Package 'WithinHostExperiment'

November 7, 2026

Title Within-Host Pathogen Variant QC, Diversity, and Bottleneck

Version 0.99.2

Description Provides a SummarizedExperiment-based container and workflow functions for analysing intra-host single nucleotide variants (iSNV) from pathogen deep sequencing data. Supports replicate-aware quality control with full provenance tracking, multi-caller import harmonisation, within-host diversity quantification, and standardised export to transmission bottleneck estimation tools. Designed as infrastructure that bridges upstream variant callers (iVar, LoFreq, Freebayes) with downstream analysis packages through a common data object with uncertainty-preserving QC.

License Artistic-2.0

Depends R (>= 4.6.0)

Imports methods, BiocGenerics, S4Vectors (>= 0.32.0), IRanges, GenomicRanges, SummarizedExperiment, GenomeInfoDb, Biostrings, GenomicFeatures, VariantAnnotation, graphics, stats, tools, utils

Suggests BiocStyle, ggplot2 (>= 3.4.0), patchwork, testthat (>= 3.1.5), knitr, rmarkdown, BiocCheck, rtracklayer, txdbmaker, MSA2dist

biocViews Genetics, VariantAnnotation, Sequencing, GenomicVariation, QualityControl, VariantDetection, DataImport

URL https://github.com/CuiweiG/WithinHostExperiment

BugReports https://github.com/CuiweiG/WithinHostExperiment/issues

Encoding UTF-8

LazyData false

VignetteBuilder knitr

Config/testthat/edition 3

Collate 'AllClasses.R' 'AllGenerics.R' 'WithinHostExperiment-package.R' 'adaptive-filter.R' 'methods.R' 'annotate-genes.R' 'annotate-codon.R' 'bridge.R' 'bottleneck-exact.R' 'bottleneck-quick.R' 'consensus-sequences.R' 'consensus.R' 'contamination.R' 'data.R' 'diversity.R' 'import-table.R' 'import-vcf.R' 'longitudinal.R' 'modeling.R' 'neutrality.R' 'qc-filters.R'

'qc-replicate.R' 'qc-temporal.R' 'selection.R' 'sfs.R'
'strand-bias.R' 'validation.R' 'visualization.R' 'zzz.R'

RoxygenNote 7.3.3

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

git_branch devel

git_last_commit 89c27e8

git_last_commit_date 2026-10-25

Repository Bioconductor 3.24

Date/Publication 2026-11-06

Author Cuiwei Gao [aut, cre]

Maintainer Cuiwei Gao <48gaocuiwei@gmail.com>

Contents

WithinHostExperiment-package . . . 3
adaptiveFilter . . . 6
annotateCodonChange . . . 7
annotateGenes . . . 10
asViralBottleneckInput . . . 12
benchmarkCallers . . . 13
buildSFS . . . 14
buildSFSList . . . 15
calcCallerConcordance . . . 16
calcDiversity . . . 18
calcNeutralityTests . . . 19
calcSharedVariants . . . 21
chao1ISNV . . . 22
checkDiversityPlausibility . . . 23
coerce-WHE . . . 24
compareDiversity . . . 25
compareSFS . . . 25
ConcordanceRule . . . 26
ConcordanceRule-class . . . 27
consensusISNV . . . 28
consensusSequences . . . 30
correctSFSBias . . . 32
detectCrossContamination . . . 33
detectEmergingVariants . . . 35
estimateSelectionCoefficient . . . 36
exactBottleneck . . . 37
example_whe . . . 39
exportPairFrequencies . . . 40
fitDiversityModel . . . 41
flagISNV . . . 41
flagPrimerSites . . . 43
flagReplicateDiscordance . . . 44
flagStrandBiasSCI . . . 45
flagTemporalInconsistency . . . 46
fusFs . . . 48

ISNVFilter 49
ISNVFilter-class 50
multitestCorrection 51
neutralityFromSFS 52
permutationTest 53
piISNV 54
plotFrequencySpectrum 55
plotFrequencyTrajectory 56
plotPairScatter 57
plotQCDashboard 58
plotSFS 58
pNpSWithinHost 59
qcLog 62
qcSummary 63
quickBottleneck 64
readWithinHost 65
readWithinHostTable 67
sciStrandBias 69
shannonISNV 70
simpsonISNV 70
tajimaD 71
trackFrequency 72
wattersonISNV 74
WHE-combine 75
WithinHostExperiment 76
WithinHostExperiment-class 76
WithinHostSFS-class 78

Index **80**

`WithinHostExperiment-package` *WithinHostExperiment: Within-Host Pathogen Variant QC, Diversity, and Bottleneck*

Description

Provides a `RangedSummarizedExperiment-class`-based container and workflow functions for analysing intra-host single nucleotide variants (iSNV) from pathogen deep sequencing data.

