Package 'UPDhmm'

October 24, 2025

Title Detecting Uniparental Disomy through NGS trio data

Version 1.5.1

BugReports https://github.com/martasevilla/UPDhmm/issues

Description Uniparental disomy (UPD) is a genetic condition where an individual inherits both copies of a chromosome or part of it from one parent, rather than one copy from each parent. This package contains a HMM for detecting UPDs through HTS (High Throughput Sequencing) data from trio assays. By analyzing the genotypes in the trio, the model infers a hidden state (normal, father isodisomy, mother isodisomy, father heterodisomy and mother heterodisomy).

biocViews Software, HiddenMarkovModel, Genetics

License MIT + file LICENSE

Encoding UTF-8

LazyData false

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

Imports HMM, utils, VariantAnnotation, GenomicRanges, S4Vectors, IRanges, SummarizedExperiment, Biobase, stats, BiocParallel, GenomeInfoDb

Suggests knitr, testthat (>= 2.1.0), BiocStyle, rmarkdown, markdown, karyoploteR, regioneR, dplyr, BiocManager

VignetteBuilder knitr

Roxygen list(markdown = TRUE)

URL https://github.com/martasevilla/UPDhmm

Language en-US

git_url https://git.bioconductor.org/packages/UPDhmm

git_branch devel

git_last_commit 34fc12a

git_last_commit_date 2025-10-15

Repository Bioconductor 3.23

Date/Publication 2025-10-24

Author Marta Sevilla [aut, cre] (ORCID:
<https://orcid.org/0009-0005-0179-920X>),
Carlos Ruiz-Arenas [aut] (ORCID:
<https://orcid.org/0000-0002-6014-3498>)

Maintainer Marta Sevilla <marta.sevilla@upf.edu>

Contents

`UPDhmm-package` *UPDhmm: Detecting Uniparental Disomy through NGS trio data*

Description

Uniparental disomy (UPD) is a genetic condition where an individual inherits both copies of a chromosome or part of it from one parent, rather than one copy from each parent. This package contains a HMM for detecting UPDs through HTS (High Throughput Sequencing) data from trio assays. By analyzing the genotypes in the trio, the model infers a hidden state (normal, father isodisomy, mother isodisomy, father heterodisomy and mother heterodisomy).

Author(s)

Maintainer: Marta Sevilla <marta.sevilla@upf.edu> (ORCID)

Authors:

- Carlos Ruiz-Arenas <cruizarenas@unav.es> (ORCID)

See Also

Useful links:

- https://github.com/martasevilla/UPDhmm
- Report bugs at https://github.com/martasevilla/UPDhmm/issues

---

`addOr` *Function to transform a large collapsed VCF into a dataframe, incorporating predicted states along with the log-likelihood ratio and p-value.*

Description

Function to transform a large collapsed VCF into a dataframe, incorporating predicted states along with the log-likelihood ratio and p-value.

Usage

```
addOr(filtered_def_blocks_states, largeCollapsedVcf, hmm, genotypes)
```

Arguments

`filtered_def_blocks_states`	data.frame object containing the blocks
`largeCollapsedVcf`	Input VCF file
`hmm`	Hidden Markov Model used to infer the events. The format should adhere to the general HMM format from HMM package with a series of elements: 1. The hidden states names in the "States" vector. 2. All possible observations in the "Symbols" vector. 3. Start probabilities of every hidden state in the "startProbs" vector. 4. Transition probabilities matrix of the hidden states in "transProbs". 5. Probabilities associated between every hidden state and all possible observations in the "emissionProbs" matrix.
`genotypes`	Possible GT formats and its correspondence with the hmm

Value

data.frame containing the transformed information.

addRatioDepth *Add depth ratios (inside vs outside events) per sample*

Description

This function computes the average depth inside each block and compares it to the average depth outside the block, generating a ratio for each sample (proband, mother, father).

Usage

```
addRatioDepth(filtered_def_blocks_states, largeCollapsedVcf, field = NULL)
```

Arguments

filtered_def_blocks_states	data.frame object containing the block
largeCollapsedVcf	Input VCF file
field	Optional character. Read Depth field to use (e.g. "DP", "DS"). If NULL, the function will try DP first, then AD as fallback.

Value

data.frame with added depth and inside/outside ratios per sample

applyViterbi *Apply the hidden Markov model using the Viterbi algorithm.*

Description

Apply the hidden Markov model using the Viterbi algorithm.

Usage

```
applyViterbi(largeCollapsedVcf, hmm, genotypes)
```

Arguments

largeCollapsedVcf	input vcf file
hmm	Hidden Markov Model used to infer the events
genotypes	Possible GT formats and its correspondence with the hmm

Value

largeCollapsedVcf

`asDfVcf` *Function to transform a large collapsed VCF into a dataframe with predicted states, including chromosome, start position, end position and metadata.*

Description

Function to transform a large collapsed VCF into a dataframe with predicted states, including chromosome, start position, end position and metadata.

