Package ‘STRMPS’

July 2, 2018

Type Package

Title Analysis of Short Tandem Repeat (STR) Massively Parallel Sequencing (MPS) Data

Version 0.5.8

Date 2018-07-01

Encoding UTF-8

Author Søren B. Vilsen

Maintainer Søren B. Vilsen <svilsen@math.aau.dk>

Description Loading, identifying, aggregating, manipulating, and analysing short tandem repeat regions of massively parallel sequencing data in forensic genetics. 'STRMPS' can work with the package 'STRaitRazoR' (an R interface to the 'STRaitRazor' commandline tool) for added speed. 'STRaitRazoR' only works on Linux and can be found at <https://github.com/svilsen/STRaitRazoR>. The analyses and framework implemented in this package relies on the papers of Vilsen et al. (2017) <doi:10.1016/j.fsigen.2017.01.017> and Vilsen et al. (2018) <doi:10.1016/j.fsigen.2018.04.003> and parallelisation in the package relies on mclapply() and, thus, speed-ups will only be seen on UNIX based systems.

License MIT + file LICENSE

LazyData TRUE

Depends R (>= 3.1.0), Biostrings, ShortRead

Imports methods, utils, IRanges, tidyr, tibble, dplyr, stringr, purrr, parallel

Suggests STRaitRazoR

biocViews Biostrings, ShortRead, IRanges

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2018-07-02 08:30:06 UTC
## R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLMM</td>
<td>3</td>
</tr>
<tr>
<td>extractedReadsList-class</td>
<td>4</td>
</tr>
<tr>
<td>extractedReadsListCombined-class</td>
<td>4</td>
</tr>
<tr>
<td>extractedReadsListNonCombined-class</td>
<td>4</td>
</tr>
<tr>
<td>extractedReadsListReverseComplement-class</td>
<td>4</td>
</tr>
<tr>
<td>findNeighbours</td>
<td>5</td>
</tr>
<tr>
<td>findNeighbours,stringCoverageGenotypeList-method</td>
<td>5</td>
</tr>
<tr>
<td>findStutter</td>
<td>6</td>
</tr>
<tr>
<td>findStutter,stringCoverageGenotypeList-method</td>
<td>7</td>
</tr>
<tr>
<td>flankingRegions</td>
<td>7</td>
</tr>
<tr>
<td>genotypeIdentifiedList-class</td>
<td>8</td>
</tr>
<tr>
<td>genotypeList</td>
<td>8</td>
</tr>
<tr>
<td>getGenotype</td>
<td>9</td>
</tr>
<tr>
<td>getGenotype,stringCoverageList-method</td>
<td>10</td>
</tr>
<tr>
<td>identifiedSTRs</td>
<td>11</td>
</tr>
<tr>
<td>identifyNoise</td>
<td>11</td>
</tr>
<tr>
<td>identifyNoise,stringCoverageList-method</td>
<td>12</td>
</tr>
<tr>
<td>identifySTRRegions</td>
<td>13</td>
</tr>
<tr>
<td>identifySTRRegions,character-method</td>
<td>14</td>
</tr>
<tr>
<td>identifySTRRegions,ShortReadQ-method</td>
<td>15</td>
</tr>
<tr>
<td>identifySTRRegions.control</td>
<td>16</td>
</tr>
<tr>
<td>mergeGenotypeStringCoverage</td>
<td>17</td>
</tr>
<tr>
<td>mergeGenotypeStringCoverage,genotypeIdentifiedList-method</td>
<td>18</td>
</tr>
<tr>
<td>mergeNoiseStringCoverage</td>
<td>19</td>
</tr>
<tr>
<td>mergeNoiseStringCoverage,noisIdentifiedList-method</td>
<td>20</td>
</tr>
<tr>
<td>neighbourList-class</td>
<td>21</td>
</tr>
<tr>
<td>noiseIdentifiedList-class</td>
<td>21</td>
</tr>
<tr>
<td>noiseList</td>
<td>21</td>
</tr>
<tr>
<td>phredQualityProbability</td>
<td>22</td>
</tr>
<tr>
<td>phredQualityScore</td>
<td>22</td>
</tr>
<tr>
<td>stringCoverage</td>
<td>23</td>
</tr>
<tr>
<td>stringCoverage,extractedReadsList-method</td>
<td>24</td>
</tr>
<tr>
<td>stringCoverage,extractedReadsListCombined-method</td>
<td>25</td>
</tr>
<tr>
<td>stringCoverage,extractedReadsListNonCombined-method</td>
<td>26</td>
</tr>
<tr>
<td>stringCoverage,extractedReadsListReverseComplement-method</td>
<td>27</td>
</tr>
<tr>
<td>stringCoverage.control</td>
<td>28</td>
</tr>
<tr>
<td>stringCoverageGenotypeList</td>
<td>29</td>
</tr>
<tr>
<td>stringCoverageGenotypeList-class</td>
<td>29</td>
</tr>
<tr>
<td>stringCoverageList</td>
<td>29</td>
</tr>
<tr>
<td>stringCoverageList-class</td>
<td>30</td>
</tr>
<tr>
<td>stringCoverageNoiseList-class</td>
<td>30</td>
</tr>
<tr>
<td>STRMPSWorkflow</td>
<td>30</td>
</tr>
<tr>
<td>STRMPSWorkflowBatch</td>
<td>31</td>
</tr>
<tr>
<td>STRMPSWorkflowCollectStutters</td>
<td>32</td>
</tr>
<tr>
<td>workflow.control</td>
<td>32</td>
</tr>
</tbody>
</table>

**Index**

34
**BLMM**

*Block length of the missing motif.*

**Description**

Given a motif length and a string it finds the blocks of the string.

