Package 'lymphclon'

June 15, 2013

Version 1.0.12

Date 2013-06-14

Title Estimating the Clonality Score, a Measure of Conincidences with Replicated Abundances, with Applications in the Study of Lymphocytes

Author Yi Liu, Richard Olshen, Andrew Fire, Scott Boyd

Maintainer Yi Liu <liuyipei@stanford.edu>

Depends R (>= 2.15.0), VGAM, MASS, expm, Matrix, corpcor

Imports VGAM, MASS, expm, Matrix, corpcor

Suggests VGAM, MASS, expm, Matrix, corpcor

Description We provide a clonality score estimator that takes full advantage of the multi-biological-replicate structure of modern sequencing experiments; it specifically takes into account the reality that, typically, the clonal coverage is well below 0.1%.

License LGPL-2

URL http://www.r-project.org, http://www.another.url

BugReports http://pkgname.bugtracker.url

R topics documented:

lymphclon-package 2
infer.clonality 3
simulate.clonality.data 4

Index **6**

`lymphclon-package` *Estimates the clonality score from replicate of abundances data*

Description

There are an enormous number of clones(species) distributed in a highly unequal distribution. The experimenters collect a small number of very coarse samplings, and perform noisy measurements on the samplings, leading to abundance estimations for the indiviudal biological replicates. The goal of this package is to estimate the probability that two random cells belong to the same clone. Clonality estimation arises as a naturally interesting question when trying to understand the diversity of B cell populations, T cell populations, microbial populations, cancer cell subclones, and artificial biologicl libraries, and also broader ecological settings.

This package provides two primary functions; one computes the clonality score estimate (probability that two random individuals belong to the same clone), given replicate of abundances; it also infers the underlying clonal distribution, which may be of interest in diagnostic and scientific settings.

The other function generates reasonable simulation data, for evaluation purposes. The default behavior of this function generates a very large number of classes, which is suitable for B and T cell settings. It can be easily repurposed for other applications. The simulation assumes random sampling, amplifying, and sequencing of individual cells; this is implemented by introducing sparse sampling of individual cells, followed by introduction of log-normal error, and finally unbiased "rounding" by use of a poisson distribution.

Details

Package:	lymphclon
Type:	Package
Version:	1.0.4
Date:	2013-06-04
License:	LGPL-2

Author(s)

Author and Maintainer: Yi Liu <liuyipei@stanford.edu>

References

~~ Literature or other references for background information ~~

See Also

<vegan>

Examples

```
my.data<-simulate.clonality.data(n=2e3)
# n ~ 2e7 is more appropriate for a realistic B cell repertoire
```

```
infer.clonality(my.data$read.count.matrix)
```

`infer.clonality` *infer.clonality (part of lymphclon package)*

Description

Clonality score, a useful metric used in immunology, refers to the probability that two random lymphocyte receptor chain reads drawn with replacement (makes no difference in the immunology context) from an individual corresponded to the same clone, within some given repertoire of either B cells or T cells. This package implements ane estimator which understands the multi-replicate-with-PCR structure of thes sequencing experiments.

Usage

```
infer.clonality(
  read.count.matrix,
  variance.method = 'fpc.1',
  estimate.abundances = F,
  loo.squared.err.est = c(),
  use.squared.err.est = c(),
  num.iterations = 1,
  regularization.method = '')
```

Arguments

`read.count.matrix`
: A matrix of read frequences, where each row corresponds to a distinct clone, and each column corresponds to a particular biological (rather than technical) replicate. All biological replicates should be drawn from the same person. Reads from technical replicates of the same underlying templates should be merged into a single column.

`variance.method`
: The method is a code defaulting to "loo.2". usr.1: argument use.squared.err.est specifies 1st order variances usr.2: argument use.squared.err.est specifies 2nd order variances mle.1: use maximum likelihood estimate of 1st order variances mle.2: use maximum likelihood estimate of 2nd order variances loo.1: use leave-one-out estimate of 1st order variances loo.2: use leave-one-out estimate of 2nd order variances corpcor.1: use shrinkage-based estimate of 1st order variances (see package corpcor) corpcor.2: use shrinkage-based estimate of 2nd order variances (see package corpcor)

`estimate.abundances`
: A boolean value defaulting to false. If set to true, then the return value will be in the form of a list, and include an additional item named "estiamted.abundances". The estimated abundances are computed as the (conditional) precision weighted average of the read.count.matrix.

