Package 'tidyGenR'

May 8, 2026

Title Tidy Multilocus Amplicon Genotypes

Version 0.1.7

Description Variant determination and genotyping from high throughput sequences from multilocus amplicon libraries, typically sequenced in Illumina MiSeq or similar. It provides a set of core functions for the central steps: demultiplex by locus, truncate reads, variant calling, and genotype calling. Additionally, it provides a set of functions for diagnosis and estimation of best running parameters and multiple extensions for genotype/variants manipulation and reformatting. Output variants and genotypes are output in 'tidy' format, thus facilitating reformatting, manipulation and potential connection to other R packages.

License GPL (>= 3)

URL https://github.com/csmiguel/tidyGenR

BugReports https://github.com/csmiguel/tidyGenR/issues

Depends R (>= 4.1.0)

Imports Biostrings, dada2 (>= 1.16), digest, dplyr (>= 1.0.0), ggplot2, glue, methods, plyr, readr, ShortRead (>= 1.32.0), stats, stringr, tibble, tidyr (>= 1.1.0), tidyselect, DECIPHER, patchwork, writexl

Suggests ape, knitr, rmarkdown, testthat (>= 3.0.0), tools, BiocStyle

biocViews Software, Sequencing, Classification, Phylogenetics

Encoding UTF-8

LazyData false

NeedsCompilation no

RoxygenNote 7.3.3

SystemRequirements cutadapt >=2.0 (https://cutadapt.readthedocs.io/en/stable/)

Config/testthat/edition 3

VignetteBuilder knitr

Author Miguel Camacho [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-6385-7963>), Jennifer Leonard [fnd]

Maintainer Miguel Camacho <miguelcamachosanchez@gmail.com>

Repository CRAN

Date/Publication 2026-02-17 22:10:02 UTC

Contents

add_allele_no *add allele_no to genotypes*

Description

add allele_no to genotypes

Usage

```
add_allele_no(gen)
```

Arguments

gen	genotypes

amplisas2tidy *Read AmpliSAT results into tidy variants*

Description

Reads AmpliSAT results as tidy variants.

Usage

```
amplisas2tidy(fp)
```

Arguments

fp	Character vector with paths to 'txt' files from AmpliSAT results. E.x. from 'allseqs', 'filtered', 'clustered'.

Details

Converts text files in folder with AmpliSAS (Sebastian et al. 2016) results into a tidy variants dataframe. MD5 is regenerated as the one in AmpliSAT does not coincide with the MD5 associated with the DNA sequence.

Value

Tidy variants (see 'variant_call()").

References

Sebastian et al. (2016). *AMPLISAS: a web server for multilocus genotyping using next-generation amplicon sequencing data*. Molecular Ecology Resources, 16(2), 498-510.

Examples

```
fp <- list.files(system.file("extdata", "amplisas", package = "tidyGenR"),
    full.names = TRUE
)
amplisas2tidy(fp)
```

`check1_lowcount` *Check1*

Description

All FASTQ have a number of reads above threshold.

Usage

```
check1_lowcount(list_fp, low_readcount = low_readcount)
```

Arguments

`list_fp`	List of length 1 or 2 character vectors with absolute F (and R) file paths.
`low_readcount`	'low_readcount' from 'check_raw_reads()'.

Value

Logical, TRUE if test is passed. Warning if FALSE.

`check2_unique_snames` *Check2*

Description

Sample names are unique.

Usage

```
check2_unique_snames(snames = snames)
```

Arguments

`snames`	List of length 1 or 2 character vectors with sample names.

Value

Logical, TRUE if test is passed. STOPs if FALSE.

check3_non_orphan *Check3*

Description

Not orphan F/R FASTQ files.

Usage

```
check3_non_orphan(list_fp, snames = snames)
```

Arguments

list_fp	List of length 1 or 2 character vectors with absolute F (and R) file paths.
snames	List of length 1 or 2 character vectors with sample names.

Value

Logical, TRUE if test is passed. Warning if FALSE. NULL for single end reads.

---

check_raw_reads *Check input raw FASTA/Q files*

Description

Check input raw FASTA/Q files

Usage

```
check_raw_reads(freads, rreads = NULL, low_readcount = 10)
```

Arguments

freads	Character vector with file paths to forward reads.
rreads	Character vector with file paths to reverse reads.
low_readcount	Numeric threshold to warn on number of reads in FASTA (DEFAULT = 10).

Details

Multiple checks on raw input FASTQ:

- Reads in FASTQ above threshold.
- Unique sample names.
- Not orphan F/R files for paired reads. For single-end data check 3 is excluded and NULL is returned in checks[4].

Value

List with 4 elements:

- 'f_reads', character vector with basename of forward reads.
- 'r_reads', character vector with basename of reads reads or NULL for single-end experiments.
- 'snames', character vector with sample names.
- 'checks', Logical vector with passed checks 1-3.

Examples

```
freads <-
list.files(system.file("extdata", "raw", package = "tidyGenR"),
pattern = "1.fastq.gz",
    full.names = TRUE)
check_raw_reads(freads)
```

---

compare_calls *Comparision between multiple tidy variants or genotypes.*

Description

Comparision between multiple tidy variants or genotypes.

Usage

```
compare_calls(td, dest_file = FALSE, creads = FALSE)
```

Arguments

td	List of named tidy dataframes with at least 'sample', 'locus', 'sequence'.
dest_file	Path to EXCEL file to write results. Default (FALSE), no file is written.
creads	Account for reads. dataframes in 'td' must have 'reads' column with the number of reads supporting each variant. If FALSE (default) distances between calls and plots are produced on the basis of presence (1)/absense (0) of the variant. If TRUE, read count are accounted for estimating distances and producing the plots. Default (FALSE).