**Usage**

```r
BLMM(s, motifLength = 4, returnType = "numeric")
```

**Arguments**

- `s`: a string of either class: 'character' or 'DNAString'.
- `motifLength`: the known motif length of the STR region.
- `returnType`: the type of return wanted. Takes three values 'numeric', 'string', or 'fullList' (or any other combination cased letters).

**Details**

If `returnType` is 'numeric', the function returns the numeric value of the LUS. If `returnType` is instead chosen as 'string', the function returns "[AATG]x" i.e. the motif, AATG, is repeated 'x' times. Lastly if the `returnType` is set to fullList, the function returns a list of data.frames containing every possible repeat structure their start and the numeric value of the repeat unit length.

**Value**

Depending on `returnType` it return an object of class 'numeric', 'string', or 'fulllist'.

**Examples**

```r
# Creating compound string 's'
stretch1 = paste0(rep("AATG", 10), collapse = "")
stretch2 = paste0(rep("ATCG", 4), collapse = "")

s = paste0(stretch1, stretch2)

# Return BLMM only
BLMM(s, motifLength = 4, returnType = "numeric")

# Return BLMM and motif of stretch
BLMM(s, motifLength = 4, returnType = "string")

# Return all blocks of 's'
BLMM(s, motifLength = 4, returnType = "fulllist")
```
**extractedReadsList-class**

*Extract STR region information*

**Description**

Identifies the marker of the read using flanking regions and trims the read to include what is between the flanking regions.

**extractedReadsListCombined-class**

*Combined extract STR region information.*

**Description**

Identifies the marker of the read for both the provided and reverse complement flanking regions. The resulting lists are then combined into a single list.

**extractedReadsListNonCombined-class**

*Combined extract STR region information.*

**Description**

Identifies the marker of the read for both the provided and reverse complement flanking regions.

**extractedReadsListReverseComplement-class**

*Extract STR region information of the reverse complement DNA strand.*

**Description**

Identifies the marker of the read using reverse complement flanking regions and trims the read to include what is between the flanking regions.
findNeighbours

Find neighbours

Description

Generic function for finding neighbouring strings, given identified alleles.

Usage

findNeighbours(stringCoverageGenotypeListObject, searchDirection, trace = FALSE)

Arguments

stringCoverageGenotypeListObject
A stringCoverageGenotypeList-class object.

searchDirection
The direction to search for neighbouring strings. Default is -1, indicating a search for '-1' stutters.

trace
Should a trace be shown?

Value

A 'neighbourList' with the neighbouring strings, in the specified direction, for the identified allele regions.

findNeighbours,stringCoverageGenotypeList-method

Find neighbours

Description

Generic function for finding neighbouring strings, given identified alleles.

Usage

## S4 method for signature 'stringCoverageGenotypeList'
findNeighbours(stringCoverageGenotypeListObject, searchDirection = -1, trace = FALSE)
findStutter

Description

Given identified alleles it search for `-1` stutters of the alleles.

Usage

```r
findStutter(stringCoverageGenotypeListObject, trace = FALSE)
```

Arguments

- `stringCoverageGenotypeListObject`: A `stringCoverageGenotypeList-class` object.
- `trace`: Should a trace be shown?

Value

A `neighbourList` with the stutter strings for the identified allele regions.

Examples

```r
# The object returned by merging a stringCoverageList-Object
# and a genotypeList-Object.
data("stringCoverageGenotypeList")

stutterList <- findStutter(stringCoverageGenotypeList)
stutterTibble <- subset(do.call("rbind", stutterList), !is.na(Genotype))

stutterTibble$BlockLengthMissingMotif
stutterTibble$NeighbourRatio
```
### findStutter,stringCoverageGenotypeList-method

**Find stutters**

#### Description
Given identified alleles it search for `-1` stutters of the alleles.

#### Usage
```r
## S4 method for signature 'stringCoverageGenotypeList'
findStutter(stringCoverageGenotypeListObject,
             trace = FALSE)
```

#### Arguments
- `stringCoverageGenotypeListObject`:
  A `stringCoverageGenotypeList-class` object.
- `trace`:
  Should a trace be shown?

#### Value
A `neighbourList` with the stutter strings for the identified allele regions.

#### Examples
```r
# The object returned by merging a stringCoverageList-Object
# and a genotypeList-Object.
data("stringCoverageGenotypeList")

stutterList <- findStutter(stringCoverageGenotypeList)
stutterTibble <- subset(do.call("rbind", stutterList), !is.na(Genotype))

stutterTibble$BlockLengthMissingMotif
stutterTibble$NeighbourRatio
```

---

### flankingRegions

**Flanking regions**

#### Description
The flanking regions searched for to identify the markers and STR regions of all autosomal/X/Y STR's in the Illumina ForenSeq prep kit.
Usage
data("flankingRegions")

Format

A tibble containing the flanks (forward and reverse), motif, motif length, adjustment need to make it compatible with CE, and the shifts needed for further trimming, for each marker

Author(s)

Søren B. Vilsen <svilsen@math.aau.dk>

---

genotypeIdentifiedList-class

Genotype list

Description

A reduced stringCoverageList restricted to the identified genotypes.