`loo.squared.err.est`
: A boolean value defaulting to true. When estimating the squared errors and precisions associated with each replicate, we can use the maximum likelihood estimate (i.e. set loo.squared.err.est to false), or use an averaged leave-one-out estimate (i.e. set loo.squared.err.est to true). The leave-one-out estimate is

slightly biased, and slightly slower to compute, but much less variance. Especially if there are at least three replicates, this value probably should be set to true.

`use.squared.err.est`
: A vector defaulting to the empty vector. If a non-empty vector is provided, then this forces infer.clonality to use these values as the conditional variances of the respective replicates. If non-empty, then this vector should be positive, and of length equal to the number of replicates.

`num.iterations`
: An integer specifying the number of iterations in estimation; applicable only if the variance method is set to one of the loo or mle methods.

`regularization.method`
: A flag which determines the method by which the covariance of the pairwise comparisons are regularized. Current options are '', 'ue.zr.full', 'eq.zr.half', 'ue.zr.half', 'eq.eq'half', 'ue.eq.half'.

Value

`simple.precision.clonality`
: This is a base line estimator of the clonality score based on simple modeling. Briefly, for each pairing of reads possible, where the two reads arise from different biological replicates – this pairing is considered as an observation from a single share undelrying bernoulli distribution with parameter equal to the clonality score. This estimator computes maximum likelihood estimate of the underlying clonality score. This estimator can be thought of as a baseline estimate. Unlike the Gini-Simpson-Estimator, this baseline does not suffer from any convexity-based bias.

`rao.blackwell.mvg.clonality`
: This is an estimator is a weighted average of n-choose-2 individually unbiased estimators that arise from comparing the provided n biological replicates pairwise. The weighting is determined by the covariance between these estimators. This has much lower variance than the simple.precision.clonality.

Examples

```
my.data<-simulate.clonality.data(n=2e3)
# n ~ 2e7 is more appropriate for a realistic B cell repertoire
infer.clonality(my.data$read.count.matrix)
```

`simulate.clonality.data`
simulate.clonality.data (part of lymphclon package)

Description

This function generates simulated data, for evaluation purposes. We start with an underlying multinomial population with entries proportional to rank^-power distribution, where power is fixed. Next, we draw multinomially from this distribution, a fixed number of cells, to generate each desired replicate. Then, this distribution is subject to log-normal error, and subsequently scaled up to the expected number of reads. To round into integers, the expected number of reads for each clone is finally pushed through a poisson process to generate integer read counts. The poisson "rounding" process is why the resulting read counts are not exactly as specified.

Usage

```
simulate.clonality.data(
  n = 2e+07,
  num.cells.taken.vector = c(2000, 5000, 10000, 20000, 50000, 50000),
  read.count.per.replicate.vector = rep(20000, length(num.cells.taken.vector)),
  clonal.distribution.power = -sqrt(2))
```

Arguments

`n`	The true number of distinct clones in the underlying assemblage
`num.cells.taken.vector`	A vector specifying the number of cells taken in each independent biological replicate
`read.count.per.replicate.vector`	A vector of the same length as num.cells.taken.vector, specifying the number of reads generated from each biological replicate, of the same corresponding indices
`clonal.distribution.power`	The true underlying clonal multinomial distribution is proportional to (1:n)^-clonal.distribution.power

Value

`read.count.matrix`	This is a matrix of simulated counts, with rows corresponding to clones (classes, or species), and columns corresponding to biological replicates
`true.clone.prob`	This is the underlying simulated assemblage multinomial distribution used to generate read.count.matrix
`true.clonality`	This is the true clonality score of the underlying simulated assemblage

Examples

```
my.data<-simulate.clonality.data(n=2e3)
# n ~ 2e7 is more appropriate for a realistic B cell repertoire
infer.clonality(my.data$read.count.matrix)
```

Index

∗Topic **\textasciitildekwd1**
infer.clonality, 3
simulate.clonality.data, 4
∗Topic **\textasciitildekwd2**
infer.clonality, 3
simulate.clonality.data, 4
∗Topic **diversity, clonality score, clonality**
lymphclon-package, 2
<vegan>, *2*

infer.clonality, 3

lymphclon *(*lymphclon-package*)*, 2
lymphclon-package, 2

simulate.clonality.data, 4