Value

A list of 4 elements:

- *tidy_dat*: combined tidy number of reads for all elements in 'td'. If 'creads = FALSE', cells are coded as '1' (presence) and '0' (absence).
- *variants*: list with dataframes in wide format with all variants as columns. Variants are named 'locus_{first 6 characters of md5 of DNA seq}'. Zeroes '0' are included where no variant was found. If 'creads = FALSE', cells are coded as '1' (presence) and '0' (absence)

- *comp*: T/F matrix. samples x variants. TRUE, the cell*ij* across all matrices has the same value; FALSE, any of them has a different value.
- *dist*: pairwise distances between matrices (analogue to Manhattan distance).
- *plot1*: heatmap with frequency of calls (>1 == 1) across elements compared.
- *plot2*: heatmap with calls for all elements compared. Black tiles indicate calls absent in the element but present in another. element.
- *plot3*: boxplot of reads ordered by frecuency. (output only if 'creads' == TRUE).
- *plot4*: MDS plot (output only when length(td) > 2).
- (it writes an EXCEL file with the results to 'dest_file'.)

Examples

```
data("variant_calls")
compare_calls(variant_calls, creads = TRUE)
```

`cutadapt_command` *Create cutadapt command*

Description

Create cutadapt command

Usage

```
cutadapt_command(
  mode,
  interpreter,
  samples_file,
  cutadapt,
  extraArgs,
  overlap,
  e,
  g,
  G,
  o,
  p,
  readsfw,
  readsrv,
  log_out
)
```

Arguments

mode	"pe", paired-end; "se", single-end; or "linked" (beta, non-tested), for linked primers.
interpreter	Path to interpreter.
samples_file	Text file with sample names.
cutadapt	Path to cutadapt executable.
extraArgs	Other arguments for cutadapt (eg "–max-n 0 –max-expected-errors 3 –minimum-length=20").
overlap	–overlap (see cutadapt documentation).
e	-e (see cutadapt documentation).
g	Fasta file with forward primers.
G	Fasta file with reverse primers.
o	Path to demultiplexed forward reads.
p	Path to demultiplexed reverse reads.
readsfw	Path to input forward reads.
readsrv	Path to input reverse reads.
log_out	Path to write cutadapt log file.

Value

cutadapt command based on the 'mode" ('pe', 'se', 'linked').

---

demultiplex *Demultiplex reads by locus*

Description

Sequencing reads are demultiplexed by locus into separate FASTQ files (locus.sample.[1|2].fastq.gz) based on the locus-specific primer sequences using `cutadapt`.

Usage

```
demultiplex(
  interpreter = "/bin/bash",
  cutadapt = system2("which", "cutadapt", stdout = TRUE),
  freads,
  rreads = NULL,
  primers = NULL,
  sh_out = tempfile(fileext = ".sh"),
  outdir = tempdir(),
  log_out = tempfile(fileext = ".log"),
  temp_folder = tempdir(),
  mode = "pe",
```

```
  overlap = 15,
  e = 0.15,
  extraArgs = "",
  run = TRUE
)
```

Arguments

interpreter	Path to interpreter.
cutadapt	Path to cutadapt executable.
freads	Character vector with file paths to forward reads.
rreads	Character vector with file paths to reverse reads.
primers	Dataframe with primers
sh_out	File name for cutadapt command.
outdir	Directory to write demultiplexed FASTQ files. Created if it does not exist.
log_out	Path to write cutadapt log file.
temp_folder	Directory to save temp files.
mode	"pe", paired-end; "se", single-end; or "linked" (beta, non-tested), for linked primers.
overlap	–overlap (see cutadapt documentation).
e	-e (see cutadapt documentation).
extraArgs	Other arguments for cutadapt (eg "–max-n 0 –max-expected-errors 3 –minimum-length=20").
run	T/F, whether to run the script (T) or just write it (F).

Details

This function creates a bash script to run cutadapt. By turning on running 'run = T', the function will try to execute the bash script produced (written to 'sh_out'). If the function produces any errors it is recommended to turn off running 'run = FALSE' and inspect the bash script produced to detect mis-specified cutadapt arguments or erroneous paths, and to solve them using the cutadapt documentation.

Value

Demultiplexed 'sample.locus.[12].fastq.gz'.

Examples

```
data("primers")
freads <-
 list.files(system.file("extdata", "raw",
                        package = "tidyGenR"),
                        pattern = "1.fastq.gz",
            full.names = TRUE)
rreads <-
```

```
 list.files(system.file("extdata", "raw",
                        package = "tidyGenR"),
                        pattern = "2.fastq.gz",
            full.names = TRUE)
demultiplex(
    freads = freads,
    rreads = rreads,
    primers = primers,
    run = FALSE)
```

`dereplicate` *De-replicate reads into frequency tables for a set of FASTQ files*

Description

Reads FASTQ files and computes frequency of unique sequences per sample. Depth for unique sequences are organized in *samples* x *unique sequences* tables. Unique Ssequences are ordered in descending frequency.

Usage

```
dereplicate(
  fs,
  min_sam_fr = 2,
  min_loc_fr = 0.001,
  by = "_([a-zA-Z0-9]*_[F|R])",
  out_xlsx = NULL
)
```

Arguments

`fs`	Character vector with paths to all FASTQ to de-replicate.
`min_sam_fr`	Numeric. Minimum number of sequence counts in a sample to be retained (cell number).
`min_loc_fr`	Numeric. Minimum frequency of de-replicated sequence in the group to be retained. If $min\_loc\_fr \in (0, 1)$, then a proportion relative to the most frequent sequence is applied.
`by`	Regex pattern to group FASTQ files in the list. Passed to `stringr::str_extract()`.
`out_xlsx`	File name to write tables with de-replicated sequences (*Default: NULL; no file is written*).

