Package 'NBPSeq'

May 8, 2026

Type Package

Version 0.3.1

Date 2022-06-09

Title Negative Binomial Models for RNA-Sequencing Data

Author Yanming Di <diy@stat.oregonstate.edu>, Daniel W Schafer <schafer@stat.oregonstate.edu>, with contributions from Jason S Cumbie <cumbiej@onid.orst.edu> and Jeff H Chang <changj@cgrb.oregonstate.edu>.

Maintainer Yanming Di <diy@stat.oregonstate.edu>

Description Negative Binomial (NB) models for two-group comparisons and regression inferences from RNA-Sequencing Data.

Depends R (>= 3.00)

Imports splines, qvalue

License GPL-2

LazyLoad yes

NeedsCompilation yes

Repository CRAN

Date/Publication 2022-06-09 11:02:06 UTC

Contents

`NBPSeq-package` *Negative Binomial Regression Models for Statistical Analysis of RNA-Sequencing Data*

Description

Negative binomial (NB) two-group and regression models for RNA-Sequencing data analysis.

Details

See the examples of `test.coefficient` and `exact.nb.test` for typical workflows of using this package.

`arab` *Arabidopsis RNA-Seq Data Set*

Description

An RNA-Seq dataset from a pilot study of the defense response of Arabidopsis to infection by bacteria. We performed RNA-Seq experiments on three independent biological samples from each of the two treatment groups. The matrix contains the frequencies of RNA-Seq reads mapped to genes in a reference database. Rows correspond to genes and columns correspond to independent biological samples.

Usage

```
data(arab)
```

Format

A 26222 by 6 matrix of RNA-Seq read frequencies.

Details

We challenged leaves of Arabidopsis with the defense-eliciting Δ*hrcC* mutant of *Pseudomonas syringae* pathovar *tomato* DC3000. We also infiltrated leaves of Arabidopsis with 10mM MgCl2 as a mock inoculation. RNA was isolated 7 hours after inoculation, enriched for mRNA and prepared for RNA-Seq. We sequenced one replicate per channel on the Illumina Genome Analyzer (http://www.illumina.com). The length of the RNA-Seq reads can vary in length depending on user preference and the sequencing instrument. The dataset used here are derived from a 36-cycle sequencing reaction, that we trimmed to 25mers. We used an in-house computational pipeline to process, align, and assign RNA-Seq reads to genes according to a reference database we developed for Arabidopsis.

Author(s)

Jason S Cumbie <cumbiej@onid.orst.edu> and Jeff H Chang <changj@cgrb.oregonstate.edu>.

References

Di Y, Schafer DW, Cumbie JS, and Chang JH (2011): "The NBP Negative Binomial Model for Assessing Differential Gene Expression from RNA-Seq", Statistical Applications in Genetics and Molecular Biology, 10 (1).

`estimate.disp` *Fit a parametric disperison model to thinned counts*

Description

Fit a parametric dispersion model to RNA-Seq counts data prepared by `prepare.nbp`. The model parameters are estimated from the pseudo counts: thinned/down-sampled counts that have the same effective library size.

Usage

```
estimate.disp(obj, model = "NBQ", print.level = 1, ...)
```

Arguments

`obj`	output from `prepare.nbp`.
`model`	a string, one of "NBQ" (default), "NBP" or "NB2".
`print.level`	a number, controls the amount of messages printed: 0 for suppressing all messages, 1 for basic progress messages, larger values for more detailed messages.
`...`	additional parameters controlling the estimation of the parameters.

Details

For each individual gene i, a negative binomial (NB) distribution uses a dispersion parameter ϕ_i to capture the extra-Poisson variation between biological replicates: the NB model imposes a mean-variance relationship $\sigma_i^2 = \mu_i + \phi_i \mu_i^2$. In many RNA-Seq data sets, the dispersion parameter ϕ_i tends to vary with the mean μ_i. We proposed to capture the dispersion-mean dependence using parametric models.

With this function, `estimate.disp`, users can choose from three parametric models: NB2, NBP and NBQ (default).

Under the NB2 model, the dispersion parameter is a constant and does not vary with the mean expression levels.

Under the NBP model, the log dispersion is modeled as a linear function of preliminarily estimated log mean relative frequencies (`pi.pre`):

log(phi) = par[1] + par[2] * log(pi.pre/pi.offset),

Under the NBQ model, the log dispersion is modeled as a quadratic function of preliminarily estimated log mean relative frequencies (`pi.pre`):

log(phi) = par[1] + par[2] * log(pi.pre/pi.offset) + par[3] * (log(pi.pre/pi.offset))^2;

The NBQ model is more flexible than the NBP and NB2 models, and is the current default option.

In the NBP and NBQ models, `pi.offset` is fixed to be 1e-4, so par[1] corresponds to the dispersion level when the relative mean frequency is 100 reads per million (RPM).