---

genotypeList Genotype list

Description

The identified genotypes of the stringCoverageList data, created by the getGenotype function.

Usage

data("genotypeList")

Format

A list of tibble’s one for each of the 10 markers, showing which strings are the potential alleles based on the 'Coverage'.

Author(s)

Søren B. Vilsen <svilsen@math.aau.dk>
getGenotype

Assigns genotype.

Description

getGenotype takes a stringCoverageList-object, assumes the sample is a reference file and assigns a genotype, based on a heterozygote threshold, for every marker in the provided list.

Usage

getGenotype(stringCoverageListObject, colBelief = "Coverage", 
thresholdSignal = 0, thresholdHeterozygosity = 0.35, 
thresholdAbsoluteLowerLimit = 1)

Arguments

stringCoverageListObject
    an stringCoverageList-object, created using the stringCoverage-function.

colBelief
    the name of the column used for identification.

thresholdSignal
    threshold applied to the signal (generally the coverage) of every string.

thresholdHeterozygosity
    threshold used to determine whether a marker is hetero- or homozygous.

thresholdAbsoluteLowerLimit
    a lower limit on the coverage for it to be called as an allele.

Value

Returns a list, with an element for every marker in stringCoverageList-object, each element contains the genotype for a given marker.

Examples

# Strings aggregated by 'stringCoverage()'
data("stringCoverageList")

getGenotype(stringCoverageList)
getGenotype,stringCoverageList-method

**Assigns genotype.**

**Description**

getGenotype takes an stringCoverageList-object, assumes the sample is a reference file and assigns a genotype, based on a heterozygote threshold, for every marker in the provided list.

**Usage**

```r
## S4 method for signature 'stringCoverageList'
getGenotype(stringCoverageListObject, 
  colBelief = "Coverage", thresholdSignal = 0, 
  thresholdHeterozygosity = 0.35, thresholdAbsoluteLowerLimit = 1)
```

**Arguments**

- `stringCoverageListObject` 
an stringCoverageList-object, created using the `stringCoverage`-function.
- `colBelief` 
  the name of the column used for identification.
- `thresholdSignal` 
  threshold applied to the signal (generally the coverage) of every string.
- `thresholdHeterozygosity` 
  threshold used to determine whether a marker is hetero- or homozygous.
- `thresholdAbsoluteLowerLimit` 
  a lower limit on the coverage for it to be called as an allele.

**Value**

Returns a list, with an element for every marker in stringCoverageList-object, each element contains the genotype for a given marker.

**Examples**

```r
# Strings aggregated by 'stringCoverage()' 
data("stringCoverageList")
getGenotype(stringCoverageList)
```
identifiedSTRs

<table>
<thead>
<tr>
<th>identifiedSTRs</th>
<th>Identified STR regions</th>
</tr>
</thead>
</table>

**Description**

The identified STR regions of the sampleSequences.fastq file, created by the `identifySTRRegions` function.

**Usage**

```
data("identifiedSTRs")
```

**Format**

A list with an element for each of the 10 identified markers indicating which sequences were identified for each marker.

**Author(s)**

Søren B. Vilsen <svilsen@math.aau.dk>

**identifyNoise**

Identifies the noise.

**Description**

`identifyNoise` takes a `stringCoverageList`-object and identifies the noise based on a signal threshold for every marker in the provided list.

**Usage**

```
identifyNoise(stringCoverageListObject, colBelief = "Coverage", thresholdSignal = 0.01)
```

**Arguments**

- `stringCoverageListObject`: an `stringCoverageList`-object, created using the `stringCoverage`-function.
- `colBelief`: the name of the column used for identification.
- `thresholdSignal`: threshold applied to the signal (generally the coverage) of every string.

**Value**

Returns a list, with an element for every marker in `stringCoverageListObject`, each element contains the genotype for a given marker.
identifyNoise,stringCoverageList-method

Identifies the noise.

Description

identifyNoise takes an stringCoverageList-object and identifies the noise based on a signal threshold for every marker in the provided list.

Usage

```r
## S4 method for signature 'stringCoverageList'
identifyNoise(stringCoverageListObject, 
  colBelief = "Coverage", thresholdSignal = 0.01)
```

Arguments

- `stringCoverageListObject`: an stringCoverageList-object, created using the `stringCoverage`-function.
- `colBelief`: the name of the column used for identification.
- `thresholdSignal`: threshold applied to the signal (generally the coverage) of every string.

Value

Returns a list, with an element for every marker in stringCoverageList-object, each element contains the genotype for a given marker.

Examples

```r
# Strings aggregated by 'stringCoverage()' 
data("stringCoverageList")

identifyNoise(stringCoverageList, thresholdSignal = 0.03)
```
identifySTRRegions

Identify the STR regions of a fastq-file or ShortReadQ-object.

Description

identifySTRRegions takes a fastq-file location or a ShortReadQ-object and identifies the STR regions based on a directly adjacent flanking regions. The function allows for mutation in the flanking regions through the numberOfMutation argument.

Usage

identifySTRRegions(reads, flankingRegions, numberOfMutation, control)

Arguments

- reads: either a fastq-file location or a ShortReadQ-object
- flankingRegions: containing marker ID/name, the directly adjacent forward and reverse flanking regions, used for identification.
- numberOfMutation: the maximum number of mutations (base-calling errors) allowed during flanking region identification.
- control: an identifySTRRegions.control-object.