Details

The *by* parameter allows flexible grouping of files in the list. However, the results are not added within each group; individual results for each sample are always returned. For instance, given 3 files s1_loc1_F.fq, s1_loc1_R.fq and s2_loc1_F.fq:

- `"([a-zA-Z0-9]*_[a-zA-Z0-9]*)"`, returns *s1_loc1* and *s2_loc2*.
- `"_([a-zA-Z0-9]*)_"`, returns *loc1* and *loc2*.
- `"([a-zA-Z0-9]*_[F|R])"`, returns *loc1_F*, *loc1_R* and *loc2_F*. The `min_sam_fr` and `min_loc_fr` filters drop data not passing the filters, so they will become zero or absent when combined. If a path to an EXCEL file is set, each element in the list is written to a different sheet in the workbook.

Value

List of extracted groups (see 'by'). Each element is the list is a dataframe with:

- column 1: 'sequence', DNA sequence of read.
- column 2: 'md5', md5 hash of DNA sequence.
- column >= 3: frequency (*integers*) of sequences per sample passing 'min_sam_fr' and 'min_loc_fr' filters.

Examples

```
fq <-
 list.files(system.file("extdata", "truncated",
                        package = "tidyGenR"),
                        pattern = "fastq.gz",
            full.names = TRUE)
dereplicate(fq)
```

---

dist_m *Distance between two matrices*

Description

Computed as the sum of absolute difference across cells. Analogue to Manhattan distance.

Usage

```
dist_m(m1, m2, prop = FALSE)
```

Arguments

`m1`	Matrix 1.
`m2`	Matrix 2.
`prop`	If TRUE, it divides number of differences by length(matrix).

Value

Numeric distance, analogue to Manhattan distance.

`dist_m_all` *Applies distance between all tables in a list.*

Description

Requires a list of matrices in which the first column are samples, and col2...coln are numeric. Ex. number of observed variants in each cell.

Usage

```
dist_m_all(ld)
```

Arguments

`ld`	List of dataframes.

Value

Dataframe with pairwise distances between all combinations. Absolute distance 'dist_euc' and relative 'dist_eucp'.

`explore_dada` *Explore variants output by DADA2 in the parameter space*

Description

DADA2 is run with a set of desired parameters for a set of FASTQ files. The element 'clustering' output by `dada2::dada()` containing all relevant statistics from cluster formation are output in tidy format as the first element. These tidy results are plotted in 4 different ways.

Usage

```
explore_dada(
  fs,
  sample_locus = "(^[a-zA-Z0-9]*)_([a-zA-Z0-9]*)",
  value_na = 10,
  reduced = TRUE,
  omega_a = 0.9,
  band_size = 16,
  pool = FALSE,
  vline = NULL,
  hline_fr = NULL,
  p_titles = NULL
)
```

Arguments

`fs`	Character vector with full paths to FASTQ files.
`sample_locus`	Regex expression with groups to extract sample (group 1) and loci (group 2) from "(^[a-zA-Z0-9]*)_([a-zA-Z0-9]*)".
`value_na`	Numeric to replace 'NA' or infinite values assigned to 'pval' or 'birth_pval' in clustering element from 'dada-class'.
`reduced`	If TRUE, a reduced number of columns is returned. If FALSE, all columns from from 'dada-class' clustering are returned.
`omega_a`	"OMEGA_A" passed to `dada2::dada()`.
`band_size`	"BAND_SIZE" passed to `dada2::dada()`.
`pool`	Passed to `dada2::dada()`.
`vline`	Numeric x-intersection to annotate in plots p1 and p3.
`hline_fr`	Numeric y-intersection to annotate 'frequency' in plots.
`p_titles`	character(4) with plot names. Passed to 'ggtitle()' in plots p1:p4.

Details

'OMEGA_A', 'BAND_SIZE' and 'pool = T/F' parameters can have a strong effect in the variants called by `dada2::dada()`. This function explores the relation of read count, relative frequency of the variants and the 'birth_pval' assigned to the clusters for any given values of starting parameters. Critical parameters 'pool' and 'BAND_SIZE' can be specified as arguments. Additional parameters can be set with `dada2::setDadaOpt()`. The 'log(-log(birth_pval))' proposed in Rosen et al. (2012) is computed and represented in plots. For represenation purposes, a virtually infinite value of 'log(-log(birth_pval)) = 10', is assigned by default to the first cluster and to 'birth_pval = 0'. Variants in each, sample/locus combination are ranked by abundance and plotted in the legend. Ranks >= 3 are named as "3". For biallelic markers, a rank >= 3 implies a likely false positive. This, visual representation can be used to decide to tune `variant_call()`.

- *omega_a*: threshold for variants to be significant overabundant 'log(-log(birth_pval))' (see Rosen et al. 2012). For exploration, it is recommended to run `explore_dada()` with a large `omega_a`.
- *band_size*: positive numbers set a band size in Needleman-Wunsch alignments. In this context, ends free alignment is performed. Zero turns off banding, triggering full Needleman-Wunsch alignments, in which gapless alignment is performed (see issue).
- *pool*: calling variants pooling samples can increase sensitivity (see dicussion).

Value

List with tidy 'dada-class' clustering element and plots.

1. tidy_dada: tidy 'clustering' element from 'dada-class' merged across loci.
2. p1: plot 1, frequency of variants (sample x locus) against 'log(-log(birth_pval))'.
3. p2: plot 2, read count of variants against their frequency.
4. p3: plot 3, p1 facetted by locus.
5. p4: plot 4, p2 facetted by locus.

References

Rosen et al. (2012). *Denoising PCR-amplified metagenome data*. BMC Bioinformatics, 13(1).