The dispersion parameters are estimated from the pseudo counts (counts adjusted to have the same effective library sizes). The parameters are estimated by maximizing the log conditional likelihood of the model parameters given the row sums. The log conditional likelihood is computed for each gene in each treatment group and then summed over genes and treatment groups.

Value

The list `obj` from the input with some added components summarizing the fitted dispersion model. Users can print and plot the output to see brief summaries of the fitted dispersion model. The output is otherwise not intended for use by end users directly.

Note

Users should call `prepare.nbp` before calling this function. The function `prepare.nbp` will normalize the counts and adjust the counts so that the effective library sizes are approximately the same (computing the conditional likelihood requires the library sizes to be the same).

References

Di Y, Schafer DW, Cumbie JS, and Chang JH (2011): "The NBP Negative Binomial Model for Assessing Differential Gene Expression from RNA-Seq", Statistical Applications in Genetics and Molecular Biology, 10 (1).

See Also

`nbp.test`, `exact.nb.test`

Examples

```
## See the example for nb.exact.test
```

`estimate.dispersion` *Estimate Negative Binomial Dispersion*

Description

Estimate NB dispersion by modeling it as a parametric function of preliminarily estimated log mean relative frequencies.

Usage

```
estimate.dispersion(nb.data, x, model = "NBQ", method = "MAPL", ...)
```

Arguments

`nb.data`	output from `prepare.nb.data`.
`x`	a design matrix specifying the mean structure of each row.
`model`	the name of the dispersion model, one of "NB2", "NBP", "NBQ" (default), "NBS" or "step".
`method`	a character string specifying the method for estimating the dispersion model, one of "ML" or "MAPL" (default).
`...`	(for future use).

Details

We use a negative binomial (NB) distribution to model the read frequency of gene i in sample j. A negative binomial (NB) distribution uses a dispersion parameter ϕ_{ij} to model the extra-Poisson variation between biological replicates. Under the NB model, the mean-variance relationship of a single read count satisfies $\sigma^2_{ij} = \mu_{ij} + \phi_{ij}\mu^2_{ij}$. Due to the typically small sample sizes of RNA-Seq experiments, estimating the NB dispersion ϕ_{ij} for each gene i separately is not reliable. One can pool information across genes and biological samples by modeling ϕ_{ij} as a function of the mean frequencies and library sizes.

Under the NB2 model, the dispersion is a constant across all genes and samples.

Under the NBP model, the log dispersion is modeled as a linear function of the preliminary estimates of the log mean relative frequencies (`pi.pre`):

log(phi) = par[1] + par[2] * log(pi.pre/pi.offset),

where `pi.offset` is 1e-4.

Under the NBQ model, the dispersion is modeled as a quadratic function of the preliminary estimates of the log mean relative frequencies (pi.pre):

log(phi) = par[1] + par[2] * z + par[3] * z^2,

where z = log(pi.pre/pi.offset). By default, pi.offset is the median of pi.pre[subset,].

Under this NBS model, the dispersion is modeled as a smooth function (a natural cubic spline function) of the preliminary estimates of the log mean relative frequencies (pi.pre).

Under the "step" model, the dispersion is modeled as a step (piecewise constant) function.

Value

a list with following components:

`estimates`	dispersion estimates for each read count, a matrix of the same dimensions as the `counts` matrix in `nb.data`.
`likelihood`	the likelihood of the fitted model.
`model`	details of the estimate dispersion model, NOT intended for use by end users. The name and contents of this component are subject to change in future versions.

Note

Currently, it is unclear whether a dispersion-modeling approach will outperform a more basic approach where regression model is fitted to each gene separately without considering the dispersion-mean dependence. Clarifying the power-robustness of the dispersion-modeling approach is an ongoing research topic.

Examples

```
## See the example for test.coefficient.
```

`estimate.norm.factors` *Estiamte Normalization Factors*

Description

`estimate.norm.factors` estiamtes normalization factors to account for apparent reduction or increase in relative frequencies of non-differentially expressing genes as a result of compensating the increased or decreased relative frequencies of truly differentially expressing genes.

Usage

```
estimate.norm.factors(counts, lib.sizes = colSums(counts),
  method = "AH2010")
```

Arguments

`counts`	a matrix of RNA-Seq read counts with rows corresponding to gene features and columns corresponding to independent biological samples.
`lib.sizes`	a vector of observed library sizes, usually and by default estimated by column totals.
`method`	a character string specifying the method for normalization, currenlty, can be NULL or "AH2010". If method=NULL, the normalization factors will have values of 1 (i.e., no normalization is applied); if method="AH2010" (default), the normalization method proposed by Anders and Huber (2010) will be used.