Usage

```
asDfVcf(largeCollapsedVcf, genotypes)
```

Arguments

`largeCollapsedVcf`	Name of the large collapsed VCF file.
`genotypes`	Possible GT formats and its correspondence with the hmm

Value

dataframe

`blocksVcf` *Function to simplify contiguous variants with the same state into blocks.*

Description

Function to simplify contiguous variants with the same state into blocks.

Usage

```
blocksVcf(df)
```

Arguments

`df`	data.frame resulting from the `as_df_vcf` function.

Value

data.frame containing information on the chromosome, start #' position of the block, end position of the block, and predicted state.

`calculateEvents` *Calculate UPD events in trio VCFs.*

Description

This function predicts the hidden states by applying the Viterbi algorithm using the Hidden Markov Model (HMM) from the UPDhmm package. It takes the genotypes of the trio as input and includes a final step to simplify the results into blocks.

Usage

```
calculateEvents(
  largeCollapsedVcf,
  hmm = NULL,
  field_DP = NULL,
  BPPARAM = BiocParallel::SerialParam(),
  verbose = FALSE
)
```

Arguments

`largeCollapsedVcf`
: The VCF file in the general format (largeCollapsedVcf) with VariantAnnotation package. Previously edited with `vcfCheck()` function from UPDhmm package.

`hmm`
: Default = `NULL`. If no arguments are added, the package will use the default HMM already implemented, based on Mendelian inheritance. If an optional HMM is desired, it should adhere to the general HMM format from `HMM` package with the following elements inside a list:

  1. The hidden state names in the "States" vector.
  2. All possible observations in the "Symbols" vector.
  3. Start probabilities of every hidden state in the "startProbs" vector.
  4. Transition probabilities matrix between states in "transProbs".
  5. Probabilities associated between every hidden state and all possible observations in the "emissionProbs" matrix.

`field_DP`
: Default = `NULL`. Character string specifying which FORMAT field in the VCF contains the read depth information to use in `addRatioDepth()`. If `NULL` (default), the function will automatically try `"DP"` (standard depth) or `"AD"` (allelic depths, summed across alleles). Use this parameter if your VCF uses a non-standard field name for depth, e.g. `field = "NR"` or `"field_DP"`.

`BPPARAM`
: Parallelization settings, passed to bplapply. By default `BiocParallel::SerialParam()` (serial execution). To enable parallelization, provide a BiocParallel backend, e.g. `BiocParallel::MulticoreParam(workers = min(2, parallel::detectCores()))` or `BiocParallel::SnowParam(workers = 2)`. Note: when running under R CMD check or Bioconductor build systems, the number of workers may be automatically limited (usually less or equal to 2).

`verbose`
: Logical, default = `FALSE`. If `TRUE`, progress messages will be printed during processing.

Value

A `data.frame` object containing all detected events in the provided trio. If no events are found, the function will return an empty `data.frame`.

Examples

```
file <- system.file(package = "UPDhmm", "extdata", "test_het_mat.vcf.gz")
vcf <- VariantAnnotation::readVcf(file)
processedVcf <- vcfCheck(vcf,
    proband = "NA19675",
    mother = "NA19678",
    father = "NA19679"
)

# Run in serial mode (default)
res <- calculateEvents(processedVcf)

# Run in parallel with 2 cores
library(BiocParallel)
param <- MulticoreParam(workers = 2)
res_parallel <- calculateEvents(processedVcf, BPPARAM = param)
```

`collapseEvents` *Collapse events per sample and chromosome*

Description

This function collapses genomic events per individual, chromosome, and group, summarising the number of events, total Mendelian errors, the total span size, and a string listing all merged event coordinates.

Usage

```
collapseEvents(subset_df)
```

Arguments

`subset_df`	A data.frame containing event-level data with columns: ID: Sample identifier. seqnames: Chromosome name. start: Start position of the event. end: End position of the event. group: Event group/class. n_mendelian_error: Number of Mendelian errors in the event.

Value

A data.frame with collapsed events and columns: ID, seqnames, group n_events: Number of events collapsed total_mendelian_error: Sum of Mendelian errors across events total_size: Total genomic span size covered by events collapsed_events: Comma-separated list of collapsed events min_start, max_end: Genomic span of collapsed block

Examples

```
all_events <- data.frame(
ID = c("S1", "S1", "S1", "S2", "S2"),
seqnames = c("1", "1", "1", "2", "2"),
start = c(100, 150, 300, 500, 550),
end = c(120, 180, 320, 520, 580),
group = c("iso_mat", "iso_mat", "het_pat", "iso_mat", "iso_mat"),
n_mendelian_error = c(5, 10, 2, 50, 30),
stringsAsFactors = FALSE
)
out <- collapseEvents(all_events)
```

---

hmm *HMM data for predicting UPD events in trio genomic data*

Description

This dataset provides Hidden Markov Model (HMM) parameters for predicting uniparental disomy (UPD) events in trio genomic data.

states Five different possible states.

symbols Code symbols used for genotype combinations.

startProbs The initial probabilities of each state.

transProbs Probabilities of transitioning from one state to another.

emissionProbs Given a certain genotype combination, the odds of each possible state.

