Package: Rabe (via r-universe)

February 13, 2025

Title Adenine base editor analysis

Version 0.0.1

Description Base editors are emerging molecular sensors for protein-RNA interaction. This package implements a workflow for analysis of adenine base editor datasets. With minimal adjustment it can be used for systematic inquiry of any known single base-pair mutagenesis patterns. Part of the y3628 analysis suite.

License GPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

biocViews CellBiology, Genetics

Imports stringr, tibble, dplyr, S4Vectors, IRanges, GenomicRanges, VariantAnnotation, SummarizedExperiment

Depends R (>= 4.1)

Suggests knitr, rmarkdown

VignetteBuilder knitr

Config/pak/sysreqs make libicu-dev libpng-dev libxml2-dev libssl-dev

Repository https://yeyuan98.r-universe.dev

RemoteUrl https://github.com/yeyuan98/Rabe

RemoteRef HEAD

RemoteSha 22e6ebb929919efdcdce8bac6c89baddbb3144dd

Contents

`assignToGRanges` *Assign variant IDs to a GRanges object based on overlap*

Description

Assign variant IDs to a GRanges object based on overlap

Usage

```
assignToGRanges(variant_ids, targetGRanges, prune = TRUE)
```

Arguments

`variant_ids`	Character vector of variant IDs
`targetGRanges`	GenomicRanges::GRanges object
`prune`	Boolean, if TRUE will only rturn rows that overlap to >=1 variants

Details

This function considers strandedness when finding overlap. "A/G" variants will only be assigned to "+" stranded ranges, and vice versa. However, you must make sure that only "A/G" and "T/C" variants are included in the input.

Value

GenomicRanges::GRanges with an additional mcol $variant_ids. $variant_ids is a IRanges::CharacterList

Examples

```
vignette("rABE-analysis")
```

assignVariantIds *Assign overlapping VCF variant IDs to a GRanges*

Description

Assign overlapping VCF variant IDs to a GRanges

Usage

```
assignVariantIds(range, vcf, sep = ";")
```

Arguments

range	GenomicRanges::GRanges object
vcf	VariantAnnotation::VCF object
sep	Separator to use if >1 variants overlap to a range

Value

Input range except adding a $variant_ids column

Examples

```
vignette("rABE-analysis")
```

diffVariants *Set difference of two variant tables*

Description

Set difference of two variant tables

Usage

```
diffVariants(varTableX, varTableY)
```

Arguments

varTableX	variant table X
varTableY	variant table Y

Value

variant table of X \ Y (based on $variant_id)

Examples

```
vignette("rABE-analysis")
```

filterAdenineEditVariants *Filter variants to include only Adenine base editing*

Description

Filter variants to include only Adenine base editing

Usage

```
filterAdenineEditVariants(assignedRange, stranded = TRUE, pruning = TRUE)
```

Arguments

assignedRange	GenomicRanges::GRanges with $variant_ids column. You can assign variants with the function `assignVariantIds()`.
stranded	Boolean whether to consider strandedness. If TRUE, A/G editing for '+' strand, T/C for '-' strand, raising error if any strand takes '*' value.
pruning	Whether to prune values in $variant_ids column. If TRUE, $variant_ids will only retain adenine base editing events. If FALSE, $variant_ids is not changed.

Value

GenomicRanges::GRanges containing ranges with at least one adenine base editing event.

Examples

```
vignette("rABE-analysis")
```

filterVariants *Filter a single merged variant table*

Description

Filter a single merged variant table

Usage

```
filterVariants(mergedVarTable, perc.limits, plot = TRUE)
```

Arguments

mergedVarTable	Single tibble::tibble of merged variants
perc.limits	Lower and upper limits of median mutation percentage.
plot	Boolean of whether to plot a simple histogram of mutation percentages

Value

Filtered merged variant table

Examples

```
vignette("rABE-analysis")
```

`getConfidentVariantIds` *Get 'confident' variants for downstream analysis*

Description

Get 'confident' variants for downstream analysis

Usage

```
getConfidentVariantIds(variantTableList, print.stats = TRUE, min.occurence = 2)
```

Arguments

`variantTableList`	A list of variant tables. Each variant table must have $variant_id column.
`print.stats`	Boolean, whether to print statistics.
`min.occurence`	Integer, minimum number of occurences for a variant to be considered as 'confident'.