Details

We take gene expression to be indicated by relative frequency of RNA-Seq reads mapped to a gene, relative to library sizes (column sums of the count matrix). Since the relative frequencies sum to 1 in each library (one column of the count matrix), the increased relative frequencies of truly over expressed genes in each column must be accompanied by decreased relative frequencies of other genes, even when those others do not truly differentially express. If not accounted for, this may give a false impression of biological relevance (see, e.g., Robinson and Oshlack (2010), for some examples.) A simple fix is to compute the relative frequencies relative to effective library sizes—library sizes multiplied by normalization factors.

Value

a vector of normalization factors.

References

Anders, S. and W. Huber (2010): "Differential expression analysis for sequence count data," Genome Biol., 11, R106.

Robinson, M. D. and A. Oshlack (2010): "A scaling normalization method for differential expression analysis of RNA-seq data," Genome Biol., 11, R25.

Examples

```
## Load Arabidopsis data
data(arab)

## Estimate normalization factors using the method of Anders and Huber (2010)
norm.factors = estimate.norm.factors(arab);
print(norm.factors);
```

---

exact.nb.test *Exact Negative Binomial Test for Differential Gene Expression*

Description

`exact.nb.test` performs the Robinson and Smyth exact negative binomial (NB) test for differential gene expression on each gene and summarizes the results using p-values and q-values (FDR).

Usage

```
exact.nb.test(obj, grp1, grp2, print.level = 1)
```

Arguments

obj	output from estimate.disp.
grp1	Identifier of group 1. A number, character or string (should match at least one of the obj$grp.ids).
grp2	Identifier of group 2. A number, character or string (should match at least one of the obj$grp.ids).
print.level	a number. Controls the amount of messages printed: 0 for suppressing all messages, 1 for basic progress messages, larger values for more detailed messages.

Details

The negative binomial (NB) distribution offers a more realistic model for RNA-Seq count variability and still permits an exact (non-asymptotic) test for comparing expression levels in two groups.

For each gene, let S_1, S_2 be the sums of gene counts from all biological replicates in each group. The exact NB test is based on the conditional distribution of $S_1|S_1+S_2$: a value of S_1 that is too big or too small, relative to the sum $S_1 + S_2$, indicates evidence for differential gene expression. When the effective library sizes are the same in all replicates and the dispersion parameters are known, we can determine the probability functions of S_1, S_2 explicitly. The exact p-value is computed as the total conditional probability of all possible values of (S_1, S_2) that have the same sum as but are more extreme than the observed values of (S_1, S_2).

Note that we assume that the NB dispersion parameters for the two groups are the same and library sizes (column totals of the count matrix) are the same.

Value

the list obj from the input with the following added components:

grp1	same as input.
grp2	same as input.
pooled.pie	estimated pooled mean of relative count frequencies in the two groups being compared.
expression.levels	a matrix of estimated gene expression levels as indicated by mean relative read frequencies. It has three columns grp1, grp2, pooled corresponding to the two treatment groups and the pooled mean.
log.fc	base 2 log fold change in mean relative frequency between two groups.
p.values	p-values of the exact NB test applied to each gene (row).
q.values	q-values (estimated FDR) corresponding to the p-values.

Note

Before calling exact.nb.test, the user should call estimate.norm.factors to estimate normalization factors, call prepare.nbp to adjust library sizes, and call estimate.disp to fit a dispersion model. The exact NB test will be performed using pseudo.counts in the list obj, which are normalized and adjusted to have the same effective library sizes (column sums of the count matrix, multiplied by normalization factors).

Users not interested in fine tuning the underlying statistical model should use nbp.test instead. The all-in-one function nbp.test uses sensible approaches to normalize the counts, estimate the NBP model parameters and test for differential gene expression.

A test will be performed on a row (a gene) only when the total row count is nonzero, otherwise an NA value will be assigned to the corresponding p-value and q-value.

See Also

nbp.test.

Examples

```
## Load Arabidopsis data
data(arab);

## Specify treatment groups
## grp.ids = c(1, 1, 1, 2, 2, 2); # Numbers or strings are both OK
grp.ids = rep(c("mock", "hrcc"), each=3);

## Estimate normalization factors
norm.factors = estimate.norm.factors(arab);
print(norm.factors);

## Prepare an NBP object, adjust the library sizes by thinning the
## counts. For demonstration purpose, only use the first 100 rows of
## the arab data.
set.seed(999);
obj = prepare.nbp(arab[1:100,], grp.ids, lib.size=colSums(arab), norm.factors=norm.factors);
print(obj);

## Fit a dispersion model (NBQ by default)
obj = estimate.disp(obj);
plot(obj);

## Perform exact NB test
## grp1 = 1;
## grp2 = 2;
grp1 = "mock";
grp2 = "hrcc";

obj = exact.nb.test(obj, grp1, grp2);

## Print and plot results
print(obj);
par(mfrow=c(3,2));
plot(obj);
```