Examples

```
fq <-
 list.files(system.file("extdata", "truncated",
                        package = "tidyGenR"),
                        pattern = "F_filt.fastq.gz",
            full.names = TRUE)
explore_dada(fq,
    value_na = 10,
    reduced = TRUE,
    pool = FALSE,
    vline = 2,
    hline_fr = 0.1,
    omega_a = 0.9,
    band_size = 16
)
```

`filter_variants` *Filter variants based on frequency and depth*

Description

Filter variants based on frequency and depth

Usage

```
filter_variants(tidy_var, maf = 0, ad = 0, invert = FALSE)
```

Arguments

`tidy_var`	Dataframe or tibble with tidy variants.
`maf`	Miminum variant frequency. For each sample, variants with a proportion of number of reads across all variants > maf are retained.
`ad`	Allele depth. Minimum number of reads supporting a variant. Variants with ad > 3 are retained.
`invert`	FALSE (default) returns variants passing filters; TRUE inverts the selection.

Value

Tidy tibble with filtered variants.

dataframe with filtered variants/alleles. Variants are named so column names in df are "locus", "sample", "sequence", "reads", "allele".

Examples

```
data("variants")
filter_variants(variants, ad = 100, maf = 0.2)
```

---

genotype *Genotype samples from tidy variants*

Description

Performs straight genotyping from variants based on ploidy, ADt.

Usage

```
genotype(variants, ploidy = 2, ADt = 10)

geno_direct(variants, ploidy, ADt)
```

Arguments

variants	Tidy variants.
ploidy	'1', force haploid genotypes; '2', force diploid genotypes; 'poly', all variants are assumed to be real alleles.
ADt	Threshold of AD to discriminate hemi/homo-zygotes.

Details

For 'ploidy':

- '1', force haploid genotypes.
- '2', force diploid genotypes. If number of alleles is == 1, ADt criterium is applied to differentiate between "homozygotes" and "hemizygotes".
- 'poly', all variants are assumed to be real alleles. No ADt filter is applied.

Value

Tidy genotypes. Dataframe with tidy genotypes. 'sample, 'locus', 'allele', 'allele_no', 'reads', 'nt', 'md5', 'sequence'. *reads*, supporting allele copies are corrected: 'reads' from tidy variants are divided between resulting alleles in homozygotes and hemizygotes.

Examples

```
data("variants")
genotype(variants[1:100,],
    ADt = 10, ploidy = 2
)
```

`genotypes` *Genotypes*

Description

Filtered genotypes of 20 loci and 91 samples of *Rattus baluensis* used in Camacho-Sanchez et al. (2026).

Format

tidy dataframe or tibble.

sample Sample name.

locus Locus name.

allele Allele name.

allele_no Allele ordinal number.

reads Number of reads supporting allele.

nt Length in number of nucleotides of allele sequence.

md5 MD5 hash of allele sequence.

sequence DNA sequence of allele.

`genotype_conversion` *Conversion among genotype formats: tidy, compact, wide, STRUCTURE*

Description

Conversion of genotype data between formats.

- *gen_tidy2compact*, from *tidy* to *compact*.
- *gen_compact2wide*, from *compact* to *wide*.
- *gen_tidy2wide*, from *tidy* into *wide*.
- *gen_wide2structure*, from *wide* to STRUCTURE-formatted text file.
- *gen_tidy2genalex*, from *tidy* to GENALEX-formatted *xlsx* or text file.
- *gen_tidy2integers*, recode 'allele' in *tidy* genotypes as integers.

Usage

```
gen_tidy2compact(gen, delim = "/")

gen_compact2wide(gen)

gen_tidy2wide(gen)

gen_wide2structure(gen, write_out, popdata = FALSE, delim = "/")

gen_tidy2integers(gen)

gen_tidy2genalex(gen, popdata, write_out, data_name = "dataset1")
```

Arguments

gen	Genotypes: 'tidy' in 'gen_tidy2compact()', 'gen_tidy2wide()', 'gen_tidy2genalex()', and gen_tidy2integers()'; 'compact' in 'gen_compact2wide()'; 'wide' in 'gen_wide2structure()'.
delim	Allele delimiter in genotype calls. Default "/". E.x. "AA/BB".
write_out	File to write structure formatted genotypes (gen_wide2structure), or GENALEX-formatted genotypes (in 'gen_tidy2genalex'). In 'gen_tidy2genalex', genotypes are written to tab-delimited "txt" or "xlsx", depending on the extension ("txt", "xlsx").
popdata	Dataframe with 'sample' and 'population' columns. Populations are added to STRUCTURE output. If FALSE (Default), popdata is not added to STRUCTURE output. Mandatory in 'gen_tidy2genalex'.
data_name	Name of dataset to print to GENALEX xlsx.

Details

Genotypes from genotype() are returned as *tidy* data. Tidy data implies one data point per row. Each row from *tidy* genotypes represents an allele call for a given sample and locus. Rows with missing data are excluded. Thus, *tidy* genotypes contain at least the three columns: *sample*, *locus* and *allele*; bit often contain more columns (eg, *reads*, *allele_no*, *nt*, *sequence*, *md5*, *population*, etc.). The *tidy* format can be expanded column-wise and row-wise without altering the data structure, and it can be easily manipulated. Some of the more handy functions here-included involve genotype conversion between different formats:

- *compact*, each row correspond to the genotype call for a given locus in a sample. Therefore, it only contains three columns, *sample*, *locus* and *genotype*. Default separator for alleles under *genotype* is '/'. For diploid data, hemizygotes are recoded as "allele1" instead of "allele1/NA". All other column metadata is lost.
- *wide*, genotypes are recoded in samples (rows) x loci (columns) dataframe. The first column *samples*, contain sample names. All other columns contain loci names. Each cell contains genotypes in the *A/B* format. Diploid genotypes are coded as *A/B*. Cells with missing data have *NA_character*.

