Package 'Amplicomsat'

March 1, 2024

Type Package

Title Processes microsatellite genotypes from Amplicon generated fastq read files

Version 0.1.13

Date 2024-03-1

Author Filipe Alberto

Maintainer Filipe Alberto <albertof@uwm.edu>

Description A series of functions that can all be wrapped sequentially in a single script to extract microsatellite genotypes from fastq files with Ampicon sequencing reads. It can extract mutiple alleles per sample loci combinations and identify sequence variant alleles.

Imports stringr, readr, Biostrings

License GPL (>=2)

NeedsCompilation no

R topics documented:

AlleleCum *Plots cummulative distributions of scored alleles*

Description

Plots cummulative distributions of scored alleles to help diagnose any possible errors

Usage

```
AlleleCum(GenotypeA.DF="./GenotypesOut/Genotypes allele table.txt",locus,locusInfo="primers",
          ymin=NULL,ymax=NULL)
```

Arguments

GenotypeA.DF	A data.frame as produced by alleleFinder
locus	A string with the name of the locus to plot. The name needs to be written as in the file provided to locusInfo argument to other functions, e.g., read.size.freqs
locusInfo	A tab delimited text file containing the information for the microsatellite loci used. Should contain four columns with the following header names: Locus: The loci names Fprimer: The sequence of the FWD primer in upper case Rprimer: The sequence of the RVS primer in upper case Motif: The sequence for the microsatellite motif (e.g., CTTG for tetranucleotide). Note that for some loci you might need to use the reverse complementar sequence of what your reference value is. Thus, inspect your reads first or if you are not getting data for some loci try this out. For example, for a motif CAGG, you might need to use CCTG.
ymin	The y lower lim for the y axis. Usefull for loci that are very polymorphic and need to be zoomed in.
ymax	The y upper lim for the y axis. Usefull for loci that are very polymorphic and need to be zoomed in.

Details

Plots the cumulative distribution of alleles scored for the selected locus stored in the data.frame supplied to argument GenotypeA.DF. The sample names for specific observations can be obtained by interacting with the plot using function getpoints.

Value

Plots the cumulative allele distribution for a selected locus

Author(s)

F. Alberto

References

Alberto F (in prep)...

See Also

getpoints

`alleleFinder` *Scores the diploid genotypes into an output table*

Description

Estimates the diploid genotypes based on read size (primer to primer) in bp. The results are written to two files: one with two alleles per column and the other with the two alleles collapsed in a single column. The files are written to a folder named GenotypesOut created by the function.

Usage

```
alleleFinder(sampleDir="./samplesF",locusInfo="primers",coverageF=25)
```

Arguments

`sampleDir`	The directory where the filtered reads are stored. This directory should only contain the read files and nothing else. By default this is a directory named samplesF written in the session working directory, as produced by `read.quality.filter`
`locusInfo`	A tab delimited text file containing the information for the microsatellite loci used. Should contain four columns with the following header names: Locus: The loci names Fprimer: The sequence of the FWD primer in upper case Rprimer: The sequence of the RVS primer in upper case Motif: The sequence for the microsatellite motif (e.g., CTTG for tetranucleotide). Note that for some loci you might need to use the reverse complementar sequence of what your reference value is. Thus, inspect your reads first or if you are not getting data for some loci try this out. For example, for a motif CAGG, you might need to use CCTG.
`coverageF`	Reads scored as alleles based on the sum of squares minimization method need to have coverage larger than this value, otherwise are written as missing data NA. Defaults to coverage of 25. It is possible to send a vector with different values for each locus. If you want to use the same coverage filter for all loci you can input a single value.

Details

Uses the tables created by GenotypeModel, containing expected relative frequencies of read cover for different combinations (2 out of N) of alleles, to contrast with observed data in each sample and find the diploid genotypes minimizing a sum of squares. For each observed sample, the row table that minimizes the sum of squares: sum((ORF(a)-ERF(a))^2), with ORF and ERF the Observed and Expected relative frequencies at each read size, is the estimated genotype.

Value

A data frame with a concatenated genotype (the two alleles in the same cell) is returned. This table is also written to file, as well as a version with the two alleles in one locus in different columns (two columns per locus). The output files are written to a folder named GenotypesOut that is created by the function in the R session working directory.