`irls.nb.1` *Estimate the regression coefficients in an NB GLM model*

Description

Estimate the regression coefficients in an NB GLM model with known dispersion parameters

Usage

```
irls.nb.1(y, s, x, phi, beta0 = rep(NA, p), mustart = NULL, maxit = 50,
  tol.mu = 0.001/length(y), print.level = 0)
```

Arguments

`y`	an n vector of counts
`s`	a scalar or an n vector of effective library sizes
`x`	an n by p design matrix
`phi`	a scalar or an n-vector of dispersion parameters
`mustart`	starting values for the vector of means
`beta0`	a vector specifying known and unknown components of the regression coefficients: non-NA components are hypothesized values of beta, NA components are free components
`maxit`	maximum number of iterations
`tol.mu`	a number, convergence criteria
`print.level`	a number, print level

Details

This function estimates the regression coefficients using iterative reweighted least squares (IRLS) algorithm, which is equivalent to Fisher scoring. The implementation is based on `glm.fit`.

Users can choose to fix some regression coefficients by specifying `beta0`. (This is useful when fitting a model under a null hypothesis.)

Value

a list of the following components:

`beta`	a p-vector of estimated regression coefficients
`mu`	an n-vector of estimated mean values
`conv`	logical. Was the IRLS algorithm judged to have converged?
`zero`	logical. Was any of the fitted mean close to 0?

nb.glm.test *Fit Negative Binomial Regression Model and Test for a Regression Coefficient*

Description

For each row of the input data matrix, nb.glm.test fits an NB log-linear regression model and performs large-sample tests for a one-dimensional regression coefficient.

Usage

```
nb.glm.test(counts, x, beta0, lib.sizes = colSums(counts),
  normalization.method = "AH2010", dispersion.model = "NBQ",
  tests = c("HOA", "LR", "Wald"), alternative = "two.sided",
  subset = 1:dim(counts)[1])
```

Arguments

counts	an m by n matrix of RNA-Seq read counts with rows corresponding to gene features and columns corresponding to independent biological samples.
x	an n by p design matrix specifying the treatment structure.
beta0	a p-vector specifying the null hypothesis. Non-NA components specify the parameters to test and their null values.
lib.sizes	a p-vector of observed library sizes, usually (and by default) estimated by column totals.
normalization.method	a character string specifying the method for estimating the normalization factors, can be NULL or "AH2010". If method=NULL, the normalization factors will have values of 1 (i.e., no normalization is applied); if method="AH2010", the normalization method proposed by Anders and Huber (2010) will be used.
dispersion.model	a character string specifying the dispersion model, and can be one of "NB2", "NBP", "NBQ" (default), "NBS" or "step".
tests	a character string vector specifying the tests to be performed, can be any subset of "HOA" (higher-order asymptotic test), "LR" (likelihood ratio test), and "Wald" (Wald test).
alternative	a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less".
subset	specify a subset of rows to perform the test on

Details

nbp.glm.test provides a simple, one-stop interface to performing a series of core tasks in regression analysis of RNA-Seq data: it calls estimate.norm.factors to estimate normalization factors;

it calls prepare.nb.data to create an NB data structure; it calls estimate.dispersion to estimate the NB dispersion; and it calls test.coefficient to test the regression coefficient.

To keep the interface simple, nbp.glm.test provides limited options for fine tuning models/parameters in each individual step. For more control over individual steps, advanced users can call estimate.norm.factors, prepare.nb.data, estimate.dispersion, and test.coefficient directly, or even substitute one or more of them with their own versions.

Value

A list containing the following components:

data	a list containing the input data matrix with additional summary quantities, output from prepare.nb.data.
dispersion	dispersion estimates and models, output from estimate.dispersion.
test	test results, output from test.coefficient.

Examples

```
## Load Arabidopsis data
data(arab);

## Specify treatment structure
grp.ids = as.factor(c(1, 1, 1, 2, 2, 2));
x = model.matrix(~grp.ids);

## Specify the null hypothesis
## The null hypothesis is beta[1]=0 (beta[1] is the log fold change).
beta0 = c(NA, 0);

## Fit NB regression model and perform large sample tests.
## The step can take long if the number of genes is large
fit = nb.glm.test(arab, x, beta0, subset=1:50);

## The result contains the data, the dispersion estimates and the test results
print(str(fit));

## Show HOA test results for top ten genes
subset = order(fit$test.results$HOA$p.values)[1:10];
cbind(fit$data$counts[subset,], fit$test.results$HOA[subset,]);

## Show LR test results
subset = order(fit$test.results$LR$p.values)[1:10];
cbind(fit$data$counts[subset,], fit$test.results$LR[subset,]);
```

nbp.test *NBP Test for Differential Gene Expression from RNA-Seq Counts*

Description

nbp.test fits an NBP model to the RNA-Seq counts and performs Robinson and Smyth's exact NB test on each gene to assess differential gene expression between two groups.

