Package 'texmexseq'

August 29, 2016

Type Package

Title Treatment Effect eXplorer for Microbial Ecology eXperiments (using Sequence Counts)

Version 0.3

Date 2016-07-25

Author Scott Olesen

Maintainer Scott Olesen <swo@mit.edu>

Description Analysis and visualization of community dynamics in microbial ecology experiments (that use sequence count data) using the truncated Poisson lognormal distribution.

License GPL-3

Depends dplyr (>= 0.4.2), ggplot2 (>= 1.0.1),

Imports testthat (>= 1.0.2)

URL http://almlab.mit.edu/texmex.html

NeedsCompilation yes

Repository CRAN

Date/Publication 2016-07-25 20:33:40

R topics documented:

texmexseq-package *Treatment Effect eXplorer for Microbial Ecology eXperiments*

Description

Fit OTU count data to the Poisson lognormal distribution to create values that can be compared across control and treatment units in before and after timepoints. Plotting methods for these "quads" of samples.

Details

Package:	texmexseq
Type:	Package
Version:	0.1
Date:	2015-03-10
License:	GPL-3

`texmexseq` is essentially a wrapper around the earlier R package `poilog`. The implementations of the basic functions related to the Poisson lognormal distribution in this package are nearly identical to the ones in `poilog`. There are convenience functions, notably `Experiment` deisgned to directly interact with OTU tables.

Author(s)

Scott Olesen <swo@mit.edu>

dpoilog *probability density function for the Poisson lognormal distribution*

Description

Probability density function for the Poisson lognormal distribution, along with methods for drawing random samples from that distribution and fitting experimental data to that distribution.

Usage

```
dpoilog(n, mu, sig, trunc=TRUE)
rpoilog(S, mu, sig, condS=FALSE, keep0=FALSE)
```

Arguments

n	vector of observed counts
S	number of OTUs to sample
mu	mean of lognormal part of distribution
sig	standard deviation of lognormal part of distribution
trunc	remove weight from zero counts?
condS	are random samples conditional on S?
keep0	do not discard zero counts?

Details

A detailed description of the calculation of these values can be found in the documentation for the dpoilog in the poilog package.

Value

dpoilog	returns that densities
rpoilog	returns the random counts

Author(s)

Scott Olesen <swo@mit.edu>

See Also

poilogMLE

Examples

```
# visualize the density function
plot(dpoilog(0:100, mu=1.0, sig=1.0), type='o')

# visualize the empirical distribution of a random sample of 1000 OTUs
hist(rpoilog(1000, mu=1.0, sig=1.0, condS=TRUE))
```

---

poilogMLE *maximum likelihood estimation for the Poisson lognormal*

Description

Fit a set of OTU counts to the Poisson lognormal distribution.

Usage

```
poilogMLE(n, start.mu, start.sig, trunc=TRUE, method='L-BFGS-B',
control=list(fnscale=length(n)), ...)
```

Arguments

n	vector of observed counts
start.mu	initial estimate for the lognormal mean
start.sig	initial estimate for the lognormal standard deviation
trunc	remove weight from zero counts?
method	optimization method for optim
control	list of parameters for optim
...	further parameters to go to optim

Details

The function estimates parameters mean mu and standard deviation sig.

The parameters must be given starting values for the optimization procedure. The default values here worked well when fitting OTUs in the referenced paper.

The function uses the optimization procedures in optim to make the maximum likelihood estimate. The method and control arguments are passed to optim.

A zero-truncated distribution (see dpoilog) is assumed by default. Truncation should only be turned off if all the known OTUs of the sequenced community are known. In most cases, this is not applicable, since it is usually not possible to know if an OTU had zero counts because it is not present in the environment or if it is present but in low abundance.

Value

par	Maximum likelihood estimates of the parameters
p	Approximate fraction of OTUs revealed by the sample
logLval	Log likelihood of the data given the estimated parameters

Author(s)

Scott Olesen <swo@mit.edu>

See Also

dpoilog optim

Examples

```
# create some random data
x <- rpoilog(S=1000, mu=-2.0, sig=2.0, keep0=FALSE)

# fit that data
res <- poilogMLE(x, 2.0, -2.0)

# the results should be fairly robust to the starting parameters
res2 <- poilogMLE(x, 1.0, 0.5)
```

ppplot *PP plot for Poisson lognormal distribution*

Description

Generate a plot showing the quality of the fit of the Poisson lognormal to the data.

Usage

```
ppplot(n, n.points=10)
```

Arguments

n	vector of observed counts
n.points	number of points on the graph to highlight

Details

The function fits the Poisson lognormal to the raw read counts n and uses that to generate theoretical percents for a PP ("percent-percent" or "probability-probability") plot. A perfect fit falls on the diagonal (marked with a dotted line).

The optional n.points plots some extra points on the graph. The first point (the lower-left-most) represents the fraction of all OTUs that have 1 count in the empirical (x-axis) and theoretical (y-axis) distributions. The second point represents OTUs with 1 or 2 counts. The third point represents OTUs with up to 3 counts, and so on.

Value

a ggplot object whose default data object has columns empirical and theoretical

Author(s)

Scott Olesen <swo@mit.edu>

See Also

texmex.fit

Examples

```
# make up some data
n <- rpoilog(1000, 1.0, 1.0)

# plot it
p <- ppplot(n)
p

# compare to the lognormal's fit
```

```
# first, make the empirical cumulative distribution function from the data
x <- tabulate(n + 1)
empirical <- cumsum(x / sum(x))

# then, get the theoretical percents
theoretical <- plnorm(0:max(n), meanlog=mean(log(n)), sdlog=sd(log(n)))
lognormal.fit <- data.frame(empirical=empirical, theoretical=theoretical)

# add that data in a new layer
p + geom_line(data=lognormal.fit, color='red')
```

`quad.table` *Create and plot "quads" of samples*

Description

Pull out a "quad" of samples from a larger data frame, then plot the pairs of samples in the quad against one another.