Details

The package fills an infrastructure gap between upstream variant callers (iVar, LoFreq, Freebayes) and downstream transmission bottleneck workflows. Its core contribution is replicate-aware, uncertainty-preserving quality control – variants are flagged rather than deleted, preserving full provenance for reproducible analysis. Standardised export functions provide interoperability with tools such as ViralBottleneck.

Key function groups

Import readWithinHost (from VCF), readWithinHostTable (from TSV/CSV)

Quality control ISNVFilter (configure thresholds), flagISNV (apply standard QC), flagReplicateDiscordance (replicate-aware QC), flagPrimerSites (amplicon primer masking), qcSummary (QC report), passedISNV (extract clean subset)

Diversity calcDiversity, shannonISNV, piISNV, wattersonISNV, compareDiversity

Transmission analysis transmissionPairs, exportPairFrequencies, asViralBottleneckInput, calcSharedVariants, quickBottleneck

Neutrality tests tajimaD, fusFs, calcNeutralityTests, simpsonISNV, chao1ISNV

Annotation annotateGenes (labels each site with the genes it overlaps, taking gene ranges as a GRanges or TxDb), annotateCodonChange (the codon and amino acid consequence of each SNV, from VariantAnnotation::predictCoding(), taking CDS records as a GRanges, GRangesList or TxDb and the reference as a DNAStringSet)

Selection analysis pNpSWithinHost (within-host pN/pS), consensusSequences (per-sample consensus sequences as a DNAStringSet, for dN/dS between hosts with **MSA2dist**), estimateSelectionCoefficien

Multi-caller consensus consensusISNV, calcCallerConcordance, benchmarkCallers

Longitudinal analysis trackFrequency, detectEmergingVariants, plotFrequencyTrajectory

Exact bottleneck estimation exactBottleneck

Strand bias sciStrandBias, flagStrandBiasSCI

Temporal QC flagTemporalInconsistency (longitudinal consistency filtering – flag transient iSNVs that appear at only one timepoint)

Cross-contamination detection detectCrossContamination (identify index- hopping signatures between co-sequenced samples)

Adaptive filtering adaptiveFilter

Diversity modelling fitDiversityModel, checkDiversityPlausibility

Site frequency spectrum WithinHostSFS-class (S4 class), buildSFS and buildSFSList (construct from WHE), correctSFSBias (ascertainment bias correction), neutralityFromSFS (Tajima's D + Fu's Fs + Fay & Wu's H from a single SFS), compareSFS (chi-squared SFS comparison), plotSFS (publication-quality SFS plot)

Visualisation plotFrequencySpectrum, plotPairScatter, plotQCDashboard

Scope and limitations

Single-site iSNV resolution This package operates at the level of individual single nucleotide variants (iSNVs). It does not perform haplotype phasing or quasispecies reconstruction, which require read-level (BAM) analysis that is outside the scope of RangedSummarizedExperiment-based containers. For haplotype-aware within-host analysis, consider CliqueSNV (Knyazev et al. 2021 *Nucleic Acids Res* 49:e102), ShoRAH (Zagordi et al. 2011 *BMC Bioinformatics* 12:119), or similar tools. Their output can be imported into WithinHostExperiment for downstream diversity and QC analysis.

Co-infection vs diversity Distinguishing genuine within-host evolution from co-infection (superinfection) requires phylogenetic analysis of full haplotypes. The multi-caller consensus and replicate-concordance modules reduce artefacts but do not resolve this distinction.

Selection coefficient estimation estimateSelectionCoefficient assumes a simple deterministic model. For stochastic (drift-aware) estimates, consider dedicated population-genetics frameworks.

Design philosophy

1. **Flag, don't delete.** QC functions mark variants in the `qcPass` assay rather than removing rows, preserving uncertainty and enabling re-analysis with different thresholds.
2. **Replicate-aware.** Technical replicate concordance is a first-class QC criterion, not a post-hoc check.
3. **Interoperable, don't reinvent.** The package provides standardised export functions compatible with existing downstream tools rather than reimplementing their methods.
4. **Bioconductor-native.** Built on `RangedSummarizedExperiment`, interoperable via `VRanges` coercion.

Author(s)

Maintainer: Cuiwei Gao <48gaocuiwei@gmail.com>

References

Roder AE et al. (2023). Optimized quantification of intra-host viral diversity in SARS-CoV-2 and influenza virus sequence data. *mBio* 14:e01046-23. doi:10.1128/mbio.0104623

Bendall EE et al. (2023). Rapid transmission and tight bottlenecks constrain the evolution of highly transmissible SARS-CoV-2 variants. *Nat. Commun.* 14:272. doi:10.1038/s41467023360015

Sobel Leonard A et al. (2017). Transmission Bottleneck Size Estimation from Pathogen Deep-Sequencing Data, with an Application to Human Influenza A Virus. *J Virol* 91:e00171-17. doi:10.1128/JVI.0017117