Usage

```
nbp.test(counts, grp.ids, grp1, grp2, norm.factors = rep(1, dim(counts)[2]),
  model.disp = "NBQ", lib.sizes = colSums(counts), print.level = 1, ...)
```

Arguments

counts	an n by r matrix of RNA-Seq read counts with rows corresponding to genes (exons, gene isoforms, etc) and columns corresponding to libraries (independent biological samples).
grp.ids	an r vector of treatment group identifiers (e.g. integers).
grp1	group 1 id
grp2	group 2 id
norm.factors	an r vector of normalization factors.
model.disp	a string, one of "NB2", "NBP" or "NBQ" (default).
lib.sizes	(unnormalized) library sizes
print.level	a number, controls the amount of messages printed: 0 for suppressing all messages, 1 (default) for basic progress messages, and 2 to 5 for increasingly more detailed messages.
...	optional parameters to be passed to estimate.disp, the function that estimates the dispersion parameters.

Details

nbp.test calls prepare.nbp to create the NBP data structure, perform optional normalization and adjust library sizes, calls estimate.disp to estimate the NBP dispersion parameters and exact.nb.test to perform the exact NB test for differential gene expression on each gene. The results are summarized using p-values and q-values (FDR).

Overview: For assessing evidence for differential gene expression from RNA-Seq read counts, it is critical to adequately model the count variability between independent biological replicates. Negative binomial (NB) distribution offers a more realistic model for RNA-Seq count variability than Poisson distribution and still permits an exact (non-asymptotic) test for comparing two groups.

For each individual gene, an NB distribution uses a dispersion parameter ϕ_i to model the extra-Poisson variation between biological replicates. Across all genes, parameter ϕ_i tends to vary with the mean μ_i. We capture the dispersion-mean dependence using a parametric model: NB2, NBP and NBQ. (See estimate.disp for more details.)

Count Normalization: We take gene expression to be indicated by relative frequency of RNA-Seq reads mapped to a gene, relative to library sizes (column sums of the count matrix). Since the relative frequencies sum to 1 in each library (one column of the count matrix), the increased relative frequencies of truly over expressed genes in each column must be accompanied by decreased relative frequencies of other genes, even when those others do not truly differentially express. Robinson and Oshlack (2010) presented examples where this problem is noticeable.

A simple fix is to compute the relative frequencies relative to effective library sizes—library sizes multiplied by normalization factors. By default, `nbp.test` assumes the normalization factors are 1 (i.e. no normalization is needed). Users can specify normalization factors through the argument `norm.factors`. Many authors (Robinson and Oshlack (2010), Anders and Huber (2010)) propose to estimate the normalization factors based on the assumption that most genes are NOT differentially expressed.

Library Size Adjustment: The exact test requires that the effective library sizes (column sums of the count matrix multiplied by normalization factors) are approximately equal. By default, `nbp.test` will thin (downsample) the counts to make the effective library sizes equal. Thinning may lose statistical efficiency, but is unlikely to introduce bias.

Value

a list with the following components:

`counts`	an n by r matrix of counts, same as input.
`lib.sizes`	an r vector, column sums of the count matrix.
`grp.ids`	an r vector, identifiers of treatment groups, same as input.
`grp1`, `grp2`	identifiers of the two groups to be compared, same as input.
`eff.lib.sizes`	an r vector, effective library sizes, lib.sizes multiplied by the normalization factors.
`pseudo.counts`	count matrix after thinning, same dimension as counts
`pseduo.lib.sizes`	an r vector, effective library sizes of pseudo counts, i.e., column sums of the pseudo count matrix multiplied by the normalization.
`phi`, `alpha`	two numbers, parameters of the dispersion model.
`pie`	a matrix, same dimension as `counts`, estimated mean relative frequencies of RNA-Seq reads mapped to each gene.
`pooled.pie`	a matrix, same dimenions as `counts`, estimated pooled mean of relative frequencies in the two groups being compared.
`expression.levels`	a n by 3 matrix, estimated gene expression levels as indicated by mean relative frequencies of RNA-Seq reads. It has three columns `grp1`, `grp2`, `pooled` corresponding to the two treatment groups and the pooled mean.
`log.fc`	an n-vector, base 2 log fold change in mean relative frequency between two groups.
`p.values`	an n-vector, p-values of the exact NB test applied to each gene (row).
`q.values`	an n-vector, q-values (estimated FDR) corresponding to the p-values.

Note

Due to thinning (random downsampling of counts), two identical calls to nbp.test may yield slightly different results. A random number seed can be used to make the results reproducible. The regression analysis method implemented in nb.glm.test does not require thinning and can also be used to compare expression in two groups.