Usage

```
quad.table(otu, control.before, control.after, treatment.before, treatment.after)
 quad.plot(quad)
```

Arguments

`otu`	data frame of a z- or F-transformed OTU table
`control.before`	name of the sample (in the OTU table) from the control unit before the treatment is applied to the treatment unit
`control.after`	sample from the control unit after treatment is applied
`treatment.before`	sample from the treatment unit before the treatment is applied
`treatment.after`	sample from the treatment unit after the treatment is applied
`quad`	a quad generated by `quad.table`

Details

`texmexseq` was designed to compare four samples at a time: two are "control" samples (before and after the treatment was applied to the "treatment" samples), the other two are the treatment samples.

`quad.table` will grab the columns with the four given names and make them into a new data frame (with OTU IDs kept as a separate column). This object can be plugged into `quad.plot` for viewing. `quad.plot` expects that the OTU table will be z- or F-transformed.

Value

quad.table	returns a data frame
quad.plot	returns a ggplot object

Author(s)

Scott Olesen <swo@mit.edu>

See Also

z.transform.table f.transform.table

Examples

```
# make up some data
sim.data <- function() rpoilog(1000, 1.0, 1.0, condS=TRUE)
otu <- data.frame(sample0=sim.data())
for (i in 1:10) otu[[paste('sample', i, sep='')]] <- sim.data()
otu.ids <- paste('otu', seq(1:1000), sep='')
rownames(otu) <- otu.ids
z.table <- z.transform.table(otu)

# pull out a quad, imagining that samples 1 and 2 were the control samples
# and 3 and 4 were the treatment
q <- quad.table(z.table, 'sample1', 'sample2', 'sample3', 'sample4')

# plot it
p <- quad.plot(q)
p

# ok, it's just a blob because we generated the data, but imagine we
# were particularly interested in OTUs that bloomed in the treatment
# but not in the control
interesting.otus <- filter(q, d.treatment > 2, d.control < 0)

# we can plot those in a different color
p + geom_point(data=interesting.otus, color='red')

# or see what their names are
head(arrange(interesting.otus, desc(d.treatment)))
```

---

texmex.fit *wrapper for fitting the Poisson lognormal*

Description

A nice wrapper for the poilogMLE function.

Usage

```
texmex.fit(n, start.mus=c(-2.0, -1.0, 0.0, 1.0, 2.0), start.sigs=rep(1.0, times=5), ...)
```

Arguments

n	vector of observed counts
start.mus	vector of starting mu values
start.sigs	vector of starting sig values
...	further parameters to go to poilogMLE

Details

It can be a little annoying to direclty use poilogMLE: the optimization can fail to converge if you choose a bad starting mu value. This function makes multiple attempts at fitting the poilog distribution, returning the first one that works.

For the first attempt, the first mu and sig are used as starting values. For the second attempt, the second elements of those vectors, etc.

Value

par	Maximum likelihood estimates of the parameters
p	Approximate fraction of OTUs revealed by the sample
logLval	Log likelihood of the data given the estimated parameters

Author(s)

Scott Olesen <swo@mit.edu>

See Also

poilogMLE

Examples

```
# create some random data
x <- rpoilog(S=1000, mu=-2.0, sig=2.0, keep0=FALSE)

# fit that data
res <- texmex.fit(x)
```

`z.transform.sample` *transform from raw counts to Poisson-lognormal-based metrics*

Description

Functions for transforming raw read counts into rescaled reads (`z`) and theoretical cumulative distribution function values (`f`).

Usage

```
z.transform.sample(n)
z.transform.table(otu, ignore='otu')
f.transform.sample(n)
f.transform.table(otu, ignore='otu')
```

Arguments

`n`	vector of observed counts
`otu`	data frame OTU table
`ignore`	columns that should not be transform (e.g., OTU IDs) have names that match this pattern (i.e., those columns are excluded using `dplyr`'s `-matches(ignore)`

Details

`z.transform.sample` fits the Poisson lognormal distribution to the count data and uses that fit to transform those raw read counts into rescaled reads.

`z.transform.table` performs the same function on an OTU table, which should have one sample per column and, potentially, an ID column that matches the `ignore` option (e.g., `'otu'` will match `OTU` or `OTU_ID`). This function is just for convenience: it applies `z.transform.sample` to each column (that does not match the `ignore` option) and packages the result into a data frame. Because it uses a `dplyr` function, the rownames will be lost.

`f.transform.sample` and `f.transform.table` are analogous to the `z` functions, only they return theoretical cumulative distribution function values.

The resulting tables could be used on their own for analysis, but `texmexseq` is designed to slice that data into smaller "quads" (using `quad.table`).

Value

`z.transform.sample`
: returns a vector of transformed values

`z.transform.table`
: returns a data frame of transformed values

Author(s)

Scott Olesen <swo@mit.edu>

See Also

texmex.fit quad.table

Examples

```
# make up a table of data
sim.data <- function() rpoilog(100, 1.0, 1.0, condS=TRUE)
sample1 <- sim.data()

# transform it
hist(f.transform.sample(sample1))

# make up a table of data
sample2 <- sim.data()
otu.ids <- paste('otu', seq(1, 100), sep='')
otu.table <- data.frame(OTU_ID=otu.ids, sample1=sample1, sample2=sample2)

# make a new table from those fitted values
f.table <- f.transform.table(otu.table)
hist(f.table$sample1)
```

Index