Mostefai F et al. (2024). Refining SARS-CoV-2 intra-host variation by leveraging large-scale sequencing data. *NAR Genomics and Bioinformatics* 6:lqae145. doi:10.1093/nargab/lqae145

Zheng B, Johnson PCD, Hughes J (2025). ViralBottleneck: an R package for estimating viral transmission bottlenecks from deep sequencing data using multiple methods. *Virus Evolution* 11:veaf071. doi:10.1093/ve/veaf071

See Also

Useful links:

- https://github.com/CuiweiG/WithinHostExperiment
- Report bugs at https://github.com/CuiweiG/WithinHostExperiment/issues

Examples

```
# List exported functions
ls("package:WithinHostExperiment")
```

`adaptiveFilter` *Create a technology-appropriate ISNVFilter*

Description

Returns an `ISNVFilter` with default thresholds tuned for the error profile and read characteristics of the specified sequencing technology.

Usage

```
adaptiveFilter(technology = c("illumina", "ont", "pacbio_hifi", "generic"))
```

Arguments

`technology`	Character scalar. One of `"illumina"`, `"ont"`, `"pacbio_hifi"`, or `"generic"`.

Details

Each technology has distinct error modes that motivate different default thresholds:

Illumina (`minFreq = 0.03, minDepth = 100`) Illumina short reads have low per-base error rates (~0.1–0.5%) but systematic context-dependent errors. A 3% frequency threshold safely exceeds the noise floor while retaining true low-frequency variants. Grubaugh et al. (2019) measured iSNVs above 3% accurately from at least 1,000 RNA copies sequenced to at least 400x, so 100x is a permissive floor; low-frequency false positives inflate diversity estimates markedly (McCrone & Lauring 2016).

Oxford Nanopore (ONT) (`minFreq = 0.10, minDepth = 60, maxStrandBias = Inf`) ONT reads have higher per-base error rates (~1–5% with R10 chemistry), requiring a more conservative frequency threshold of 10% to avoid false positives; Bull et al. (2020) found that ONT sequencing does not accurately detect variants at low read-count frequencies, and reported over 99% sensitivity and precision for single-nucleotide variants above about 60-fold depth, which sets the depth threshold. Strand bias filtering is disabled (`Inf`) rather than given an untested threshold; supply `maxStrandBias` explicitly when the data support one.

PacBio HiFi (`minFreq = 0.05, minDepth = 50`) HiFi consensus reads reach about 99.8% read accuracy via circular consensus sequencing (Wenger et al. 2019), allowing a frequency threshold below ONT but above Illumina. Depth of 50x reflects the lower throughput but high per-read accuracy.

Generic Falls back to the standard `ISNVFilter` defaults (`minFreq = 0.03, minDepth = 100`), suitable for Illumina or when the technology is unknown.

Value

An `ISNVFilter` object.

References

McCrone JT, Lauring AS (2016). Measurements of intrahost viral diversity are extremely sensitive to systematic errors in variant calling. *J Virol* 90:6884-6895. doi:10.1128/JVI.0066716

Grubaugh ND et al. (2019). An amplicon-based sequencing framework for accurately measuring intrahost virus diversity using PrimalSeq and iVar. *Genome Biology* 20:8. doi:10.1186/s13059018-16187

Bull RA et al. (2020). Analytical validity of nanopore sequencing for rapid SARS-CoV-2 genome analysis. *Nature Communications* 11:6272. doi:10.1038/s41467020200756

Wenger AM et al. (2019). Accurate circular consensus long-read sequencing improves variant detection and assembly of a human genome. *Nature Biotechnology* 37:1155–1162. doi:10.1038/s4158701902179

See Also

`ISNVFilter` for the underlying constructor, `flagISNV` for applying filters.

Examples

```
# Illumina defaults
adaptiveFilter("illumina")

# ONT with strand bias disabled
filt <- adaptiveFilter("ont")
slot(filt, "maxStrandBias")  # Inf

# PacBio HiFi
adaptiveFilter("pacbio_hifi")

# Generic (same as ISNVFilter())
adaptiveFilter("generic")
```

---

annotateCodonChange *Codon and amino acid consequence of each SNV*

Description

Annotates every single-nucleotide substitution in a `WithinHostExperiment` with the codon it changes and the resulting amino acid, by passing the sites to `VariantAnnotation::predictCoding()`. The coding sequences are supplied either as a `TxDb` or as the CDS records of an annotation in a `GRanges` or `GRangesList`; the reference sequence is a `DNAStringSet`, as returned by `Biostrings::readDNAStringSet()`.