Value

The returned object is a list of lists. If the reverse complement strings are not included or if the control$combineLists == TRUE, a list, contains lists of untrimmed and trimmed strings for each row in flankingRegions. If control$combineLists == FALSE, the function returns a list of two such lists, one for forward strings and one for the reverse complement strings.

Examples

library("Biostrings")
library("ShortRead")

# Path to file
readPath <- system.file('extdata', "sampleSequences.fastq", package = 'STRMPS')

# Flanking regions
data("flankingRegions")

# Read the file into memory
readFile <- readFastq(readPath)
sread(readFile)
quality(readFile)

# Identify the STR's of the file, both readPath and readFile can be used.
identifySTRRegions(reads = readFile, flankingRegions = flankingRegions,
   numberOfMutation = 1,
   control = identifySTRRegions.control(
       numberOfThreads = 1,
       includeReverseComplement = FALSE)
)

identifySTRRegions, character-method
Identify the STR regions of a fastq-file or ShortReadQ-object.

Description
identifySTRRegions takes a fastq-file location or a ShortReadQ-object and identifies the STR regions based on a directly adjacent flanking regions. The function allows for mutation in the flanking regions through the numberOfMutation argument.

Usage
## S4 method for signature 'character'
identifySTRRegions(reads, flankingRegions,
   numberOfMutation = 1, control = identifySTRRegions.control())

Arguments
readspath to fastq-file.
flankingRegionscontaining marker ID/name, the directly adjacent forward and reverse flanking regions, used for identification.
numberOfMutationthe maximum number of mutations (base-calling errors) allowed during flanking region identification.
controlan identifySTRRegions.control-object.

Value
The returned object is a list of lists. If the reverse complement strings are not included or if the control$combineLists == TRUE, a list contains lists of untrimmed and trimmed strings for each row in flankingRegions. If control$combineLists == FALSE, the function returns a list of two such lists, one for forward strings and one for the reverse complement strings.
Examples

```r
library("Biostrings")
library("ShortRead")

# Path to file
readPath <- system.file('extdata', "sampleSequences.fastq", package = 'STRMPs')

# Flanking regions
data("flankingRegions")

# Read the file into memory
readFile <- readFastq(readPath)
sread(readFile)
quality(readFile)

# Identify the STR's of the file, both readPath and readFile can be used.

identifySTRRegions(reads = readFile, flankingRegions = flankingRegions,
                   numberOfMutation = 1,
                   control = identifySTRRegions.control(
                     numberOfThreads = 1,
                     includeReverseComplement = FALSE)
)
```

**identifySTRRegions, ShortReadQ-method**

*Identify the STR regions of a fastq-file or ShortReadQ-object.*

Description

identifySTRRegions takes a fastq-file location or a ShortReadQ-object and identifies the STR regions based on a directly adjacent flanking regions. The function allows for mutation in the flanking regions through the numberOfMutation argument.

Usage

```r
## S4 method for signature 'ShortReadQ'
identifySTRRegions(reads, flankingRegions,
                   numberOfMutation = 1, control = identifySTRRegions.control())
```

Arguments

- `reads` a ShortReadQ-object
- `flankingRegions` containing marker ID/name, the directly adjacent forward and reverse flanking regions, used for identification.
identifySTRRegions.control

numberOfMutation
default number of mutations (base-calling errors) allowed during flanking region identification.

control
an identifySTRRegions.control-object.

Value
The returned object is a list of lists. If the reverse complement strings are not included or if the control$combineLists == TRUE, a list contains lists of untrimmed and trimmed strings for each row in flankingRegions. If control$combineLists == FALSE, the function returns a list of two such lists, one for forward strings and one for the reverse complement strings.

Examples

library("Biostrings")
library("ShortRead")

# Path to file
readPath <- system.file('extdata', "sampleSequences.fastq", package = 'STRMPS')

# Flanking regions
data("flankingRegions")

# Read the file into memory
readFile <- readFastq(readPath)
sread(readFile)
quality(readFile)

# Identify the STR's of the file, both readPath and readFile can be used.
identifySTRRegions(reads = readFile, flankingRegions = flankingRegions,
    numberOfMutation = 1,
    control = identifySTRRegions.control(
        numberOfThreads = 1,
        includeReverseComplement = FALSE)
)

identifySTRRegions.control
Control function for identifySTRRegions

Description
A list containing default parameters passed to the identifySTRRegions function.
mergeGenotypeStringCoverage

Usage

identifySTRRegions.control(collist = NULL, numberOfThreads = 4L,
  reversed = TRUE, includeReverseComplement = TRUE, combineLists = TRUE,
  removeEmptyMarkers = TRUE, matchPatternMethod = "mclapply")

Arguments

collist The position of the forward, reverse, and motifLength columns in the flanking region tibble/data.frame. If 'NULL' a function searches for the words 'forward', 'reverse', and 'motif' to identify the columns.

numberOfThreads The number of threads used by mclapply (stuck at '2' on windows).

reversed TRUE/FALSE: In a reverse complementary run, should the strings/quality be reversed (recommended)?

includeReverseComplement TRUE/FALSE: Should the function also search for the reverse complement DNA strand (recommended)?

combineLists TRUE/FALSE: If 'includeReverseComplement' is TRUE, should the sets be combined?

removeEmptyMarkers TRUE/FALSE: Should markers returning no identified regions be removed?

matchPatternMethod Which method should be used to identify the flanking regions (only 'mclapply' implemented at the moment)?