Author(s)

F. Alberto

References

Alberto F (in prep)...

See Also

GenotypeModel, AlleleCum

`GenotypeModel` *Creates tables with expected relative frequencies of read sizes*

Description

Creates tables with expected relative frequencies of read sizes for each possible diploid genotype within an allelic range set by the user.

Usage

```
GenotypeModel(sampleDir="./samplesF",locusInfo="primers",
              calibrationFiles="CalibrationSamples.txt")
```

Arguments

`sampleDir`	The directory where the filtered reads are stored. This directory should only contain the read files and nothing else. By default this is a directory named samplesF written in the session working directory, as produced by `read.quality.filter`

`locusInfo` A tab delimited text file containing the information for the microsatellite loci used. Should contain four columns with the following header names:

Locus: The loci names

Fprimer: The sequence of the FWD primer in upper case

Rprimer: The sequence of the RVS primer in upper case

Motif: The sequence for the microsatellite motif (e.g., CTTG for tetranucleotide). Note that for some loci you might need to use the reverse complementar sequence of what your reference value is. Thus, inspect your reads first or if you are not getting data for some loci try this out. For example, for a motif CAGG, you might need to use CCTG.

`calibrationFiles`
A tab delimited file with as many columns as loci amplified in the samples. The header names must start with the locus name, written exactly has in the file passed to locusInfo. The first row contains the range of alleles for each locus, the format should be 100-250, for an example of an allelic range between 100 and 250 bp. Users should inspect the plots created by `reads.plots` to determine the ranges. Be conservative and add a buffer of about 10 bp on each side but know that the larger the range the slower the function will be. The following rows contain sample names (one per row) for a homozygote samples that will be used to generate the expected relative frequencies under different combinations of alleles. Users can select these homozygote representative samples by inspecting the plots saved in the Plots directory (generated by `reads.plots`). Use the exact sample names as the file names stored in directory samplesF. For convenience, sample names are written to the plots and can be copied from there when selecting homozygote samples. I used six samples per locus in my test (thus my calibration table had a total of seven rows).

Details

Using a set of representative homozygote samples and an allelic range provided by the user, the function produces a large table with as many rows as combinations of diploid genotypes, i.e., all combinations of 2 in N, N being defined by seq(lower.range, upper.range, 1). To these combination of 2 in N the homozygote cases, for all alleles in the defined range, are added as additional possible genotypes. Then a pattern of relative frequencies from a single allele, generated from the analysis of reads in the homozygote reference samples, is shifted along the allelic range to match the bp position of each different combination of alleles. This procedure generates all the possible overlaps of read relative frequencies for the heterozygote cases. For heterozygotes the relative frequencies are then divided by two so that the row sum adds to 1. The table rows (one for each possible genotype) are then compared against the observed data for each sample by function `alleleFinder` to find the row table that minimizes a sum of squares relative to the observed relative frequencies for each read in each sample: sum((ORF(a)-ERF(a))^2), with ORF and ERF the Observed and Expected relative frequencies at each read size a. The row with the minimum sum is the estimated genotype.

Value

One model table for each locus is written in a directory, named ModelTables, created by the function in the working directory.

Author(s)

F. Alberto

References

Alberto F (in prep)...

See Also

alleleFinder

`getpoints` *Interacts with an open plot to extract sample names*

Description

A tool to extract sample name information

Usage

```
getpoints(GenotypeA.DF,locus)
```

Arguments

`GenotypeA.DF`	A data.frame as produced by `alleleFinder`
`locus`	A string with the name of the locus to plot. The name needs to be written as in the file provided to locusInfo argument to other functions, e.g., `read.size.freqs`

Details

Interacts with the plots produced by `AlleleCum` to extract sample name information for specific allele sizes. Upon running the command the R locator function is called for plot interaction. The user needs to perform two clicks on the plot, first by moving the cursor below the targeted allele points, the second click above them. The list of samples in GenotypeA.DF that contain alleles sizes within the y range (size in bp) of the two clicks is printed to the screen.

Value

The sample names containing the selected allele sizes are printed to screen.

Author(s)

F. Alberto

References

Alberto F (in prep)...