Usage

```
annotateCodonChange(x, cds, ref, ...)

## S4 method for signature 'WithinHostExperiment,TxDb,DNAStringSet'
annotateCodonChange(x, cds, ref, collapse = c("longest", "all"), ...)

## S4 method for signature 'WithinHostExperiment,GRangesList,DNAStringSet'
annotateCodonChange(
  x,
  cds,
  ref,
  geneCol = NULL,
  collapse = c("longest", "all"),
  ...
)
```

```
## S4 method for signature 'WithinHostExperiment,GRanges,DNAStringSet'
annotateCodonChange(
  x,
  cds,
  ref,
  txCol = "ID",
  geneCol = "gene",
  collapse = c("longest", "all"),
  ...
)

## S4 method for signature 'WithinHostExperiment,ANY,character'
annotateCodonChange(x, cds, ref, ...)

## S4 method for signature 'WithinHostExperiment,ANY,XString'
annotateCodonChange(x, cds, ref, ...)

## S4 method for signature 'WithinHostExperiment,character,DNAStringSet'
annotateCodonChange(x, cds, ref, ...)
```

Arguments

x	A WithinHostExperiment with ref and alt columns in mcols(rowRanges(x)).
cds	The coding sequences: a TxDb; a GRanges of CDS records; or a GRangesList with one element per transcript, holding its CDS parts.
ref	A DNAStringSet with the reference sequence of each seqlevel carrying coding sequences, for example from Biostrings::readDNAStringSet().
...	Not used; supplying anything is an error.
collapse	Either "longest" (default) or "all"; see Details.
geneCol	The column of gene labels reported as GENEID. For a GRanges it is a column of the CDS records, mcols(cds), defaulting to "gene"; for a GRangesList it is a column of the list's own metadata, one value per transcript, defaulting to NULL. NULL reports the transcript name instead of a gene label.
txCol	For a GRanges cds: the column identifying the transcript each CDS record belongs to (default "ID").

Details

Coding sequences. predictCoding() reads coding sequences from a TxDb. A GRanges of CDS records, for example the rows of rtracklayer::import() with type == "CDS", is grouped into transcripts by the column txCol: in GFF3 files from NCBI and Ensembl the CDS records of one protein share an ID. Each transcript's parts are joined in order of transcription, so a CDS split across a ribosomal frameshift, such as SARS-CoV-2 ORF1ab, is translated as one sequence. The TxDb is then built with txdbmaker::makeTxDb(), which is why that route needs the **txdbmaker** package.

Building a TxDb directly from a GFF3 file with txdbmaker::makeTxDbFromGFF() is not equivalent for every annotation. For the NCBI SARS-CoV-2 annotation (NC_045512.2) it drops the ORF1ab gene, whose two CDS parts overlap by one nucleotide, with a warning that its exon ranks could not be inferred; sites in ORF1a and ORF1ab are then left unannotated. A TxDb gives no record of what was dropped while it was built, so this function cannot detect it. Pass the CDS records as a GRanges when in doubt.

A GFF3 phase column, when present, is honoured: the first part of each transcript is shortened by its phase, the number of bases before the first complete codon, and the phase of every later part must agree with the length of the parts before it. A later part whose phase disagrees would shift the reading frame of the joined sequence, so the function stops.

Reference. Sequence names in `ref` are cut at the first space, so a FASTA header carrying a description still matches the annotation. The function stops if the variant sites and the coding sequences share no sequence name (see `annotateGenes`), or if a sequence carrying coding sequences is missing from `ref`. Sites whose `ref` allele differs from the reference base at that position are left unannotated, with a warning, since they were called against a different sequence. The length of each reference sequence is recorded in `seqlengths(rowRanges(x))`.

Sites annotated. Only single-nucleotide substitutions with alleles A, C, G or T are annotated; insertions, deletions and multi-nucleotide substitutions get `NA`. Alleles are read on the plus strand and complemented by `predictCoding()` for transcripts on the minus strand.

Classification. `CONSEQUENCE` is taken from `predictCoding()` unchanged: `"synonymous"` when the amino acid is kept, `"nonsense"` when a stop codon is gained, and `"nonsynonymous"` for every other change, including the loss of a stop codon. A codon holding an ambiguous base is not translated; such a site is left unannotated, with a warning.

`predictCoding()` applies the initiator rule of `Biostrings::translate()` only when the substitution is the first base of a coding sequence: there an alternative start codon (`CTG` or `TTG`) is reported as `M`, so `ATG` to `CTG` at that one position is synonymous. A change at the second or third base of the first codon is translated with the standard table, giving `L` for `CTG`.

Sites in several transcripts. `predictCoding()` returns one row per site and transcript, so a site where coding sequences overlap (ORF1a and ORF1ab in SARS-CoV-2) has several. With `collapse = "longest"` (the default) the row of the transcript with the longest coding sequence is kept, ties going to the transcript that appears first in `cds`, and a message reports how many sites had more than one transcript and at how many of them the consequence differed between transcripts. With `collapse = "all"` every row is kept: each column becomes a `CharacterList` or `IntegerList` with one element per transcript.