Advanced users can call estimate.norm.factors, prepare.nbp, estimate.disp, exact.nb.test directly to have more control over modeling and testing.

References

Di, Y, D. W. Schafer, J. S. Cumbie, and J. H. Chang (2011): "The NBP Negative Binomial Model for Assessing Differential Gene Expression from RNA-Seq", Statistical Applications in Genetics and Molecular Biology, 10 (1).

Robinson, M. D. and G. K. Smyth (2007): "Moderated statistical tests for assessing differences in tag abundance," Bioinformatics, 23, 2881-2887.

Robinson, M. D. and G. K. Smyth (2008): "Small-sample estimation of negative binomial dispersion, with applications to SAGE data," Biostatistics, 9, 321-332.

Anders, S. and W. Huber (2010): "Differential expression analysis for sequence count data," Genome Biol., 11, R106.

Robinson, M. D. and A. Oshlack (2010): "A scaling normalization method for differential expression analysis of RNA-seq data," Genome Biol., 11, R25.

See Also

prepare.nbp, estimate.disp, exact.nb.test.

Examples

```
## Load Arabidopsis data
data(arab);

## Specify treatment groups and ids of the two groups to be compared
grp.ids = c(1, 1, 1, 2, 2, 2);
grp1 = 1;
grp2 = 2;

## Estimate normalization factors
norm.factors = estimate.norm.factors(arab);

## Set a random number seed to make results reproducible
set.seed(999);

## Fit the NBP model and perform exact NB test for differential gene expression.
## For demonstration purpose, we will use the first 100 rows of the arab data.
res = nbp.test(arab[1:100,], grp.ids, grp1, grp2,
  lib.sizes = colSums(arab), norm.factors = norm.factors, print.level=3);

## The argument lib.sizes is needed since we only use a subset of
## rows. If all rows are used, the following will be adequate:
```

```
##
## res = nbp.test(arab, grp.ids, grp1, grp2, norm.factors = norm.factors);

## Show top ten most differentially expressed genes
subset = order(res$p.values)[1:10];
print(res, subset);

## Count the number of differentially expressed genes (e.g. qvalue < 0.05)
alpha = 0.05;
sig.res = res$q.values < alpha;
table(sig.res);

## Show boxplots, MA-plot, mean-variance plot and mean-dispersion plot
par(mfrow=c(3,2));
plot(res);
```

`plot.nb.data` *Boxplot and scatterplot matrix of relative frequencies (after normalization)*

Description

Boxplot and scatterplot matrix of relative frequencies (after normalization)

Usage

```
## S3 method for class 'nb.data'
plot(x, ...)
```

Arguments

`x`	output from `prepare.nb.data`
`...`	currently not used

`plot.nb.dispersion` *Plot the estimated dispersion as a function of the preliminarily estimated mean relative frequencies*

Description

Plot the estimated dispersion as a function of the preliminarily estimated mean relative frequencies

Usage

```
## S3 method for class 'nb.dispersion'
plot(x, ...)
```

Arguments

x	output from estimate.dispersion
...	additional parameters, currently unused

---

plot.nbp *Diagnostic Plots for an NBP Object*

Description

For output from nbp.test, produce a boxplot, an MA plot, mean-variance plots (one for each group being compared), and mean-dispersion plots (one for each group being compared). On the mean-variance and the mean-dispersion plots, overlay curves corresponding to the estimated NBP model.

Usage

```
## S3 method for class 'nbp'
plot(x, ...)
```

Arguments

x	output from nbp.test.
...	for future use

See Also

nbp.test

Examples

```
## See the example for nbp.test
```

---

prepare.nb.data *Prepare the NB Data Structure for RNA-Seq Read Counts*

Description

Create a data structure to hold the RNA-Seq read counts and other relevant information.

Usage

```
prepare.nb.data(counts, lib.sizes = colSums(counts), norm.factors = rep(1,
  dim(counts)[2]), tags = NULL)
```

Arguments

counts	an mxn matrix of RNA-Seq read counts with rows corresponding to gene features and columns corresponding to independent biological samples.
lib.sizes	an n-vector of observed library sizes. By default, library sizes are estimated to the column totals of the matrix counts.
norm.factors	an n-vector of normalization factors. By default, have values 1 (no normalization is applied).
tags	a matrix of tags associated with genes, one row for each gene (having the same number of rows as counts.

Value

A list containing the following components:

counts	the count matrix, same as input.
lib.sizes	observed library sizes, same as input.
norm.factors	normalization factors, same as input.
eff.lib.sizes	effective library sizes (lib.sizes x norm.factors).
rel.frequencies	relative frequencies (counts divided by the effective library sizes).
tags	a matrix of gene tags, same as input.

---

prepare.nbp *Prepare the Data Structure for Exact NB test for Two-Group Comparison*

Description

Create the NBP data structure, (optionally) normalize the counts, and thin the counts to make the effective library sizes equal.