For the *STRUCTURE* format, '-9' are introduced as missing data in STRUCTURE output. In hemizygous calls *A*, the missing allele is encoded as missing data *-9*. Ploidy is retrieved from *attributes*. Only, diploid genotypes are allowed.

Value

- *gen_tidy2compact*, *compact* genotypes with at least 'sample' ,'locus' and 'genotype' columns.
- *gen_compact2wide*, *wide* genotypes.
- *gen_tidy2wide*, *wide* genotypes.
- *gen_wide2structure*, plain text file formatted for STRUCTURE.
- *gen_tidy2genalex*, 'xls' file with genotype data for GENALEX.
- *gen_tidy2integers*, tidy genotypes with alleles recoded as integers (1..n).

Examples

```
data("genotypes")
gencom <- gen_tidy2compact(genotypes)
gen_compact2wide(gencom)
gen_tidy2wide(genotypes)
gen_tidy2integers(genotypes)
# read metadata for sample populations
gen_str <- tempfile(fileext = ".str")
meta <-
    read.csv(system.file("extdata/metadata.csv", package = "tidyGenR"))
gen_wide2structure(gen_tidy2wide(genotypes),
    write_out = gen_str, popdata = meta
)
genalex_txt <- tempfile(fileext = ".txt")
gen_tidy2genalex(genotypes,
    popdata = meta,
    write_out = genalex_txt,
)
```

---

mds_comp *plot MDS*

Description

Plot MDS from distances between compared calls

Usage

```
mds_comp(df)
```

Arguments

df	Dataframe with pairwise distances from 'dist_m()'. One row per unique comparison.

out_popart *Format MSA and traits for PopArt*

Description

It takes a multiple sequence alignment and data to create a NEXUS file which can be read by PopART (https://popart.maths.otago.ac.nz).

Usage

```
out_popart(
  msa,
  x,
  blocks = c("DATA", "TRAITS"),
  sname,
  xgroups,
  outnex = tempfile(fileext = ".nex")
)
```

Arguments

msa	Multiple sequence alignment as DNAStringSet or DNAMultipleAlignment.
x	Dataframe with trait data.
blocks	NEXUS blocks to add to file. Either "DATA", "TRAITS" or c("DATA", "TRAITS").
sname	Column name in 'x' with sequence names which must be identical to sequence names in msa.
xgroups	Column name in x used for creating groups in TRAITS.
outnex	Path to write NEXUS file.

Value

No return value. Called for its side effect of writing a NEXUS file to outnex. Invisibly returns NULL.

Examples

```
data("genotypes")
x <-
    dplyr::filter(genotypes, locus == "abcg8")
nr <- seq_len(nrow(x))
y <-
    dplyr::mutate(x, sname = paste0("seq_", nr))
msa <-
    tidy2sequences(y, fasta_header = "{sname}") |>
    DECIPHER::AlignSeqs()
out_popart(msa, y,
    sname = "sname",
```

```
    xgroups = "sample",
    blocks = c("DATA", "TRAITS")
)
```

`primers` *Primers for multiplex PCR*

Description

A dataframe containing 27 primer pairs used to amplify intron loci in *Rattus baluensis* (Igea et al. 2010; Camacho-Sanchez et al. 2018).

Format

A data frame with 27 rows and 3 variables.

locus Locus name.

fw 5' to 3' forward primer sequence.

rv 5' to 3' reverse primer sequence.

Source

Igea et al. (2010); Camacho-Sanchez et al. (2018).

References

Camacho-Sanchez et al. (2018). *Interglacial refugia on tropical mountains: novel insights from the summit rat (Rattus baluensis), a Borneo mountain endemic*. Diversity and Distributions.

Igea et al. (2010) *Novel intron markers to study the phylogeny of closely related mammalian species*. BMC Evolutionary Biology.

`reads_loci_samples` *Output reads per locus*

Description

Creates dataframe with reads distribution across samples (rows) and loci (columns).

Usage

```
reads_loci_samples(
  path,
  pattern_fq = "1.fastq.gz",
  sample_locus = "(^[a-zA-Z0-9]*).([a-zA-Z0-9]*)",
  all.variants = FALSE,
  var_id = "md5"
)
```

Arguments

path	Folder with FASTQ files or tidy variants.
pattern_fq	Pattern to select FASTQ files from the folder
sample_locus	Sample and locus patterns to extract from file names. Regex group 1 is assumed to be sample name and group 2 locus name. By default, both groups are a alphanumeric string of any length separated by any non-alphanumeric character.
all.variants	(T/F) TRUE, one column per variant; F, one column per locus. Only for tidy variants.
var_id	'c('allele', 'md5', 'sequence')' column to use for naming variant.

Value

Dataframe with columns being loci and rows being samples. Samples are in first column.

Examples

```
reads_loci_samples(
     path = system.file("extdata", "truncated",
                        package = "tidyGenR"),
     pattern_fq = "F_filt.fastq.gz")
# with variants dataframe
data("variants")
reads_loci_samples(path = variants)
```

---

reads_track *Output reads across pipeline*

Description

It generates a dataframe with tracked reads per samples per locus. through the pipeline, from dataframes or FASTQ files.

Usage

```
reads_track(l, sample_pattern = "^[A-Za-z0-9]*")
```

Arguments

l	List of elements to retrieved reads from.
sample_pattern	Pattern used to extract sample names from FASTQ files.

Details

'l' is a named list. It can be either a dataframe with two mandatory fields ('sample' and 'reads'), or a character vector with 2 elements: 1, the path to a directory containing FASTQ files; and ,2, a pattern matching desired FASTQ files. By, default, sample names matching 'sample_pattern' '^[A-Za-z0-9]*' are extracted from FASTQ files. 'left_join()' is used iteratively from the first to the last element in 'l', so the elements in 'l' need to be nested from element 1 to the last.