A nucleotide that one transcript reads twice, the nucleotide shared by the two parts of a CDS split across a -1 ribosomal frameshift (SARS-CoV-2 ORF1ab at 13468), gives that transcript two rows, one for each reading. With `collapse = "longest"` the first reading, at the lower CDS position, is kept, and a message says how many sites this affected and at how many the two readings differ in consequence; `collapse = "all"` keeps both. `pNpSWithinHost` counts such a site by the first of its readings in a sense codon.

Value

`x`, with these columns added to `mcols(rowRanges(x))` (`NA`, or an empty element, for sites not annotated):

TXNAME Transcript the row describes.

GENEID Gene of that transcript.

CONSEQUENCE `"synonymous"`, `"nonsynonymous"` or `"nonsense"`; see Details. A site whose reference codon holds an ambiguous base is `NA` in every column.

CDSLOC Position of the site within the coding sequence.

PROTEINLOC Position of the affected codon in the protein.

REFCODON, VARCODON Reference and variant codon, on the transcript's strand.

REFAA, VARAA Reference and variant amino acid, `"*"` for a stop codon.

References

Obenchain V, Lawrence M, Carey V, Gogarten S, Shannon P, Morgan M (2014). VariantAnnotation: a Bioconductor package for exploration and annotation of genetic variants. *Bioinformatics* 30:2076-2078. doi:10.1093/bioinformatics/btu168

See Also

`annotateGenes` for gene labels without codons; `VariantAnnotation::predictCoding()`.

Examples

```
## A 15-nucleotide coding sequence, ATG CTG AAA GGG TAA
ref <- Biostrings::DNAStringSet(c(seg1 = "ATGCTGAAAGGGTAA"))
cds <- GenomicRanges::GRanges("seg1", IRanges::IRanges(1, 15),
    strand = "+", ID = "cds-A", gene = "geneA")
gr <- GenomicRanges::GRanges("seg1",
    IRanges::IRanges(c(1, 6, 13), width = 1))
S4Vectors::mcols(gr)$ref <- c("A", "G", "T")
S4Vectors::mcols(gr)$alt <- c("G", "A", "C")
whe <- WithinHostExperiment(
    assays = list(altFreq = matrix(c(0.1, 0.2, 0.3), ncol = 1)),
    rowRanges = gr,
    colData = S4Vectors::DataFrame(sample_id = "S1"))
if (requireNamespace("txdbmaker", quietly = TRUE)) {
    whe <- annotateCodonChange(whe, cds, ref)
    mc <- S4Vectors::mcols(SummarizedExperiment::rowRanges(whe))
    mc[, c("GENEID", "REFCODON", "VARCODON", "REFAA", "VARAA",
           "CONSEQUENCE")]
}
```

`annotateGenes` *Label variant sites with the genes they overlap*

Description

Adds a `GENE` column to `mcols(rowRanges(x))` giving, for each variant site, the label of the gene range it falls in. Genes are supplied as a `GRanges`, which is how `rtracklayer::import()` returns a GFF3 or GTF file, or as a `TxDb`, from which `GenomicFeatures::genes()` extracts them. The overlap itself is `GenomicRanges::findOverlaps()` with strand ignored, since variant sites carry no strand.

Usage

```
annotateGenes(x, genes, ...)

## S4 method for signature 'WithinHostExperiment,GRanges'
annotateGenes(x, genes, nameCol = "Name", collapse = c("longest", "all"), ...)

## S4 method for signature 'WithinHostExperiment,TxDb'
annotateGenes(
  x,
  genes,
```

```
  nameCol = "gene_id",
  collapse = c("longest", "all"),
  ...
)
```

Arguments

x	A WithinHostExperiment.
genes	A GRanges of gene ranges, or a TxDb.
...	Not used; supplying anything is an error.
nameCol	Character scalar. The column of mcols(genes) holding the label to copy. Defaults to "Name", the GFF3 attribute that rtracklayer::import() returns as a column; for a TxDb it defaults to "gene_id". An unknown column is an error that lists the columns available.
collapse	Either "longest" (default) or "all"; see Details.

Details

Before overlapping, the sequence names in use on both sides are compared. If they share none, the function stops and prints both sets: the same reference is often named differently by different sources (for SARS-CoV-2, NC_045512.2 in RefSeq and MN908947.3 in GenBank), and an overlap across mismatched names would otherwise return a column that is NA everywhere. If they share some but not all, a warning names the variant sequences that have no gene range.

A site can fall in more than one gene where genes overlap. collapse = "longest" keeps the label of the longest overlapping range (the first in genes on a tie) and reports with a message how many sites carried more than one distinct label; collapse = "all" keeps every distinct label in a CharacterList, one element per site, in the order of genes. Several ranges with the same label, such as the two CDS parts of a gene split by a ribosomal frameshift, count as one gene.

Ranges whose label is missing are dropped with a warning; if none is left, the function stops. For a TxDb, GenomicFeatures::genes() leaves out, with a message, genes whose transcripts lie on more than one strand or sequence, so sites in them are not labelled.

