Package 'tripsAndDipR'

July 22, 2025

Type Package

Title Identification of 2n and 3n Samples from Amplicon Sequencing Data

Version 0.1.0

Description Uses read counts for biallelic single nucleotide polymorphisms (SNPs) to compare the likelihoods for the observed read counts given that a sample is either diploid or triploid. It allows parameters to be specified to account for sequencing error rates and allelic bias. For details of the algorithm, please see Delomas (2019) <doi:10.1111/1755-0998.13073>.

Imports stats

URL https://github.com/delomast/tripsAndDipR

BugReports https://github.com/delomast/tripsAndDipR/issues

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

NeedsCompilation no

Author Thomas Delomas [aut, cre]

Maintainer Thomas Delomas <thomas.delomas@idfg.idaho.gov>

Repository CRAN

Date/Publication 2019-08-28 09:40:03 UTC

Contents

`tripsAndDip` *Uses read counts for biallelic SNPs to determine if a sample is diploid or triploid*

Description

`tripsAndDip` calculates log-likelihood ratios comparing whether a sample is likely diploid or triploid based on the read counts for biallelic SNPs.

Usage

```
tripsAndDip(counts, counts_alt = NA, h, eps, min_reads = 30,
  min_loci = 15, binom_p_value = 0.05)
```

Arguments

`counts`	Either a numeric matrix or a dataframe with each row corresponding to a different sample. There are two options for formatting the input. Either the columns correspond to the read counts for each locus, in a two column per locus format: column 1 is the read counts for locus1ReferenceAllele, column two is the read counts for locus1AlternateAllele2, locus2Reference, locus2Alternate, ... OR this contains read counts for the reference allele, and `counts_alt` contains read counts for the alternate allele The rownames should be the sample names.
`counts_alt`	This is a numeric matrix or a dataframe with each row corresponding to a different sample. The matrix contains counts for the alternate allele, with samples and loci having the same order as in `counts` If this parameter is NA or NULL, `counts` is assumed to have both the reference and alternate allele counts.
`h`	A numeric vector of h values for each locus in the same order that the loci are ordered in counts. These h values are as defined by Gerard et al. (2018) "Genotyping polyploids from messy sequencing data" Genetics 210:789-807. with h expressed as alternate / reference. These values can be estimated using the R package "updog".
`eps`	A numeric vector of values for the error rate per read for each locus in the same order that the loci are ordered in counts. These are expressed as proportions, so a rate of 1% should be given as 0.01. These values can be estimated using the R package "updog".
`min_reads`	The minimum number of reads to consider a locus.
`min_loci`	The minimum number of usable loci in a sample to calculate a log-likelihood ratio.
`binom_p_value`	The alpha value to use when applying a binomial test to determine whether to include a locus in the calculation.

Details

tripsAndDip calculates log-likelihood ratios comparing the likelihoods of the read counts under diploidy or triploidy for a sample using biallelic SNPs.This function was designed with amplicon sequencing data in mind, but may be useful for other genotyping techniques that also yield read counts for each allele in a given locus. Full details of the calculations can be found in Delomas (2019) Differentiating diploid and triploid individuals using single nucleotide polymorphisms genotyped by amplicon-sequencing. Molecular Ecology Resources.

Value

a dataframe with column 1 containing sample names, column 2 containing calculated LLRs (larger means more likely given triploidy) and column 3 containing the number of loci used to calculate the LLR

Examples

```
# make up some data
triploid_allele1 <- rbinom(60, 75, 2/3)
triploid_allele2 <- 75 - triploid_allele1
diploid_allele1 <- rbinom(60, 75, 1/2)
diploid_allele2 <- 75 - diploid_allele1
# interleave allele counts
triploid <- c(rbind(triploid_allele1, triploid_allele2))
diploid <- c(rbind(diploid_allele1, diploid_allele2))

# create counts matrix
allele_counts <- matrix(data = c(triploid, diploid), byrow = TRUE, nrow = 2, ncol = 120)
rownames(allele_counts) <- c("triploid", "diploid")

#create h and eps vectors
h_constant <- rep(1, 60)
eps_constant <- rep(.01, 60)

#run function
ploidy <- tripsAndDip(allele_counts, h = h_constant, eps = eps_constant)
```

Index