Value

A control list setting default behaviour.

mergeGenotypeStringCoverage

Merge genotypeIdentifiedList and stringCoverageList.

Description

mergeGenotypeStringCoverage merges genotypeIdentifiedList-objects and stringCoverageList-objects.

Usage

mergeGenotypeStringCoverage(stringCoverageListObject,
  noiseGenotypeIdentifiedListObject)
Arguments

- `stringCoverageListObject` - a stringCoverageList-object, created using the `stringCoverage`-function.
- `noiseGenotypeIdentifiedListObject` - a noiseGenotypeIdentifiedList-object, created using the `getGenotype`-function.

Value

Returns a list, with an element for every marker in `extractedReadsList-object`, each element contains the string coverage of all unique strings of a given marker.

Examples

```r
# Strings aggregated by 'stringCoverage()'
data("stringCoverageList")
# Genotypes identified by 'getGenotype()'
data("genotypeList")
# Noise identified by 'identifyNoise()'
data("noiseList")

mergeGenotypeStringCoverage(stringCoverageList, genotypeList)
mergeNoiseStringCoverage(stringCoverageList, noiseList)
```

Description

`mergeGenotypeStringCoverage` merges `genotypeIdentifiedList-objects` and `stringCoverageList-objects`.

Usage

```r
## S4 method for signature 'genotypeIdentifiedList'
mergeGenotypeStringCoverage(stringCoverageListObject,
noiseGenotypeIdentifiedListObject)
```

Arguments

- `stringCoverageListObject` - a stringCoverageList-object, created using the `stringCoverage`-function.
- `noiseGenotypeIdentifiedListObject` - a noiseGenotypeIdentifiedList-object, created using the `getGenotype`-function.
mergeNoiseStringCoverage

Value

Returns a list, with an element for every marker in extractedReadsList-object, each element contains the string coverage of all unique strings of a given marker.

Examples

# Strings aggregated by 'stringCoverage()'
data("stringCoverageList")
# Genotypes identified by 'getGenotype()'
data("genotypeList")
# Noise identified by 'identifyNoise()'
data("noiseList")

mergeGenotypeStringCoverage(stringCoverageList, genotypeList)
mergeNoiseStringCoverage(stringCoverageList, noiseList)

mergeNoiseStringCoverage

Merge noiseIdentifiedList and stringCoverageList.

Description

mergeNoiseStringCoverage merges noiseIdentifiedList-objects and stringCoverageList-objects.

Usage

mergeNoiseStringCoverage(stringCoverageListObject, noiseGenotypeIdentifiedListObject)

Arguments

stringCoverageListObject
  a stringCoverageList-object, created using the stringCoverage-function.
noiseGenotypeIdentifiedListObject
  a noiseGenotypeIdentifiedList-object, created using the identifyNoise-function.

Value

Returns a list, with an element for every marker in extractedReadsList-object, each element contains the string coverage of all unique strings of a given marker.

Examples

# Strings aggregated by 'stringCoverage()'
data("stringCoverageList")
# Genotypes identified by 'getGenotype()'
data("genotypeList")
# Noise identified by 'identifyNoise()'
mergeNoiseStringCoverage,noiseIdentifiedList-method

Merge noiseIdentifiedList and stringCoverageList.

Description

mergeNoiseStringCoverage merges noiseIdentifiedList-objects and stringCoverageList-objects.

Usage

## S4 method for signature 'noiseIdentifiedList'
mergeNoiseStringCoverage(stringCoverageListObject, noiseGenotypeIdentifiedListObject)

Arguments

stringCoverageListObject
  a stringCoverageList-object, created using the stringCoverage-function.

noisgenotypeIdentifiedListObject
  a noiseGenotypeIdentifiedList-object, created using the identifyNoise-function.

Value

Returns a list, with an element for every marker in extractedReadsList-object, each element contains the string coverage of all unique strings of a given marker.

Examples

# Strings aggregated by 'stringCoverage()'  
data("stringCoverageList")
# Genotypes identified by 'getGenotype()'  
data("genotypeList")
# Noise identified by 'identifyNoise()'  
data("noiseList")

mergeGenotypeStringCoverage(stringCoverageList, genotypeList)  
mergeNoiseStringCoverage(stringCoverageList, noiseList)
neighbourList-class

Description

A list of the identified neighbours of the called alleles in a stringCoverageGenotypeList.

noiseIdentifiedList-class

Noise list

Description

Creates a flag to the sequences in a stringCoverageList which could be classified as noise.

noiseList

Noise list

Description

The identified noise of the stringCoverageList data, created by the identifyNoise function.

Usage

data("noiseList")

Format

A list of tibble's one for each of the 10 markers, showing which strings can be safely classified as noise based on the 'Coverage'.

Author(s)

Søren B. Vilsen <svilsen@math.aau.dk>
### phredQualityProbability

**Quality score to probability**

**Description**

Converts a quality score (Phred or Solexa) to a probability of error.

**Usage**

- `phredQualityProbability(q)`
- `solexaQualityProbability(q)`

**Arguments**

- `q` Quality score.

**Value**

`phredQualityScore(q_phred)` and `solexaQualityScore(q_solexa)` returns a probability of error.

**Examples**

```python
q_phred = phredQualityScore(1e-3)
q_solexa = solexaQualityScore(1e-3)

phredQualityProbability(q_phred)
solexaQualityProbability(q_solexa)
```

### phredQualityScore

**Convert probability to quality score**

**Description**

Calculates the quality score (Phred or Solexa) given a probability of error.