Value

x with a GENE column in mcols(rowRanges(x)): a character vector for collapse = "longest", NA for sites in no gene, or a CharacterList for collapse = "all", with an empty element for sites in no gene.

See Also

annotateCodonChange for the codon and amino acid consequence of each substitution.

Examples

```
vcf <- system.file("extdata", "test_donor.vcf",
    package = "WithinHostExperiment")
whe <- readWithinHost(vcf,
    colData = S4Vectors::DataFrame(sample_id = "donor"),
    caller = "ivar")

## Gene ranges built directly
genes <- GenomicRanges::GRanges("test_segment",
```

```
    IRanges::IRanges(c(1, 501), c(500, 1000)), strand = "+",
    Name = c("geneA", "geneB"))
whe <- annotateGenes(whe, genes)
table(S4Vectors::mcols(SummarizedExperiment::rowRanges(whe))$GENE,
    useNA = "ifany")

## The same genes read from a GFF3 file
if (requireNamespace("rtracklayer", quietly = TRUE)) {
    gff <- tempfile(fileext = ".gff3")
    writeLines(c("##gff-version 3",
        "test_segment\t.\tgene\t1\t500\t.\t+\t.\tID=gA;Name=geneA",
        "test_segment\t.\tgene\t501\t1000\t.\t+\t.\tID=gB;Name=geneB"),
        gff)
    features <- rtracklayer::import(gff)
    whe <- annotateGenes(whe, features[features$type == "gene"])
}

## Sequence names that do not match are an error, not a column of NA
renamed <- GenomeInfoDb::renameSeqlevels(genes,
    c(test_segment = "segment_1"))
try(annotateGenes(whe, renamed))
```

`asViralBottleneckInput` *Export data for ViralBottleneck package*

Description

Converts a `WithinHostExperiment` to the input format expected by the **ViralBottleneck** package.

Usage

```
asViralBottleneckInput(
  whe,
  pairId = NULL,
  threshold = 0.03,
  usePassedOnly = TRUE
)
```

Arguments

`whe`	A `WithinHostExperiment`.
`pairId`	Character (optional). Specific pair.
`threshold`	Numeric. Variant calling threshold (default: 0.03).
`usePassedOnly`	Logical (default: TRUE).

Details

One row is written for each donor variant at or above `threshold`. Where the recipient has no call, its frequency is written as 0, which an estimator reads as a variant lost in transmission. A recipient call at or above `threshold` that QC removed is not such an absence: the call may have been removed for lying above `maxFreq`, as when the variant rose towards fixation in the recipient. With

usePassedOnly = TRUE, a donor variant at such a site is therefore left out, and a message gives the number of sites. A removed call below threshold is an absence by the estimators' own definition, and is written as 0. (n_donor_only_recipient_failed_qc in calcSharedVariants counts every removed call above 0, whatever its frequency.) quickBottleneck and exactBottleneck use the same donor variants.

For comprehensive bottleneck analysis with multiple methods (presence-absence, KL divergence, binomial, beta-binomial approximate/exact, and Wright-Fisher), use the **ViralBottleneck** package (Zheng et al. 2025, *Virus Evolution* 11:veaf071). This function prepares a simplified input suitable for the beta-binomial methods.

Value

A data.frame with columns: pos, donor_freq, recipient_freq – a simplified format for beta-binomial MLE. For the full ViralBottleneck input format (per-base read counts), prepare data externally using the read count assays (altCount, refCount).

Examples

```
vcf1 <- system.file("extdata", "test_donor.vcf",
    package = "WithinHostExperiment")
vcf2 <- system.file("extdata", "test_recipient.vcf",
    package = "WithinHostExperiment")
whe <- readWithinHost(c(vcf1, vcf2),
    colData = S4Vectors::DataFrame(
        sample_id = c("donor", "recipient"),
        role = c("donor", "recipient"),
        pair_id = c("p1", "p1")),
    caller = "ivar")
asViralBottleneckInput(whe, pairId = "p1")
```

---

benchmarkCallers *Benchmark variant callers against truth or each other*

Description

If a truth WithinHostExperiment is provided, computes true positives, false positives, false negatives, precision, recall, and F1 for each caller. If no truth is provided, computes pairwise agreement using calcCallerConcordance.

Usage

```
benchmarkCallers(whe_list, truth = NULL)
```

Arguments

whe_list	A named list of WithinHostExperiment objects, one per variant caller.
truth	An optional WithinHostExperiment containing the true variant set. Default NULL.

Value

A DataFrame. When truth is provided:

caller Caller name.

n_total Total sites called by this caller.

TP True positives.

FP False positives.

FN False negatives.

precision TP / (TP + FP).

recall TP / (TP + FN).

F1 Harmonic mean of precision and recall.