Value

Dataframe with tracked reads through the pipeline. The first column is named 'sample' and the following

Examples

```
# from folder with FASTQ files
path2truncated <-
   system.file("extdata", "truncated",
               package = "tidyGenR")
l <- list(
      truncated = c(path2truncated, "F_filt.fastq.gz"))
reads_track(l)
```

---

remove_hemizygotes *Remove hemizygote calls*

Description

Removes all rows with hemizygous alleles from tidy genotypes.

Usage

```
remove_hemizygotes(gen)
```

Arguments

gen	Tidy diploid genotypes.

Details

If "A/B" -> "A/B"; if "A" -> NULL. It only is applicable to diploid data.

Value

Tidy genotypes without hemizygous alleles.

Examples

```
data("genotypes")
remove_hemizygotes(genotypes)
```

remove_monomorphic *Remove monorphic loci*

Description

Removes monomorphic (or polymorphic) loci from tidy genotypes.

Usage

```
remove_monomorphic(gen, invert = FALSE)
```

Arguments

gen	Tidy genotypes (output from 'genotype').
invert	T/F, if T, polymorphic loci are removed.

Value

Tidy genotypes with mono/poly-morphic loci.

Examples

```
data(genotypes)
remove_monomorphic(genotypes)
```

remove_poor_fastq *Remove empty FASTQ*

Description

Removes FASTA/FASTQ files with a number of reads less than or equal to `min_reads` from a specified directory.

Usage

```
remove_poor_fastq(path2fastq, pattern = "fastq.gz", min_reads = 0)
```

Arguments

path2fastq	Character. Path to the directory containing FASTQ files.
pattern	Character. Pattern used to identify FASTQ files in `path2fastq`.
min_reads	Integer. FASTQ files with a number of reads less than or equal to this threshold are removed.

Value

No return value. Called for its side effects: FASTA/FASTQ files in `path2fastq` with a number of reads less than or equal to `min_reads` are removed from disk. Invisibly returns `NULL`.

Examples

```
tmp <- tempdir()
fq <- file.path(tmp, "no_reads.fastq")
file.create(fq)
remove_poor_fastq(tmp, pattern = "fastq", min_reads = 0)
```

---

rename_alleles *Rename alleles in a dataframe based on reference alleles*

Description

Rename alleles in a dataframe based on reference alleles

Usage

```
rename_alleles(df_alleles, ref_alleles, replacements = FALSE)
```

Arguments

`df_alleles`	Dataframe with at least the following columns: 'allele' 'sequence'.
`ref_alleles`	Dataframe with at least the same columns as 'df_alleles', or path multifasta FASTA with allele names in fasta headers.
`replacements`	T/F, if TRUE, a dataframe with the replacements correspondence is returned. Optionally, a path to a multifasta with fasta headers being allele names.

Value

Dataframe as 'df_alleles' with renamed alleles as in 'ref_alleles'.

Examples

```
data("genotypes")
ref_al <-
    system.file("extdata/reference_alleles.fasta", package = "tidyGenR")
rename_alleles(df_alleles = genotypes, ref_alleles = ref_al)
```

rm_n_msa *Remove Ns introduced in 'mergePairs(justConcatenate = T)'*

Description

Detect N strings and remove them from aligned BStringSet MSA.

Usage

```
rm_n_msa(
  msa,
  pattern_n = paste0(rep("N", 10), collapse = ""),
  outfasta = FALSE
)
```

Arguments

msa	Multiple sequence alignment stored as a "DNAMultipleAlignment" or "DNAStringSet".
pattern_n	N string to match.
outfasta	Character vector with path to write FASTA file with MSA.

Details

When using 'mergePairs(justConcatenate = T)', a string of 10 Ns is introduced between paired-end concatenated reads. This function detects the coordinates of the N string in the MSA and removes it. It uses 'matchPattern()' to detect patterns. WARNING: it has not been tested for multiple hits of the pattern. It could potentially work with other patterns in addition to N strings, but it has not been tested.

Value

Filtered BStringSet MSA.

tidy2sequences *Output FASTA sequences*

Description

Write FASTA and/or return BioString object with selected sequences.

Usage

```
tidy2sequences(
  td,
  fasta_header = "{locus}_{allele}",
  filename = FALSE,
  seq = "sequence"
)
```

Arguments

td	Tidy variants or genotypes. Dataframe with at least one column with DNA sequences.
fasta_header	Naming of FASTA headers. The string is passed to 'glue' for forming the FASTA headers and selecting distinct rows.
filename	Path to write FASTA file, or FALSE, to prevent writing DNA sequences to a FASTA file.
seq	Name of the column in 'td' with DNA sequences.

Details

'fasta_header' is a flexible selector and constructor for FASTA headers. The variables included are used to filter distinct rows in 'td'. Examples of outputs:

- "{locus}_{allele}", all different allele sequences per locus.
- "{sample}*{locus}*{variant}", all sequences for all the variants for all samples, thus repeated sequences from the same variant corresponding to different samples.
- "{md5}", all different DNA sequences.
- "{sample}", one sequence per sample. Since one sample matches to many sequences, the first occurrence in the dataframe is selected.

Value

'DNAStringSet' object with selected sequences.

Examples

```
data("genotypes")
tidy2sequences(
    td = genotypes,
    fasta_header = "{locus}_{allele}",
    filename = FALSE
)
```

truncate *Truncate reads*

Description

Truncates reads using 'filterAndTrim()' from DADA2. It automatically detects single-end or paired-end sequencing files.