Usage

```
prepare.nbp(counts, grp.ids, lib.sizes = colSums(counts),
  norm.factors = NULL, thinning = TRUE, print.level = 1)
```

Arguments

counts	an n by r matrix of RNA-Seq read counts with rows corresponding to genes (exons, gene isoforms, etc) and columns corresponding to libraries (independent biological samples).
grp.ids	an r vector of treatment group identifiers (can be a vector of integers, chars or strings).
lib.sizes	library sizes, an r vector of numbers. By default, library sizes are estimated by column sums.

norm.factors	normalization factors, an r vector of numbers. If NULL (default), no normalization will be applied.
thinning	a boolean variable (i.e., logical). If TRUE (default), the counts will be randomly down sampled to make effective library sizes approximately equal.
print.level	a number, controls the amount of messages printed: 0 for suppressing all messages, 1 (default) for basic progress messages, and 2 to 5 for increasingly more detailed messages.

Details

Normalization

We take gene expression to be indicated by relative frequency of RNA-Seq reads mapped to a gene, relative to library sizes (column sums of the count matrix). Since the relative frequencies sum to 1 in each library (one column of the count matrix), the increased relative frequencies of truly over expressed genes in each column must be accompanied by decreased relative frequencies of other genes, even when those others do not truly differently express. Robinson and Oshlack (2010) presented examples where this problem is noticeable.

A simple fix is to compute the relative frequencies relative to effective library sizes—library sizes multiplied by normalization factors. Many authors (Robinson and Oshlack (2010), Anders and Huber (2010)) propose to estimate the normalization factors based on the assumption that most genes are NOT differentially expressed.

By default, prepare.nbp does not estimate the normalization factors, but can incorporate user specified normalization factors through the argument norm.factors.

Library Size Adjustment

The exact test requires that the effective library sizes (column sums of the count matrix multiplied by normalization factors) are approximately equal. By default, prepare.nbp will thin (downsample) the counts to make the effective library sizes equal. Thinning may lose statistical efficiency, but is unlikely to introduce bias.

Value

A list containing the following components:

counts	the count matrix, same as input.
lib.sizes	column sums of the count matrix.
grp.ids	a vector of identifiers of treatment groups, same as input.
eff.lib.sizes	effective library sizes, lib.sizes multiplied by the normalization factors.
pseudo.counts	count matrix after thinning.
pseduo.lib.sizes	effective library sizes of pseudo counts, i.e., column sums of the pseudo count matrix multiplied by the normalization.

Note

Due to thinning (random downsampling of counts), two identical calls to prepare.nbp may yield slightly different results. A random number seed can be used to make the results reproducible.

See Also

nbp.test

Examples

```
## See the example for exact.nb.test
```

`print.nb.data` *Print summary of the nb counts*

Description

Print summary of the nb counts

Usage

```
## S3 method for class 'nb.data'
print(x, ...)
```

Arguments

x	output from prepare.nb.data
...	additional parameters, currently not used

`print.nb.dispersion` *Print the estimated dispersion model*

Description

Print the estimated dispersion model

Usage

```
## S3 method for class 'nb.dispersion'
print(x, ...)
```

Arguments

x	output from from estimate.dispersion
...	additional parameters, currently unused

print.nb.test *Print output from* test.coefficient

Description

We simply print out the structure of x. (Currenlty the method is equivalent to print(str(x)).)

Usage

```
## S3 method for class 'nb.test'
print(x, ...)
```

Arguments

x	output from test.coefficient
...	currenty not used

print.nbp *Print summary of an NBP Object*

Description

Print contents of an NBP object, output from prepare.nbp, estimate.disp, or nbp.test.

Usage

```
## S3 method for class 'nbp'
print(x, subset = 1:10, ...)
```

Arguments

x	Output from prepare.nbp, estimate.disp, or nbp.test.
subset	indices of rows of the count matrix to be printed.
...	other parameters (for future use).

See Also

nbp.test.

Examples

```
## See the example for nbp.test
```

`test.coefficient` *Large-sample Test for a Regression Coefficient in an Negative Binomial Regression Model*

Description

`test.coefficient` performs large-sample tests (higher-order asymptotic test, likelihood ratio test, and/or Wald test) for testing regression coefficients in an NB regression model.