When truth is NULL, the result of calcCallerConcordance is returned. A site counts as called, by a caller or in the truth set, when some sample has a frequency above 0 there; a recorded 0 is a site assessed without the allele and is not a call.

Examples

```
gr1 <- GenomicRanges::GRanges("seg1",
    IRanges::IRanges(c(100, 200, 300), width = 1))
S4Vectors::mcols(gr1)$ref <- c("A", "C", "G")
S4Vectors::mcols(gr1)$alt <- c("T", "G", "A")
whe1 <- WithinHostExperiment(
    assays = list(altFreq = matrix(c(0.1, 0.2, 0.3), ncol = 1)),
    rowRanges = gr1,
    colData = S4Vectors::DataFrame(sample_id = "S1"))
benchmarkCallers(list(caller1 = whe1), truth = whe1)
```

---

buildSFS *Build a site frequency spectrum from a WithinHostExperiment*

Description

Constructs a WithinHostSFS object from the allele frequencies in a WithinHostExperiment. One SFS is built per sample. Only segregating sites enter it, those whose frequency lies strictly between 0 and 1, as in calcNeutralityTests: a frequency of 1, a fixed difference from the reference, would otherwise fill the top bin of an unfolded spectrum.

Usage

```
buildSFS(
  whe,
  sampleIdx = 1L,
  nBins = 20L,
  fold = TRUE,
  usePassedOnly = TRUE,
  threshold = 0.03,
  genomeLength = NULL
)
```

Arguments

whe	A WithinHostExperiment.
sampleIdx	Integer or character. The one sample (column) to use, by position or by a column name that occurs once. Default: 1.
nBins	Integer. Number of frequency bins, a whole number of at least 1 (default: 20).
fold	Logical. If TRUE, fold the SFS to minor allele frequency (0 to 0.5), giving polarization = "none". If FALSE, bin the alternative allele frequency from 0 to 1, treating the reference allele as ancestral (polarization = "reference"). Default: TRUE.
usePassedOnly	Logical. If TRUE, only use QC-passed variants. Default: TRUE. The spectrum's maxFreq slot then records the smallest maxFreq among the filters applied with flagISNV, above which no call can appear; with the default ISNVFilter that is 0.5, so an unfolded spectrum is empty above 0.5. To keep the high-frequency classes, flag the unfiltered object with maxFreq = 1: flagISNV() adds to the flags already set, so it cannot restore calls that an earlier filter with a lower maxFreq removed.
threshold	Numeric. Detection threshold (default: 0.03). The first bin starts at this value, not 0. It must be at least 0 and below the top of the spectrum: 0.5 when folded, 1 otherwise.
genomeLength	Integer (optional). The genome length that the per-site statistics of neutralityFromSFS divide by. By default it is read from seqlengths(rowRanges(whe)), summed over every recorded sequence as in calcDiversity, and is NA when that is not recorded. The spectrum itself does not depend on it.

Value

A WithinHostSFS object.

Examples

```
vcf <- system.file("extdata", "test_donor.vcf",
    package = "WithinHostExperiment")
whe <- readWithinHost(vcf,
    colData = S4Vectors::DataFrame(sample_id = "donor"),
    caller = "ivar")
sfs <- buildSFS(whe, genomeLength = 1000L)
sfs
```

---

buildSFSList *Build SFS for all samples in a WithinHostExperiment*

Description

Convenience wrapper that calls buildSFS for each sample and returns a named list of WithinHostSFS objects.

Usage

```
buildSFSList(
  whe,
  nBins = 20L,
  fold = TRUE,
  usePassedOnly = TRUE,
  threshold = 0.03,
  genomeLength = NULL
)
```

Arguments

whe	A WithinHostExperiment.
nBins	Integer. Number of frequency bins, a whole number of at least 1 (default: 20).
fold	Logical. If TRUE, fold the SFS to minor allele frequency (0 to 0.5), giving polarization = "none". If FALSE, bin the alternative allele frequency from 0 to 1, treating the reference allele as ancestral (polarization = "reference"). Default: TRUE.
usePassedOnly	Logical. If TRUE, only use QC-passed variants. Default: TRUE. The spectrum's maxFreq slot then records the smallest maxFreq among the filters applied with flagISNV, above which no call can appear; with the default ISNVFilter that is 0.5, so an unfolded spectrum is empty above 0.5. To keep the high-frequency classes, flag the unfiltered object with maxFreq = 1: flagISNV() adds to the flags already set, so it cannot restore calls that an earlier filter with a lower maxFreq removed.
threshold	Numeric. Detection threshold (default: 0.03). The first bin starts at this value, not 0. It must be at least 0 and below the top of the spectrum: 0.5 when folded, 1 otherwise.
genomeLength	Integer (optional). The genome length that the per-site statistics of neutralityFromSFS divide by. By default it is read from seqlengths(rowRanges(whe)), summed over every recorded sequence as in calcDiversity, and is NA when that is not recorded. The spectrum itself does not depend on it.