Usage

```
trunc_amp(
  loci = NULL,
  in_dir,
  trunc_fr,
  fw_pattern,
  rv_pattern = NULL,
  outdir = tempdir(),
  filt_name = "_F_filt.fastq.gz",
  mode_trun = "pe",
  multithread = FALSE,
  max_ee = 3,
  trunc_q = 2
)
```

Arguments

loci	Character vector with loci. If NULL, all loci detected by 'check_names_demultiplexed()' are processed.
in_dir	Path to folder with demultiplexed reads "sample.locus. [1|2].fastq.gz".
trunc_fr	Flexible argument that sets truncation length for F (and R) reads. • If 'numeric' of length == 1, trunFR is used for F and optionally R reads, trunc_f = trunc_r. • If 'numeric' of length == 2, the first and second positions are used as truncation lengths in F and R reads, accross all loci. • If it is a 'dataframe' with 'colnames == c("locus", "trunc_f", "trunc_r") ', locus-spefic truncation lengths are applied from 'trunc_f' and 'trunc_r' columns. If 'se', 'trunc_r' values are ignored.
fw_pattern	Pattern matching specific extension of forward reads.
rv_pattern	Pattern matching specific extension of reverse reads.
outdir	Path where filtered reads are written. Created if it does not exist.
filt_name	Pattern to append to FASTA names: '{sample}_{locus}{filt_name}'.
mode_trun	'se': single-end; 'pe', paired-end.
multithread	T/F, see 'filterAndTrim()'.
max_ee	Maximum expected errors. See 'filterAndTrim()'.
trunc_q	Trim low-quality bases from 3' end. See 'filterAndTrim()'.

Value

List with dataframes of number of reads before and after truncation. It writes truncated sequence to "outdir"

Examples

```
dem <-
 system.file("extdata", "demultiplexed", package = "tidyGenR")
# single end
trunc_amp(
    mode_trun = "pe",
    in_dir = dem,
    fw_pattern = "1.fastq.gz",
    rv_pattern = "2.fastq.gz",
    trunc_fr = c(250, 180),
    max_ee = c(3, 3)
)
# paired-end
data("trunc_fr")
trunc_amp(
    mode_trun = "se",
    loci = c("chrna9", "nfkbia"),
    in_dir = dem,
    fw_pattern = "1.fastq.gz",
    rv_pattern = "2.fastq.gz",
    trunc_fr = trunc_fr,
    max_ee = 3,
    trunc_q = 2
)
```

---

trunc_fr *Per-locus truncation lengths for forward and reverse reads.*

Description

Truncation lengths for forward and reverse reads for the dataset containing 30 primer pairs used to amplify intron loci in *Rattus baluensis* (Camacho-Sanchez et al. 2026).

Format

Data frame with three columns:

locus Locus name.

trunc_f Truncation length for forward reads.

trunc_r Truncation length for reverse reads.

variants *Variants*

Description

Filtered tidy variants of *Rattus baluensis* from Trusmadi (Camacho-Sanchez et al. 2026), corresponding to 27 loci from 44 samples.

Format

tidy dataframe or tibble.

sample Sample name.

locus Locus name.

variant Variant name.

reads Number of reads supporting allele.

nt Length in number of nucleotides of allele sequence.

md5 MD5 checksum of allele sequence.

sequence DNA sequence of allele.

variant_call *Variant calling*

Description

'variant_call' Call variants for multiple loci.

Usage

```
variant_call(
  loci = NULL,
  in_dir,
  fw_pattern = "_F_filt.fastq.gz",
  rv_pattern = NULL,
  sample_locus = "(^[a-zA-Z0-9]*).([a-zA-Z0-9]*)",
  c_unmerged = FALSE,
  pool = FALSE,
  error_function = loessErrfun,
  multithread = FALSE,
  chim_rm = "consensus",
  ad = 1,
  maf = 0,
  omega_a_f = getDadaOpt()$OMEGA_A,
  omega_a_r = getDadaOpt()$OMEGA_A,
  band_size = getDadaOpt()$BAND_SIZE
)
```

Arguments

loci	Character vector with loci to detect from 'sample_locus' in sample names. If NULL, all loci detected according to the pattern in the target directory are used.
in_dir	Path to folder with truncated files.
fw_pattern	Pattern matching files with F reads. It has to match everything after locus name. eg see parenthesis in "samplex_locus(_2_filt.fastq.gz)"
rv_pattern	Pattern matching files with R reads. If left NULL, single-end sequencing will be assumed.
sample_locus	Patterns to extract from FASTQ file names. Group 1 captures sample name and group 2 captures locus name. (DEFAULT: `(^[a-zA-Z0-9]*)_([a-zA-Z0-9]*)`). `^[a-zA-Z0-9]*_[a-zA-Z0-9]*` will extract 'sample_locus' from default naming convention `sample_locus_[F|R]_fastq.gz`.
c_unmerged	F/R sequences that were not merged in mergePairs are concatenated using a stretch of 10 N's.
pool	Passed to 'dada()'. Denoising is done in pooled samples (T) or by sample (F).
error_function	Use default 'loessErrfun' for regular Illumina quality codification and 'loess_err_mod4' for binned NovaSeq qualities.
multithread	T/F, passed to 'multithread' in 'dada' and 'learnErrors()'.
chim_rm	If FALSE, no chimera removal is performed. If == "character", it is passed to 'method' in 'removeBimeraDenovo()' (DEFAULT = 'consensus').
ad	Allele Depth. Passed to filter_variants.
maf	Minimum Allele Frequency. Passed to filter_variants.
omega_a_f	"OMEGA_A" passed to 'dada' in forward reads (Default: getDadaOpt()$OMEGA_A).
omega_a_r	"OMEGA_A" passed to 'dada' in reverse reads (Default: getDadaOpt()$OMEGA_A).
band_size	"BAND_SIZE" passed to 'dada' (DEFAULT: getDadaOpt()$BAND_SIZE).

