compare the expected concentration with observed counts. Orange dots are the negative controls. The intercept reflects the sensitivity of the assay for that sample.

Author(s)

Daryl M. Waggott

Examples

```
## Not run:

# load data
data(NanoString);

# specifiy housekeeping genes in annotation
NanoString.mRNA[NanoString.mRNA$Name %in%
c('Eef1a1','Gapdh','Hprt1','Ppia','Sdha'),'Code.Class'] <- 'Housekeeping';

# setup the traits
sample.names <- names(NanoString.mRNA)[-c(1:3)];
strain1 <- rep(1, times = (ncol(NanoString.mRNA)-3));
strain1[grepl('HW',sample.names)] <- 2;
strain2 <- rep(1, times = (ncol(NanoString.mRNA)-3));
strain2[grepl('WW',sample.names)] <- 2;
strain3 <- rep(1, times = (ncol(NanoString.mRNA)-3));
strain3[grepl('LE',sample.names)] <- 2;
trait.strain <- data.frame(
row.names = sample.names,
strain1 = strain1,
strain2 = strain2,
strain3 = strain3
);

# normalize
NanoString.mRNA.norm <- NanoStringNorm(
x = NanoString.mRNA,
anno = NA,
CodeCount = 'geo.mean',
Background = 'mean.2sd',
SampleContent = 'housekeeping.geo.mean',
round.values = TRUE,
take.log = TRUE,
traits = trait.strain,
return.matrix.of.endogenous.probes = FALSE
);

# plot all the plots as PDF report
pdf('NanoStringNorm_Example_Plots_All.pdf')
Plot.NanoStringNorm(
x = NanoString.mRNA.norm,
label.best.guess = TRUE,
plot.type = 'all'
);
```

```
dev.off()

# publication quality tiff volcano plot
tiff('NanoStringNorm_Example_Plots_Volcano.tiff', units = 'in',  height = 6,
width = 6, compression = 'lzw', res = 1200, pointsize = 10);
Plot.NanoStringNorm(
x = NanoString.mRNA.norm,
label.best.guess = TRUE,
plot.type = c('volcano'),
title = FALSE
);
dev.off()

# all plots as seperate files output for a presentation
png('NanoStringNorm_Example_Plots_%03d.png', units = 'in',  height = 6,
width = 6, res = 250, pointsize = 10);
Plot.NanoStringNorm(
x = NanoString.mRNA.norm,
label.best.guess = TRUE,
plot.type = c('cv','mean.sd','RNA.estimates','volcano','missing','norm.factors',
'positive.controls','batch.effects')
);
dev.off()

# user specified labelling with optimal resolution for most digital displays
png('NanoStringNorm_Example_Plots_Normalization_Factors.png', units = 'in', height = 6,
width = 6, res = 250, pointsize = 10);
Plot.NanoStringNorm(
x = NanoString.mRNA.norm,
label.best.guess = FALSE,
label.ids = list(genes = rownames(NanoString.mRNA.norm$gene.summary.stats.norm),
samples = rownames(NanoString.mRNA.norm$sample.summary.stats)),
plot.type = c('norm.factors')
);
dev.off()


## End(Not run)
```

`Plot.NanoStringNorm.gvis`

Plot.NanoStringNorm.gvis

Description

Plot interactive normalization diagnostic's and differential expression results using Google charts

Usage

```
Plot.NanoStringNorm.gvis(
```

```
x,
plot.type = c('gene.norm', 'sample'),
save.plot = FALSE,
path.to.mongoose = 'web',
output.directory = 'NanoStringNorm_gvis_plots'
);
```

Arguments

x	An object of class NanoStringNorm i.e. the list output of NanoStringNorm
plot.type	A vector of the different plots that you would like. Currently the following are implemented: gene.norm, gene.raw and sample. These correspond to the list items in the NanoStringNorm output.
save.plot	By default the plot is displayed on your browser using a built in web server. This is good for quick evaluation if you have access to the underlying data. However for distribution and presentations it is easier to save the plots bundled with an embedded web server. You need to set save.plot to TRUE in order to get the bundled output.
path.to.mongoose	The path to a previously downloaded version of the mongoose embedded web server. http://code.google.com/p/mongoose/ If the path to mongoose is not specified then it will be attempted to be downloaded from the internet i.e. default path.to.mongoose = 'web'. If you do not want to attempt to download from the internet then specify 'none'.
output.directory	The name of the directory you want the output saved in.

Value

The output is a series of html files that contain interactive 4 dimensional plots. The motivation for these plots is to enable end users to explore their own data quickly and effectively without depending on informaticians.

For samples, a Google chart is rendered using sample summary statistics, traits and normalization factors as output from the NanoStringNorm function. You can change the variables associated with each axis, colour coding and point sizing. There are a number of ways to query the data in order to emphasise specific samples. You can however mouse over the point to get label information or select specific rows from the table.

Similarly, for gene level data you can explore the results summary statistics and trait associations. For example, change the x-axis to Fold-Change.trait.X and the y-axis to P-value.trait.x, the size of the dot to mean expression and the colour to Code.Class. This gives a nice interactive volcano plot. Alternatively, you can plot the results of P-value.trait.x vs P-value-trait.y in order to look for common genes effecting different traits. Note that P-values are displayed as -log10 P-value.