See Also

AlleleCum

`M.allele.Finder` *Identifies multiple alleles (possibly more than two) for each loci in each sample.*

Description

The function searches samples for multiple alleles of the same loci. Examples of such samples come from forensic analysis, where DNA from multiple individuals might be present, or eDNA samples containing alleles from several individuals. Only alleles present in the reference population can be detected.

Usage

```
M.allele.Finder(sampleDir="./samplesF.MA", locusInfo="primers",
                calibrationFiles="CalibrationSamples.txt",coverageF=20,
                ReadSizesRepeatNumbers="./ReadSizesRepeatNumbers/",
                RefGenotypes="./GenotypesOut/Genotypes allele table.txt")
```

Arguments

`sampleDir`	The directory where the filtered reads are stored. This directory should only contain the read files and nothing else. By default this is a directory named samplesF written in the session working directory, as produced by `read.quality.filter`
`locusInfo`	A tab delimited text file containing the information for the microsatellite loci used. Should contain four columns with the following header names: Locus: The loci names Fprimer: The sequence of the FWD primer in upper case Rprimer: The sequence of the RVS primer in upper case Motif: The sequence for the microsatellite motif (e.g., CTTG for tetranucleotide). Note that for some loci you might need to use the reverse complementar sequence of what your reference value is. Thus, inspect your reads first or if you are not getting data for some loci try this out. For example, for a motif CAGG, you might need to use CCTG.

`calibrationFiles`
: A tab delimited file with as many columns as loci amplified in the samples. The header names must start with the locus name, written exactly has in the file passed to locusInfo. The first row contains the range of alleles for each locus, the format should be 100-250, for an example of an allelic range between 100 and 250 bp. Users should inspect the plots created by `reads.plots` to determine the ranges. Be conservative and add a buffer of about 10 bp on each side but know that the larger the range the slower the function will be. The following rows contain sample names (one per row) for a homozygote samples that will be used to generate the expected relative frequencies under different combinations of alleles. Users can select these homozygote representative samples by inspecting the plots saved in the Plots directory (generated by `reads.plots`). Use the exact sample names as the file names stored in directory samplesF. For convenience, sample names are written to the plots and can be copied from there when selecting homozygote samples. I used six samples per locus in my test (thus my calibration table had a total of seven rows).

`coverageF`
: Reads scored as alleles based on the sum of squares minimization method need to have coverage larger than this value, otherwise are written as missing data NA. Defaults to coverage of 20. It is possible to send a vector with different values for each locus. You can input a single value if you want to use the same coverage threshold filter for all loci .

`ReadSizesRepeatNumbers`
: The path to the folder with the same name created by function `read.size.freqs`.

`RefGenotypes`
: A string with the path for the reference file genotype file in the 2 columns per locus format. The format is the same as produced by function `alleleFinder` and thus the default path is the folder where that file is written by this function "./GenotypesOut/Genotypes allele table.txt". Only alleles present in the reference population can be found.

Details

The function searches for the presence of alleles, from those present in the reference population, in samples that may have up to n alleles. Examples of such samples come from forensic analysis where DNA from multiple multiple individuals might be present or eDNA samples containing alleles from several individuals.

Value

The function returns to the working environment a list with the alleles found for each locus and individual. The function also prints to the working directory a pdf file for each sample containing plots of read (allele) distribution, with separate plots per page showing the different loci, and red vertical lines indicating the alleles scored.

Author(s)

F. Alberto

References

Alberto F (in prep)...

See Also

GenotypeModel, alleleFinder

`M.allele.seq.variant` *Identifies multiple sequence based alleles in eDNA or forensic samples.*

Description

The function searches samples for multiple sequenced based alleles for alleles of the same size in each loci. Examples of such samples come from forensic analysis, where DNA from multiple individuals might be present, or eDNA samples containing alleles from several individuals. Sequenced based AllelesSample are searched from a list containing the multiple size-based alleles found by `M.allele.Finder`.