Details

Allows single-end and paired-end data. Be careful with the use of 'c_unmerged'. It will trigger the 'justConcatenate' argument in 'mergePairs', and 10 N's will be used to concatenate non-overlapping F and R reads. Use 'c_unmerged' carefully, as it will generate artificial variant sequences. Default is deactivated. Variants are reformatted and to a "tibble" and filtered according to 'maf' and 'ad', which are added as attributes to output. Variables of output are 'sample', 'locus', 'sequence' (DNA sequence), 'variant' (name of variant), 'reads' (number of reads supporting that variant), 'nt' (sequence length), 'md5' (md5 checksum).

Value

tidy tibble with locus, sample, sequence, variant (name), nt(sequence length), md5.

Examples

```
# truncated fastq
truncated <-
 system.file("extdata", "truncated", package = "tidyGenR")
# variant calling
variant_call(in_dir = truncated)
```

---

variant_calling_dada *Variant calling using DADA2 ASV approach*

Description

- 'variant_call_dada()' Call variants for a single locus using DADA2. It is nested in 'variant_call'.
- 'loess_err_mod4()' Replaces default 'loessErrfun' when estimating error matrices for NovaSeq binned qualities.
- 'chimera_removal()' Wrapping function to remove chimeras using 'removeBimeraDenovo()'.

Usage

```
variant_call_dada(
  locus,
  in_dir,
  fw_pattern = "_F_filt.fastq.gz",
  rv_pattern = NULL,
  sample_locus = "(^[a-zA-Z0-9]*)_([a-zA-Z0-9]*)",
  c_unmerged = FALSE,
  pool = FALSE,
  error_function = loessErrfun,
  multithread = FALSE,
  chim_rm = "consensus",
  omega_a_f = getDadaOpt()$OMEGA_A,
  omega_a_r = getDadaOpt()$OMEGA_A,
  band_size = getDadaOpt()$BAND_SIZE
)

loess_err_mod4(trans)

chimera_removal(seqtab, chim_rm)

dada2list(dada_object, names)
```

Arguments

locus	Locus name.
in_dir	Path to folder with truncated files.
fw_pattern	Pattern matching files with F reads.
rv_pattern	Pattern matching files with R reads. If left NULL, single-end sequencing will be assumed.
sample_locus	Patterns to extract from FASTQ file names. Group 1 captures sample name and group 2 captures locus name. (DEFAULT: `(^[a-zA-Z0-9]*)_([a-zA-Z0-9]*)`). `^[a-zA-Z0-9]*_[a-zA-Z0-9]*` will extract 'sample_locus' from default naming convention `sample_locus_[F|R]_fastq.gz`.
c_unmerged	F/R sequences that were not merged in mergePairs are concatenated using a stretch of 10 N's.
pool	Passed to 'dada()'. Denoising is done in pooled samples (T) or by sample (F).
error_function	Use default 'loessErrfun()' for regular Illumina quality codification and 'loess_err_mod4()' for binned NovaSeq qualities. Passed to 'dada(errorEstimationFunction)'.
multithread	T/F, passed to 'dada(multithread)' and 'learnErrors(multithread)'.
chim_rm	If FALSE, no chimera removal is performed. If chim_rm is "character", it is passed to 'removeBimeraDenovo(method)'.
omega_a_f	"OMEGA_A" passed to 'dada' in forward reads (Default: 'getDadaOpt()$OMEGA_A').
omega_a_r	"OMEGA_A" passed to 'dada' in reverse reads (Default: 'getDadaOpt()$OMEGA_A').
band_size	"BAND_SIZE" passed to 'dada' (DEFAULT: 'getDadaOpt()$BAND_SIZE').
trans	See loessErrfun for details.
seqtab	Features table from dada2.
dada_object	Object from `dada2::dada()`, `dada2::mergePairs()`, etc.
names	Character vector to name the elements in the list. By default, DADA2 names are basenames from FAST(Q).

Details

- 'variant_call_dada()' Allows single-end and paired-end data. 'c_unmerged' will trigger the 'justConcatenate' argument in 'mergePairs', and 10 N's will be used to concatenate non-overlapping F and R reads. Use 'concateNotMerged' carefully, as it will generate artificial variant sequences. Default is deactivated.

- 'loess_err_mod4()' This is a beta function which has been shown to work in my experience and other users' experience. See discussion in https://github.com/benjjneb/dada2/issues/1307

- 'dada2list()' DADA2 functions return a dataframe whenever the length of the object is == 1 or a list if length > 1. To smooth prevent from bugs, this function standardizes the output of dada2 to lists when length == 1. Unaltered when input is a list.

Value

- 'variant_call_dada()' Numeric array. Column names are variant sequences; row names are sample names and each number in the array indicate the number of reads supporting variant 'i' in sample 'j'.
- 'chimera_removal()' 'seqtab' like object (see dada2).
- 'dada2list()' List with dada2 object/s.

---

variant_calls *List of tidy variants*

Description

List of 5 tidy variants dataframes with variant calls using different strategies.

Format

A list of 5 dataframes.

ind_bs16 pool = F; band_size = 16

ind_bs0 pool = F; band_size = 0

pool_bs16 pool = T; band_size = 16

pool_bs0 pool = T; band_size = 0

amplisas AmpliSAT

---

var_seq2len *Convert sequence-based variants to length-based*

Description

Convert sequence-based variants to length-based

Usage

```
var_seq2len(variants)
```

Arguments

variants	Dataframe with tidy variants.

Details

Recodes dataframe with sequence-based variants, such as the output from 'variant_call' to length-based variants. The sequence length 'nt', instead of the nucleotide sequence, is used for determining variants. Each variant is named after its number of nucleotides. Number of reads from previous variants with equal lengths are aggreated. Legacy for microsatellite data.

Value

Dataframe with length-based variants and the variables: 'locus' 'sample' 'variant' 'reads'.

Examples

```
data("variants")
var_seq2len(variants)
```

Index