Usage

```
test.coefficient(nb, dispersion, x, beta0, tests = c("HOA", "LR", "Wald"),
  alternative = "two.sided", subset = 1:m, print.level = 1)
```

Arguments

`nb`	an NB data object, output from `prepare.nb.data`.
`dispersion`	dispersion estimates, output from `estimate.disp`.
`x`	an n by p design matrix describing the treatment structure
`beta0`	a p-vector specifying the null hypothesis. Non-NA components specify the parameters to test and their null values. (Currently, only one-dimensional test is implemented, so only one non-NA component is allowed).
`tests`	a character string vector specifying the tests to be performed, can be any subset of `"HOA"` (higher-order asymptotic test), `"LR"` (likelihood ratio test), and `"Wald"` (Wald test).
`alternative`	a character string specifying the alternative hypothesis, must be one of `"two.sided"` (default), `"greater"` or `"less"`.
`subset`	an index vector specifying on which rows should the tests be performed
`print.level`	a number controlling the amount of messages printed: 0 for suppressing all messages, 1 (default) for basic progress messages, and 2 to 5 for increasingly more detailed message.

Details

`test.coefficient` performs large-sample tests for a one-dimensional ($q = 1$) component ψ of the p-dimensional regression coefficient β. The hypothesized value ψ_0 of ψ is specified by the non-NA component of the vector `beta0` in the input.

The likelihood ratio statistic,

$$\lambda = 2(l(\hat{\beta}) - l(\tilde{\beta})),$$

converges in distribution to a chi-square distribution with 1 degree of freedom. The signed square root of the likelihood ratio statistic λ, also called the directed deviance,

$$r = sign(\hat{\psi} - \psi_0)\sqrt{\lambda}$$

converges to a standard normal distribution.

For testing a one-dimensional parameter of interest, Barndorff-Nielsen (1986, 1991) showed that a modified directed

$$r^* = r - \frac{1}{r}\log(z)$$

is, in wide generality, asymptotically standard normally distributed to a higher order of accuracy than the directed deviance r itself, where z is an adjustment term. Tests based on high-order asymptotic adjustment to the likelihood ratio statistic, such as r^* or its approximation, are referred to as higher-order asymptotic (HOA) tests. They generally have better accuracy than corresponding unadjusted likelihood ratio tests, especially in situations where the sample size is small and/or when the number of nuisance parameters $(p-q)$ is large. The implementation here is based on Skovgaard (2001). See Di et al. 2013 for more details.

Value

a list containing the following components:

`beta.hat`	an m by p matrix of regression coefficient under the full model
`mu.hat`	an m by n matrix of fitted mean frequencies under the full model
`beta.tilde`	an m by p matrix of regression coefficient under the null model
`mu.tilde`	an m by n matrix of fitted mean frequencies under the null model.
`HOA`, `LR`, `Wald`	each is a list of two m-vectors, `p.values` and `q.values`, giving p-values and q-values of the corresponding tests when that test is included in `tests`.

References

Barndorff-Nielsen, O. (1986): "Infereni on full or partial parameters based on the standardized signed log likelihood ratio," Biometrika, 73, 307-322

Barndorff-Nielsen, O. (1991): "Modified signed log likelihood ratio," Biometrika, 78, 557-563.

Skovgaard, I. (2001): "Likelihood asymptotics," Scandinavian Journal of Statistics, 28, 3-32.

Di Y, Schafer DW, Emerson SC, Chang JH (2013): "Higher order asymptotics for negative binomial regression inferences from RNA-sequencing data". Stat Appl Genet Mol Biol, 12(1), 49-70.

Examples

```
## Load Arabidopsis data
data(arab);

## Estimate normalization factors (we want to use the entire data set)
norm.factors = estimate.norm.factors(arab);

## Prepare the data
## For demonstration purpose, only the first 50 rows are used
nb.data = prepare.nb.data(arab[1:50,], lib.sizes = colSums(arab), norm.factors = norm.factors);

## For real analysis, we will use the entire data set, and can omit lib.sizes parameter)
## nb.data = prepare.nb.data(arab, norm.factors = norm.factors);

print(nb.data);
plot(nb.data);
```

```
## Specify the model matrix (experimental design)
grp.ids = as.factor(c(1, 1, 1, 2, 2, 2));
x = model.matrix(~grp.ids);

## Estimate dispersion model
dispersion = estimate.dispersion(nb.data, x);

print(dispersion);
plot(dispersion);

## Specify the null hypothesis
## The null hypothesis is beta[2]=0 (beta[2] is the log fold change).
beta0 = c(NA, 0);

## Test regression coefficient
res = test.coefficient(nb.data, dispersion, x, beta0);

## The result contains the data, the dispersion estimates and the test results
print(str(res));

## Show HOA test results for top ten most differentially expressed genes
top = order(res$HOA$p.values)[1:10];
print(cbind(nb.data$counts[top,], res$HOA[top,]));

## Plot log fold change versus the fitted mean of sample 1 (analagous to an MA-plot).
plot(res$mu.tilde[,1], res$beta.hat[,2]/log(2), log="x",
     xlab="Fitted mean of sample 1 under the null",
     ylab="Log (base 2) fold change");

## Highlight top DE genes
points(res$mu.tilde[top,1], res$beta.hat[top,2]/log(2), col="magenta");
```

Index