Value

A character vector of 'confident' variant IDs.

Examples

```
vignette("rABE-analysis")
```

getNameVarTable *Convert GRanges with assigned variants into a tibble*

Description

Convert GRanges with assigned variants into a tibble

Usage

```
getNameVarTable(assignedGRanges)
```

Arguments

assignedGRanges	an "assigned GRanges" (i.e., GenomicRanges::GRanges with names and variant_ids).

Value

A tibble::tibble with columns:

- name, character of name(assignedGRanges)
- variant_ids, list(character) of assignedGRanges$variant_ids

Examples

```
vignette("rABE-analysis")
```

getPrimaryVariantTable *Get data frame of primary variants from a VCF file*

Description

Get data frame of primary variants from a VCF file

Usage

```
getPrimaryVariantTable(vcf, variant_ids)
```

Arguments

vcf	VariantAnnotation::VCF object
variant_ids	Identifiers to subset the VCF object

Value

A data.frame of counts for primary variants of each ID queried.

getVariantTable *Return detailed variant information (mutation rate, counts for REF/ALT).*

Description

Return detailed variant information (mutation rate, counts for REF/ALT).

Usage

```
getVariantTable(vcf, assignedRange)
```

Arguments

vcf	VariantAnnotation::VCF object
assignedRange	GenomicRanges::GRanges with $variant_ids column. You can assign variants with the function `assignVariantIds()`.

Value

A tibble::tibble

Examples

```
vignette("rABE-analysis")
```

mergeVariants *Merge a list of variant tables into one table.*

Description

Merge a list of variant tables into one table.

Usage

```
mergeVariants(variantTableList, variantIDsToKeep)
```

Arguments

variantTableList	list of variant tables.
variantIDsToKeep	which variant IDs to keep in the merged frame

Value

A single tibble::tibble of merged variant tables with following columns:

- variant_id, character vector
- variant_data list of tibble::tibble, each with columns $countREF, $countALT, $mutatin.perc.

Examples

```
vignette("rABE-analysis")
```

`prettyFormatSummary` *Format summary of a numeric vector*

Description

Format summary of a numeric vector

Usage

```
prettyFormatSummary(nums)
```

Arguments

`nums`	A numeric vector

Value

A character(1) showing standard summary statistics (i.e., 0/25/50/mean/75/100 quantiles).

`pruneAdenineEditVariants` *Prune a GenomicRanges containing variants to retain only A->G edits*

Description

Prune a GenomicRanges containing variants to retain only A->G edits

Usage

```
pruneAdenineEditVariants(rangeWithVariants, stranded = TRUE)
```

Arguments

`rangeWithVariants`	GenomicRanges::GRanges with $variant_ids. Each range can contain multiple variant_ids separated by ;. Example $variant_ids = "chr2L:1_A/G;chr2L:100_C/A"
`stranded`	Boolean, whether to consider strand of the ranges

Value

GenomicRanges::GRanges with only A->G edits

`variantID2GRanges` *Convert A/G variant ID vector into GenomicRanges::GRanges*

Description

Convert A/G variant ID vector into GenomicRanges::GRanges

Usage

```
variantID2GRanges(variant_ids)
```

Arguments

`variant_ids`	Character vector of VCF variant identifier. Example: "chr2L:1_A/G"

Details

Variant type is extracted but NOT checked at all. You must make sure that only "A/G" and "T/C" are present as input.

Value

GenomicRanges::GRanges object of the variants. All widths are one as this is record of single base mutations. The following mcols are defined:

- `type`, "A/G" (strand = "+") or "T/C" (strand = "-") to reflect variant type.
- `variant_id`, the original input as record

Index