Usage

```
data(hmm)
```

Format

A list with 5 different elements

Source

Created in-house based on basic Mendelian rules for calculating UPD events.

Examples

```
data(hmm)
```

`identifyRecurrentRegions` *Identify recurrent genomic regions across samples*

Description

This function finds recurrent genomic regions across samples based on overlapping intervals. Regions exceeding the Mendelian error threshold are excluded.

Usage

```
identifyRecurrentRegions(
  df,
  ID_col = "ID",
  error_threshold = 100,
  min_support = 3
)
```

Arguments

`df`	A data.frame with columns: • ID: Sample identifier. • seqnames: Chromosome name. • start: Start coordinate of the region. • end: End coordinate of the region. • n_mendelian_error: Number of Mendelian errors in the region.
`ID_col`	Character string indicating the column name containing sample identifiers. Default is `"ID"`.
`error_threshold`	Numeric, default = 100. Maximum number of Mendelian errors allowed for a region to be considered.
`min_support`	Integer, default = 3. Minimum number of unique samples required to call a region recurrent.

Value

A GRanges object containing the recurrent regions that meet the minimum support threshold.

Examples

```
df <- data.frame(
ID = c("S1", "S2", "S3", "S3"),
seqnames = c("chr1", "chr1", "chr1", "chr1"),
start = c(100, 120, 500, 510),
end = c(150, 170, 550, 560),
n_mendelian_error = c(10, 20, 5, 5)
)
identifyRecurrentRegions(df, ID_col = "ID", error_threshold = 50, min_support = 2)
```

`markRecurrentRegions` *Annotate regions as recurrent or non-recurrent*

Description

Given a results data.frame and a set of recurrent genomic regions, this function labels each row as "Yes" (recurrent) or "No".

Usage

```
markRecurrentRegions(df, recurrent_regions)
```

Arguments

`df`	Data.frame with region coordinates and sample IDs.
`recurrent_regions`	GRanges object from identifyRecurrentRegions().

Value

The same data.frame with two added columns:

- Recurrent: "Yes" or "No"
- n_samples: Number of supporting samples (if recurrent)

Examples

```
input <- data.frame(
ID = c("S1", "S2", "S3", "S4"),
seqnames = c("chr1", "chr1", "chr1", "chr2"),
start = c(100, 120, 500, 100),
end = c(150, 170, 550, 150),
n_mendelian_error = c(10, 20, 5, 200)
)

recurrent_gr <- GenomicRanges::GRanges(
 seqnames = "chr1",
 ranges = IRanges::IRanges(
   start = 100,
   end = 170
 ),
 n_samples = 2
)
markRecurrentRegions(input, recurrent_gr)
```

processChromosome *Process a single chromosome for UPD detection*

Description

Internal helper function to run the full pipeline on one chromosome:

- applyViterbi
- asDfVcf
- blocksVcf

Usage

```
processChromosome(vcf_chr, hmm, genotypes)
```

Arguments

vcf_chr	CollapsedVCF object for one chromosome
hmm	Hidden Markov Model object
genotypes	Named vector mapping genotype strings to numeric states

Value

A data.frame of detected blocks for the chromosome, or NULL if error

vcfCheck *Check quality parameters (optional) and change IDs.*

Description

This function takes a VCF file and converts it into a largeCollapsedVcf object using the VariantAnnotation package. It also rename the sample for subsequent steps needed in UPDhmm package. Additionally, it features an optional parameter, quality_check, which triggers warnings when variants lack sufficient quality based on RD and GQ parameters in the input VCF.

Usage

```
vcfCheck(largeCollapsedVcf, father, mother, proband, check_quality = FALSE)
```

Arguments

largeCollapsedVcf	The file in largeCollapsedVcf format.
father	Name of the father's sample.
mother	Name of the mother's sample.
proband	Name of the proband's sample.
check_quality	Optional argument. TRUE/FALSE. If quality parameters want to be measured. Default = FALSE

Value

largeCollapsedVcf (VariantAnnotation VCF format).

Examples

```
fl <- system.file("extdata", "test_het_mat.vcf.gz", package = "UPDhmm")
vcf <- VariantAnnotation::readVcf(fl)
processedVcf <-
   vcfCheck(vcf, proband = "Sample1", mother = "Sample3", father = "Sample2")
```

Index