**Usage**

- `phredQualityScore(p)`
- `solexaQualityScore(p)`

**Arguments**

- `p` Probability of error.
stringCoverage

Value

phredQualityScore(p) returns a Phred quality score.
solexaQualityScore(p) returns a Solexa quality score.

Examples

p <- 1e-3
phredQualityScore(p)
ssolexaQualityScore(p)

stringCoverage  Get string coverage STR identified objects.

Description

stringCoverage takes an extractedReadsList-object and finds the coverage of every unique string
for every marker in the provided list.

Usage

stringCoverage(extractedReadsListObject, control = stringCoverage.control())

Arguments

extractedReadsListObject
an extractedReadsList-object, created using the identifySTRRegions-function.
control an stringCoverage.control-object.

Value

Returns a list, with an element for every marker in extractedReadsList-object, each element contains
the string coverage of all unique strings of a given marker.

Examples

# Regions identified using 'identifySTRs()'
data("identifiedSTRs")

# Limiting and restructuring
sortedIncludedMarkers <- sapply(names(identifiedSTRs$identifiedMarkers$Sequences$UniquelyAssigned),
                                  function(m) which(m == flankingRegions$Marker))

# Aggregate the strings
stringCoverage(extractedReadsListObject = identifiedSTRs,
               control = stringCoverage.control(
                motifLength = flankingRegions$MotifLength[sortedIncludedMarkers],
                Type = flankingRegions$Type[sortedIncludedMarkers],
                numberOfThreads = 1,
                trace = FALSE,
                simpleReturn = TRUE))
stringCoverage, extractedReadsList-method

Get string coverage STR identified objects.

Description

stringCoverage takes an extractedReadsList-object and finds the coverage of every unique string for every marker in the provided list.

Usage

```r
## S4 method for signature 'extractedReadsList'
stringCoverage(extractedReadsListObject, 
               control = stringCoverage.control())
```

Arguments

- `extractedReadsListObject`:
  an extractedReadsList-object, created using the `identifySTRRegions`-function.
- `control`:
  an `stringCoverage.control`-object.

Value

Returns a list, with an element for every marker in extractedReadsList-object, each element contains the string coverage of all unique strings of a given marker.

Examples

```r
# Regions identified using 'identifySTRs()' 
data("identifiedSTRs")

# Limiting and restructuring
sortedIncludedMarkers <- sapply(names(identifiedSTRs$identifiedMarkersSequencesUniquelyAssigned),
                                 function(m) which(m == flankingRegions$Marker))

# Aggregate the strings
stringCoverage(extractedReadsListObject = identifiedSTRs, 
               control = stringCoverage.control( 
                          motifLength = flankingRegions$MotifLength[sortedIncludedMarkers], 
                          Type = flankingRegions$Type[sortedIncludedMarkers], 
                          numberOfThreads = 1, 
                          trace = FALSE, 
                          simpleReturn = TRUE))
```
Get string coverage STR identified objects.

Description

stringCoverage takes an extractedReadsList-object and finds the coverage of every unique string for every marker in the provided list.

Usage

```r
## S4 method for signature 'extractedReadsListCombined'
stringCoverage(extractedReadsListObject, control = stringCoverage.control())
```

Arguments

- `extractedReadsListObject`: an extractedReadsList-object, created using the `identifySTRRegions`-function.
- `control`: an `stringCoverage.control`-object.

Value

Returns a list, with an element for every marker in extractedReadsList-object, each element contains the string coverage of all unique strings of a given marker.

Examples

```r
# Regions identified using 'identifySTRs()'
data("identifiedSTRs")

# Limiting and restructuring
sortedIncludedMarkers <- sapply(names(identifiedSTRs$identifiedMarkersSequencesUniquelyAssigned), function(m) which(m == flankingRegions$Marker))

# Aggregate the strings
stringCoverage(extractedReadsListObject = identifiedSTRs, control = stringCoverage.control(
    motifLength = flankingRegions$MotifLength[sortedIncludedMarkers],
    Type = flankingRegions$Type[sortedIncludedMarkers],
    numberOfThreads = 1,
    trace = FALSE,
    simpleReturn = TRUE))
```
stringCoverage, extractedReadsListNonCombined-method

Get string coverage STR identified objects.

Description

stringCoverage takes an extractedReadsList-object and finds the coverage of every unique string for every marker in the provided list.

Usage

```r
## S4 method for signature 'extractedReadsListNonCombined'
stringCoverage(extractedReadsListObject, 
               control = stringCoverage.control())
```

Arguments

- `extractedReadsListObject`:
an extractedReadsList-object, created using the `identifySTRRegions`-function.
- `control`:
an `stringCoverage.control`-object.

Value

Returns a list, with an element for every marker in extractedReadsList-object, each element contains the string coverage of all unique strings of a given marker.

Examples

```r
# Regions identified using 'identifySTRs()'
data("identifiedSTRs")

# Limiting and restructuring
sortedIncludedMarkers <- sapply(names(identifiedSTRs$identifiedMarkersSequencesUniquelyAssigned), 
                                function(n) which(n == flankingRegions$Marker))

# Aggregate the strings
stringCoverage(extractedReadsListObject = identifiedSTRs, 
               control = stringCoverage.control(
               motifLength = flankingRegions$MotifLength[sortedIncludedMarkers], 
               Type = flankingRegions$Type[sortedIncludedMarkers], 
               numberOfThreads = 1, 
               trace = FALSE, 
               simpleReturn = TRUE))
```
stringCoverage, extractedReadsListReverseComplement-method

Get string coverage STR identified objects.