To display plots saved as a bundle, first navigate to the directory and execute the mongoose binary.

Then start your browser and specify the following url 'http://127.0.0.1:8080' There should be a set of links to the plots. The windows executable binary is automatically downloaded if no location is provided. For other systems the source code is downloaded. To install untar the binary, run make and then copy the resulting binary to the directory containing the output html files.

```
cd output.directory;
tar -zxvf mongoose.tgz;
cd mongoose;
make linux;
cp mongoose ..;
./mongoose;
```

For further documentation regarding these plots see the R package googleVis and Google's visualization API.

http://code.google.com/p/google-motion-charts-with-r
http://code.google.com/apis/visualization/interactive_charts.html

Author(s)

Daryl M. Waggott

Examples

```
# load data
data(NanoString);

# specify housekeeping genes in annotation
NanoString.mRNA[NanoString.mRNA$Name %in%
        c('Eef1a1','Gapdh','Hprt1','Ppia','Sdha'),'Code.Class'] <- 'Housekeeping';

# setup the traits
sample.names <- names(NanoString.mRNA)[-c(1:3)];
strain1 <- rep(1, times = (ncol(NanoString.mRNA)-3));
strain1[grepl('HW',sample.names)] <- 2;
strain2 <- rep(1, times = (ncol(NanoString.mRNA)-3));
strain2[grepl('WW',sample.names)] <- 2;
strain3 <- rep(1, times = (ncol(NanoString.mRNA)-3));
strain3[grepl('LE',sample.names)] <- 2;
trait.strain <- data.frame(
        row.names = sample.names,
        strain1 = strain1,
        strain2 = strain2,
        strain3 = strain3
        );

# normalize
NanoString.mRNA.norm <- NanoStringNorm(
        x = NanoString.mRNA,
        anno = NA,
```

```
         CodeCount = 'geo.mean',
         Background = 'mean.2sd',
         SampleContent = 'housekeeping.geo.mean',
         round.values = TRUE,
         take.log = TRUE,
         traits = trait.strain,
         return.matrix.of.endogenous.probes = FALSE
         );

# plot the sample summaries to your browser
if (requireNamespace("googleVis")) {
Plot.NanoStringNorm.gvis(
         x = NanoString.mRNA.norm,
         plot.type = c('gene.norm', 'sample'),
         save.plot = FALSE
         );

# plot the gene summaries to a directory for distribution and later viewing
Plot.NanoStringNorm.gvis(
       x = NanoString.mRNA.norm,
         plot.type = c('gene.norm', 'sample'),
         save.plot = TRUE,
         path.to.mongoose = 'none',
         output.directory = "NanoStringNorm_Interactive_Plot"
         );
}
```

`read.markup.RCC` *read.markup.RCC*

Description

A function to read and merge a set of sample specific RCC markup files.

Usage

```
read.markup.RCC(
 rcc.path = ".",
 rcc.pattern = "*.RCC|*.rcc",
 exclude = NULL,
 include = NULL,
 nprobes = -1)
```

Arguments

`rcc.path`	The directory of the rcc files. Defaults to current working directory.
`rcc.pattern`	The file pattern used to search for the RCC files. Defaults to the RCC extension
`exclude`	A list of files to ignore

include	A list of files to include
nprobes	(Optional) This defaults to the read.table default for nrows (-1). In some cases, a message follows the comment ("<") line and causes an error. In this case, the user can now define the number of rows (number of probes including controls and experimental) that should be read in.

Value

Returns a list with two components. The first is the header information which contains sample IDs and diagnostic information on the quality of the samples. The second is the count data and can be directly used in the input to NanoStringNorm.

Author(s)

Daryl M. Waggott

Examples

```
# read in all the rcc files in the current directory
# data.raw <- read.markup.RCC();
# data.norm <- NanoStringNorm(data.raw);
```

---

read.xls.RCC *read.xls.RCC*

Description

A function to read the raw counts from the RCC excel spreadsheet output by the nCounter platform.

Usage

```
read.xls.RCC(xls, sheet = 1, perl, sample.id.row = 'File.Name')
```

Arguments

xls	The excel spreadsheet output by nCounter. This should be a string with the path pointing to the desired file.
sheet	The worksheet that contains the raw counts. Make sure to check that you are using the worksheet with the "raw" counts and not something that has been processed. The name of the correct worksheet usually has "RCC" in it but not "norm". The input is an integer and it defaults to 1 or the first worksheet.
perl	The path to a perl binary. This does not need to be specified if perl is in your PATH
sample.id.row	The row in the RCC file that contains the sample IDs to be used as column names for the count data. Defaults to "File.Name" but in some cases "Sample.ID" or "Lane.ID" would be more appropriate.

Value

Returns a list with two components. The first is the header information which contains sample IDs and diagnostic information on the quality of the samples. The second is the count data and can be directly used in the input to NanoStringNorm.

Author(s)

Daryl M. Waggott

Examples

```
# directly import the nCounter output
path.to.xls.file <- system.file("extdata", "RCC_files", "RCCCollector1_rat_tcdd.xls",
package = "NanoStringNorm");
NanoString.mRNA <- read.xls.RCC(x = path.to.xls.file, sheet = 1);
```

Index