Usage

```
M.allele.seq.variant(sampleDir="./samplesF.MA",AlleleList,locusInfo,alleleDB, nmotif=2,
                     HeteroThreshold=0.4,coverageF=25)
```

Arguments

`sampleDir`	The directory where the filtered reads are stored. This directory should only contain the read files and nothing else. By default this is a directory named samplesF written in the session working directory, as produced by `read.quality.filter`
`AlleleList`	A list with the alleles found for each loci and sample as created by `M.allele.Finder`.
`locusInfo`	A tab delimited text file containing the information for the microsatellite loci used. Should contain four columns with the following header names: Locus: The loci names Fprimer: The sequence of the FWD primer in upper case Rprimer: The sequence of the RVS primer in upper case Motif: The sequence for the microsatellite motif (e.g., CTTG for tetranucleotide). Note that for some loci you might need to use the reverse complementar sequence of what your reference value is. Thus, inspect your reads first or if you are not getting data for some loci try this out. For example, for a motif CAGG, you might need to use CCTG.

`alleleDB` A data.frame containing the database of allele size, sequence variation, new allele code, and nucleotide sequence, as created by `score.seq.variants`. If necessary, the function will update the database with new allele sizes or sequence variants or both, in which case the updated database is written to the working directory. The format is a tab-delimited table with different sequence variants (rows) and the following required columns:

Locus: The locus name for the sequence-based allele. Needs to match the loci names supplied in argument locusInfo.

SIZE: The size in bp for the sequence-based allele.

VARIANT: A low case letter code to distinguish the different sequence-based alleles that share the same sequence size.

SEQ: The nucleotide sequence for each sequence-based allele

AlleleCode: A numeric code used to write alleles in format required by most population genetics software.

`nmotif` The number of microsatellite repeat motif repetitions to search for. It takes either a single value to be applied to all loci or a vector of values. We highly suggest you use the same values for this function as used for `read.size.freqs`.

`HeteroThreshold`
When sequence variation exists within reads of the same size for the same loci, two or more sequences will dominate the read frequencies, with many more in low frequency due to sequencing errors. This argument refers to the proportion of reads in the second and following most frequent variants compared to the first. The value sets the minimum allowed proportion required to score these reads as sequence-based variant alleles, i.e., not sequencing errors. It defaults to 0.4.

`coverageF` Reads scored as sequence-based alleles need to have coverage larger than this value. Defaults to coverage of 25. It is possible to send a vector with different values for each locus. You can input a single value if you want to use the same coverage threshold filter for all loci. We highly recommend using values similar to what was used in `alleleFinder` or `score.seq.variants`.

Details

The function searches for the presence of sequenced-based alleles in list containing multiple size-based alleles supplied by the user (as produced by `M.allele.Finder`). The function requires a data base of allele variants, as produced by `score.seq.variants`. If new variants are found, the allele database will be updated and written to working environment (Allele DataBase UPDATED.txt). A log file (DataBase new entries.txt) is also written to the working environment with the new alleles added to the database. The fifth column in this file informs if the new allele represents a new size or a new sequence variant.

Value

A list with the sequence-based alleles found for each locus and individual.

Author(s)

F. Alberto

References

Alberto F (in prep)...

See Also

score.seq.variants, alleleFinder

---

read.quality.filter *Filters the fastq reads for sequencing quality*

Description

Filters the reads so that if a single nucleotide is below a phred score of 30 the read is removed. Future versions will allow user defined phred scores.

Usage

```
read.quality.filter(sampleDir="./samples",filterDir="samplesF")
```

Arguments

sampleDir	A string for the path of the directory containing the samples. This should contain only the fastq files and nothing else. The default is a folder named samples
filterDir	A string with the directory path where the filtered fastq files will be stored. The default is a folder named samplesF. This folder is created in the session working directory when running the function. When running multiple times, users might want to rename this folder.

Details

A read is filtered out if a single nucleotide is below a phred score 30. The input fastq files should be stored in a directory named samples, and the output filtered files will be written in a folder named samplesF created when the function is run. Users can change these names.

Value

Fstq read files filtered by sequence quality. A Filterlog file is written to the working directory with the number and percentage of reads that were removed for each sample.

Author(s)

F. Alberto

References

Alberto F (in prep)...

See Also

read.size.freqs

---

read.size.freqs *Finds reads that contain the microsatellite primers and four or more repeats of the microsatellite motif*

Description

Searches sequenced reads for those containing the microsatellite FWD and RVS primers and a user-defined number of microsatellite motif repeats. Writes files for each sample and loci combination containing the read size and the number of msat motif repeats for each read detected.