Description

stringCoverage takes an extractedReadsList-object and finds the coverage of every unique string for every marker in the provided list.

Usage

```r
## S4 method for signature 'extractedReadsListReverseComplement'
stringCoverage(extractedReadsListObject, control = stringCoverage.control())
```

Arguments

- `extractedReadsListObject` an extractedReadsList-object, created using the `identifySTRRegions`-function.
- `control` an `stringCoverage.control`-object.

Value

Returns a list, with an element for every marker in extractedReadsList-object, each element contains the string coverage of all unique strings of a given marker.

Examples

```r
# Regions identified using 'identifySTRs()'
data("identifiedSTRs")

# Limiting and restructuring
sortedIncludedMarkers <- sapply(names(identifiedSTRs$identifiedMarkersSequencesUniquelyAssigned),
                                  function(m) which(m == flankingRegions$Marker))

# Aggregate the strings
stringCoverage(extractedReadsListObject = identifiedSTRs,
               control = stringCoverage.control(
                               motifLength = flankingRegions$MotifLength[sortedIncludedMarkers],
                               Type = flankingRegions$Type[sortedIncludedMarkers],
                               numberOfThreads = 1,
                               trace = FALSE,
                               simpleReturn = TRUE))
```
**Description**

String coverage control object

**Usage**

```r
stringCoverage.control(motifLength = 4, Type = "AUTOSOMAL", 
simpleReturn = TRUE, includeLUS = FALSE, numberOfThreads = 4L, 
meanFunction = mean, includeAverageBaseQuality = FALSE, trace = FALSE, 
uniquelyAssigned = TRUE)
```

**Arguments**

- **motifLength**: The motif lengths of each marker.
- **Type**: The chromosome type of each marker (autosomal, X, or Y).
- **simpleReturn**: TRUE/FALSE: Should the returned object be simplified?
- **includeLUS**: TRUE/FALSE: Should the LUS of each region be calculated?
- **numberOfThreads**: The number of cores used for parallelisation.
- **meanFunction**: The function used to average the base qualities.
- **includeAverageBaseQuality**: Should the average base quality of the region be included?
- **trace**: TRUE/FALSE: Show trace?
- **uniquelyAssigned**: TRUE/FALSE: Should regions not uniquely assigned be removed?

**Details**

Control function for the 'stringCoverage' function. Sets default values for the parameters.

**Value**

List of parameters used for the 'stringCoverage' function.
stringCoverageGenotypeList

Combined string coverage and genotype information

Description
A merge of the stringCoverageList and genotypeList data.

Usage
data("stringCoverageGenotypeList")

Format
A list of tibble’s one for each of the 10 markers containing the combined string coverage and genotypic information.

Author(s)
Søren B. Vilsen <svilsen@math.aau.dk>

stringCoverageGenotypeList-class

Combined stringCoverage- and genotypeIdentifiedList

Description
Merges a stringCoverageList with a genotypeIdentifiedList.

stringCoverageList
Aggregated string coverage.

Description
The aggregated string coverage of the identifiedSTRs data, created by the stringCoverage function.

Usage
data("stringCoverageList")

Format
A list of tibble’s one for each of the 10 markers, showing the aggregated information on a string-by-string basis.
**Author(s)**

Søren B. Vilsen <svilsen@math.aau.dk>

---

**stringCoverageList-class**

A string coverage list

**Description**

A list of tibbles, one for every marker, used to contain the sequencing information of STR MPS data. The tibbles should include columns with the following names: "Marker", "BasePairs", "Allele", "Type", "MotifLength", "ForwardFlank", "Region", "ReverseFlank", "Coverage", "AggregateQuality", and "Quality".

---

**stringCoverageNoiseList-class**

Combined stringCoverage- and noiseIdentifiedList

**Description**

Merges a stringCoverageList with a noiseIdentifiedList

---

**strmpsworkflow**

*Workflow function*

**Description**

The function takes an input file and performs all preliminary analyses. The function creates a series of objects which can be further analysed. An output folder can be provided to store the objects as .RData-files.

**Usage**

```r
strmpsworkflow(input, output = NULL, continueCheckpoint = NULL,
               control = workflow.control())
```

**Arguments**

- **input**: A path to a .fastq-file.
- **output**: A directory where output-files are stored.
- **continueCheckpoint**: Choose a checkpoint to continue from in the workflow. If NULL the function will run the entire workflow.
- **control**: Function controlling non-crucial parameters and other control functions.
Value

If 'output' not provided the function simply returns the stringCoverageList-object. If an output is provided the function will store ALL created objects at the output-path, i.e. nothing is returned.

Examples

readPath <- system.file('extdata', 'sampleSequences.fastq', package = 'STRMPS')
STRMPSWorkflow(readPath, 
  control = workflow.control(restrictType = "Autosomal",
    numberOfThreads = 1)
)

Description

The function takes an input directory and performs the entire analysis workflow described in (ADD REF). The function creates a series of objects needed for further analyses and stores them at the output location.