Usage

```
read.size.freqs(locusInfo="primers",sampleDir="./samplesF",nmotif=2)
```

Arguments

locusInfo	A tab delimited text file containing the information for the microsatellite loci used. Should contain four columns with the following header names: Locus: The loci names Fprimer: The sequence of the FWD primer in upper case Rprimer: The sequence of the RVS primer in upper case Motif: The sequence for the microsatellite motif (e.g., CTTG for tetranucleotide). Note that for some loci you might need to use the reverse complementar sequence of what your reference value is. Thus, inspect your reads first or if you are not getting data for some loci try this out. For example, for a motif CAGG, you might need to use CCTG.
sampleDir	The directory where the filtered reads are stored. This directory should only contain the read files and nothing else. By default this is a directory named samplesF written in the session working directory, as produced by read.quality.filter
nmotif	Reads are retained if they have at least these many repetitions of the repeat motif. It takes either a single value to be applied to all loci or a vector of values

Details

Searches sequenced reads for those containing the microsatellite FWD and RVS primers and a user-defined number of microsatellite motif repeats. Writes files for each sample and loci combination containing the read size and the number of msat motif repeats for each read detected. Running this function is the slowest step in the pipeline, but the files only need to be created once per set of new samples.

Value

Files for each sample and locus combinations with the following two columns for each read detected:

ReadSize: The length in bp of each read

RepeatNumber: The number of microsatellite motifs for each read

These files are written to a folder named ReadSizesRepeatNumbers that is created by the function in the working environment and will be used down the pipeline by the functions `reads.plots`, `GenotypeModel`, and `alleleFinder`.

Author(s)

F. Alberto

References

Alberto F (in prep)...

See Also

`reads.plots`, `alleleFinder`

`reads.plots` *Creates frequency plots for the reads of specfic locus for each sample*

Description

Creates plots to visualize the read frequencies and help discerning the possible microsatellite alleles.

Usage

```
reads.plots(sampleDir="./samplesF",locusInfo="primers",x.range=c(90,350),
            x.label.cex=0.8,RepNumberPlot=FALSE )
```

Arguments

sampleDir	A string with the path for the directory where the filtered reads are stored. This directory should only contain the read files and nothing else. This should be the samplesF directory written in the session working directory, as produced by read.quality.filter.
locusInfo	A tab delimited text file containing the information for the microsatellite loci used. Should contain four columns with the following header names: Locus: The loci names Fprimer: The sequence of the FWD primer in upper case Rprimer: The sequence of the RVS primer in upper case Motif: The sequence for the microsatellite motif (e.g., CTTG for tetranucleotide). Note that for some loci you might need to use the reverse complementar sequence of what your reference value is. Thus, inspect your reads first or if you are not getting data for some loci try this out. For example, for a motif CAGG, you might need to use CCTG.
x.range	The x-axis limits plot limits. If some sample has alleles outside this range the function will adjust the x-axis limit for that sample. The defautl range is 90-350 bp.
x.label.cex	The plot size for allele labels (sizes in bp) along the x-axis. The default size is 0.8.
RepNumberPlot	A logical to decide if the second plot per sample should be plotted. It defaults to FALSE.

Details

The function provides a visualization tool that separates samples by loci, i.e. one plot per loci, with all samples and two pages per sample. The first sample plot is the classical barplot with x-axis for read sizes and y-axis for their frequency. The second plot shows read size vs number of motif repeats for reads. The plots are based on information from reads that contain the locus primers and a certain number of repeats for the msat motif in each locus. The function reads the files created by function read.size.freqs written to the folder ReadSizesRepeatNumbers.

Value

Plots for each loci, each containing two pages per sample. The first plot is the classical barplot with x-axis for read sizes and y-axis for their frequency. The second plot shows read size vs number of motif repeats for reads containing both primers and a minimum number of motif repeats (as set by function read.size.freqs).

Author(s)

F. Alberto

References

Alberto F (in prep)...

See Also

read.size.freqs

`sampleDist` *Plots barplots with filtered read frequency (coverage) for a selected sample and given locus.*

Description

Plots coverage barplots of a selected sample for a given locus

Usage

```
sampleDist(sample,xlim=c(75,400))
```

Arguments

`sample`	The name of the sample to plot including the name of the loci. The format is exactly as shown in the main tittle of the plots saved in the Plots directory, thus copy them from here.
`xlim`	A vector with two values for lower and upper limits of the x-axis

Details

A plotting function to zoom in to specific sample locus filtered read plots. Produces a plot similar to those printed to pdfs in the Plots directory for a single sample.

Value

A barplot with the read size coverage of a selected sample for a given locus.

Author(s)

F. Alberto

References

Alberto F (in prep)...

See Also

reads.plots

`score.seq.variants` *Scores alleles based on sequence polymorphism*

Description

The function searches for sequence based polymorphism in a data set of size-only scored alleles.

Usage

```
score.seq.variants(locusInfo="primers",RefGenotypes,sampleDir="./samplesF",
              nmotif=2,HeteroThreshold=0.4,alleleDB=NULL,coverageF=25)
```

Arguments

`locusInfo`	A tab delimited text file containing the information for the microsatellite loci used. Should contain four columns with the following header names: Locus: The loci names Fprimer: The sequence of the FWD primer in upper case Rprimer: The sequence of the RVS primer in upper case Motif: The sequence for the microsatellite motif (e.g., CTTG for tetranucleotide). Note that for some loci you might need to use the reverse complementar sequence of what your reference value is. Thus, inspect your reads first or if you are not getting data for some loci try this out. For example, for a motif CAGG, you might need to use CCTG.
`RefGenotypes`	A tab delimited file with two columns per locus containing the size only based alleles to be inspected for sequence based variation. The format is of file "/GenotypesOut/Genotypes allele table.txt" created by function `alleleFinder`.
`sampleDir`	The directory where the filtered reads are stored. This directory should only contain the read files and nothing else. By default this is a directory named samplesF written in the session working directory, as produced by `read.quality.filter`
`nmotif`	Reads are retained if they have at least these many repetitions of the repeat motif. It takes either a single value to be applied to all loci or a vector of values
`HeteroThreshold`	When sequence variation exists within reads of the same size for the same loci, two variants will dominate the read frequencies, with many more in low frequency due to sequencing errors. This argument refers to the proportion of reads in the second most frequent variant compared to the first. The value sets the minimum allowed proportion required to score the second allele as a sequence-based variant allele, i.e., not a sequencing error. It defaults to 0.4.
`alleleDB`	A database with allele size, sequence variation, new allele code, and nucleotide sequence. The default is NULL, in which case the function will produce the

database from the data supplied. When a database is supplied, the function will update it with any new variants that might be identified. The format is a tab-delimited table with different sequence variants (rows) and the following required columns:

Locus: The locus name for the sequence-based allele. Needs to match the loci names supplied in argument locusInfo.

SIZE: The size in bp for the sequence-based allele.

VARIANT: A low case letter code to distinguish the different sequence-based alleles that share the same sequence size.

SEQ: The nucleotide sequence for each sequence-based allele

AlleleCode: A numeric code used to write alleles in format required by most population genetics software.

`coverageF` Reads scored as sequence-based alleles need to have coverage above this frequency otherwise are written as missing data NA. Defaults to coverage of 25.

Details

The function searches for sequence-based polymorphism within a size-only genotype table supplied by the user (refGenotypes argument to function). The filtered reads are reanalyzed and searched for sequence-based polymorphism. The analysis of sequence-based variation within loci reads of the same size reveals many variants in low frequency that represent sequencing errors. Two high-frequency reads, which would otherwise be cryptic (by homoplasy) alleles, are often revealed. The function is intended for diploid data, and therefore, if three or more high-frequency sequence-based alleles are identified, the genotype is scored as NA, and a message is logged to a WarningLog file written to the working directory.

Value

Outputs three files written in a folder created in the working directory named AlleleBySeq. The three files are Allele DataBase.txt, containing the reference database of alllele sequence variants found; Genotypes.by.Sequence.txt, a population genetics file with two columns per locus and alleles represented by the concatenation of allele size and variant letter code; and Genotype Codes.txt contains new numeric allele codes, one column per locus, that are required for most population genetics software. The allele database file contains the correspondence between the different allele code formats. A warningLog file is also written directly to the working directory reporting the cases where more than two alleles were found, in which case NAs were written to the output file.

Author(s)

F. Alberto

References

Alberto F (in prep)...

See Also

GenotypeModel, alleleFinder

Index