Usage

STRMPSWorkflowBatch(input, output, ignorePattern = NULL, 
  continueCheckpoint = NULL, control = workflow.control())

Arguments

input A directory where fastq input-files are stored.
output A directory where output-files are stored.
ignorePattern A pattern parsed to grepl used to filter input strings.
continueCheckpoint Choose a checkpoint to continue from in the workflow. If NULL the function will run the entire workflow.
control Function controlling non-crucial parameters and other control functions.

Value

If 'output' not provided the function simply returns the stringCoverageList-object. If an output is provided the function will store ALL created objects at the output-path, i.e. nothing is returned.
### STRMPSWorkflowCollectStutters

**Collect stutters files**

**Description**
Collects all stutter files created by the batch version of the STRMPSWorkflow function.

**Usage**
```r
STRMPSWorkflowCollectStutters(stutterDirectory, storeCollection = TRUE)
```

**Arguments**
- **stutterDirectory**: The out most directory containing all stutter files to be collected.
- **storeCollection**: TRUE/FALSE: Should the collected tibble be stored? If 'FALSE' the tibble is returned.

**Value**
If 'storeCollection' is TRUE nothing is returned, else the stutter collection is returned.

---

### workflow.control

**Workflow default options**

**Description**
Control object for workflow function returning a list of default parameter options.

**Usage**
```r
workflow.control(numberOfMutations = 1, numberOfThreads = 4, createdThresholdSignal = 0.05, thresholdHomozygote = 0.4, internalTrace = FALSE, simpleReturn = TRUE, identifyNoise = FALSE, identifyStutter = FALSE, flankingRegions = NULL, useSTRaitRazor = FALSE, trimRegions = TRUE, restrictType = NULL, trace = TRUE, variantDatabase = NULL, reduceSize = FALSE)
```
**workflow.control**

**Arguments**

- **numberOfMutations**: The maximum number of mutations (base-calling errors) allowed during flanking region identification.

- **numberOfThreads**: The number of threads used by either the `mclapply`-function (stuck at ’2’ on windows) or STRaitRazor.

- **createdThresholdSignal**: Noise threshold.

- **thresholdHomozygote**: Homozygote threshold for genotype identification.

- **internalTrace**: Show trace.

- **simpleReturn**: TRUE/FALSE: Should the regions be aggregated without including flanking regions?

- **identifyNoise**: TRUE/FALSE: Should noise be identified.

- **identifyStutter**: TRUE/FALSE: Should stutters be identified.

- **flankingRegions**: The flanking regions used to identify the STR regions. If ’NULL’ a default set is loaded and used.

- **useSTRaitRazor**: TRUE/FALSE: Should the STRaitRazor command line tool (only linux is implemented) be used for flanking region identification.

- **trimRegions**: TRUE/FALSE: Should the identified regions be further trimmed.

- **restrictType**: A character vector specifying the marker ’Types’ to be identified.

- **trace**: TRUE/FALSE: Should a trace be shown?

- **variantDatabase**: A tibble of 'trusted' STR regions.

- **reduceSize**: TRUE/FALSE: Should the size of the data-set be reduced using the quality and the variant database?

**Value**

List of default of options.
Index

**Topic data**
- flankingRegions, 7
- genotypeList, 8
- identifiedSTRs, 11
- noiseList, 21
- stringCoverageGenotypeList, 29
- stringCoverageList, 29

BLMM, 3
- extractedReadsList-class, 4
- extractedReadsListCombined-class, 4
- extractedReadsListNonCombined-class, 4
- extractedReadsListReverseComplement-class, 4
- findNeighbours, 5
- findNeighbours, stringCoverageGenotypeList-method, 5
- findStutter, 6
- findStutter, stringCoverageGenotypeList-method, 7
- flankingRegions, 7
- genotypeIdentifiedList-class, 8
- genotypeList, 8, 29
- getGenotype, 8, 9, 18
- getGenotype, stringCoverageList-method, 10
- identifiedSTRs, 11, 29
- identifyNoise, 11, 19–21
- identifyNoise, stringCoverageList-method, 12
- identifySTRRegions, 11, 13, 16, 23–27
- identifySTRRegions, character-method, 14
- identifySTRRegions, ShortReadQ-method, 15
- identifySTRRegions.control, 13, 14, 16, 16

mclapply, 33
- mergeGenotypeStringCoverage, 17
- mergeGenotypeStringCoverage, genotypeIdentifiedList-method, 18
- mergeNoiseStringCoverage, 19
- mergeNoiseStringCoverage, noiseIdentifiedList-method, 20
- neighbourList-class, 21
- noiseIdentifiedList-class, 21
- noiseList, 21
- phredQualityProbability, 22
- phredQualityScore, 22
- solexaQualityProbability
  (phredQualityProbability), 22
- solexaQualityScore (phredQualityScore), 22
- stringCoverage, 9–12, 18–20, 23, 29
- stringCoverage, extractedReadsList-method, 24
- stringCoverage, extractedReadsListCombined-method, 25
- stringCoverage, extractedReadsListNonCombined-method, 26
- stringCoverage, extractedReadsListReverseComplement-method, 27
- stringCoverage.control, 23–27, 28
- stringCoverageGenotypeList, 29
- stringCoverageGenotypeList-class, 5–7, 29
- stringCoverageList, 8, 21, 29, 29
- stringCoverageList-class, 30
- stringCoverageNoiseList-class, 30
- STRMPSWorkflow, 30, 32
- STRMPSWorkflowBatch, 31
- STRMPSWorkflowCollectStutters, 32
- tibble, 8, 21, 29, 33

34
INDEX

workflow.control, 32