Value

A named list of WithinHostSFS objects.

Examples

```
data(example_whe)
sfs_list <- buildSFSList(example_whe, genomeLength = 1000L)
length(sfs_list)
```

---

calcCallerConcordance *Pairwise caller concordance matrix*

Description

For a list of WithinHostExperiment objects (one per variant caller), computes pairwise concordance statistics including shared sites, caller-unique sites, and Jaccard index.

Usage

```
calcCallerConcordance(whe_list)
```

Arguments

whe_list	A named list of WithinHostExperiment objects, one per variant caller.

Value

A DataFrame with columns:

caller1 First caller name.

caller2 Second caller name.

n_shared Number of sites detected by both callers.

n_only1 Sites unique to caller1.

n_only2 Sites unique to caller2.

jaccard Jaccard similarity index (shared / union).

A caller detects a site when some sample has a frequency above 0 there; a recorded 0 is a site assessed without the allele.

References

Roder AE et al. (2023). Optimized quantification of intra-host viral diversity in SARS-CoV-2 and influenza virus sequence data. *mBio* 14:e01046-23. doi:10.1128/mbio.0104623

Examples

```
gr1 <- GenomicRanges::GRanges("seg1",
    IRanges::IRanges(c(100, 200, 300), width = 1))
S4Vectors::mcols(gr1)$ref <- c("A", "C", "G")
S4Vectors::mcols(gr1)$alt <- c("T", "G", "A")
whe1 <- WithinHostExperiment(
    assays = list(altFreq = matrix(c(0.1, 0.2, 0.3), ncol = 1)),
    rowRanges = gr1,
    colData = S4Vectors::DataFrame(sample_id = "S1"))
gr2 <- GenomicRanges::GRanges("seg1",
    IRanges::IRanges(c(100, 200, 400), width = 1))
S4Vectors::mcols(gr2)$ref <- c("A", "C", "T")
S4Vectors::mcols(gr2)$alt <- c("T", "G", "C")
whe2 <- WithinHostExperiment(
    assays = list(altFreq = matrix(c(0.12, 0.22, 0.15), ncol = 1)),
    rowRanges = gr2,
    colData = S4Vectors::DataFrame(sample_id = "S1"))
calcCallerConcordance(list(ivar = whe1, lofreq = whe2))
```

calcDiversity *Compute within-host diversity indices*

Description

Generic for computing diversity metrics from within-host variant data. Methods should return a DataFrame with one row per sample.

Calculates diversity metrics per sample from a WithinHostExperiment. Implements standard population genetics indices adapted for deep sequencing iSNV data, following Lauring (2020).

Usage

```
calcDiversity(x, ...)

## S4 method for signature 'WithinHostExperiment'
calcDiversity(
  x,
  indices = c("shannon", "pi", "watterson", "richness", "simpson", "chao1"),
  usePassedOnly = TRUE,
  ...
)
```

Arguments

x	A WithinHostExperiment.
...	Not used; supplying anything is an error.
indices	Character vector. Which indices to compute: "shannon", "pi", "watterson", "richness", "simpson", "chao1" (default: all six).
usePassedOnly	Logical. If TRUE (default), only use variants where qcPass == TRUE.

Details

Genome length. "pi" and "watterson" are per site and divide by the genome length, which is read from seqlengths(rowRanges(x)) as the summed length of every sequence recorded there, including segments of a segmented genome on which no site lies. It is a property of the reference the calls were made against, not a number to type: readWithinHost records it from the VCF header's ##contig lines or from a DNAStringSet passed as genome, and annotateCodonChange from its reference. When it is not recorded the function stops and says how to record it, for example seqlengths(x) <- c(seg = 29903L).

For "watterson", the sample's median read depth is passed as the sample size n, which for the reasons given in tajimaD makes θ_W small; calcNeutralityTests caps n instead, so the two need not agree.

Reading a zero. A diversity of zero means something different when the sample had no call at all, when every call failed QC, and when calls were present but below the frequency QC allows. Two columns keep these apart: n_sites_assessed, the number of sites at which the sample has a frequency, before QC, and detection_limit, the lowest frequency a call could have kept through the filters applied (the strictest minFreq among them), NA when no filter was applied or usePassedOnly = FALSE. richness counts the sites that entered the indices after QC: the segregating sites, whose

frequency lies strictly between 0 and 1. A frequency of 0, or of 1, a fixed difference from the reference such as iVar reports, is not segregating within the host, so it enters none of the indices; the site count of `"watterson"` and the singletons and doubletons of `"chao1"` come from the same sites. A site counts as assessed only when the sample has a call there: a VCF lists only the sites a caller reported, so a site sequenced without a call cannot be told apart from one not sequenced at all, and neither is counted.

Value

A `DataFrame` of diversity metrics.

A `DataFrame` with one row per sample and columns `sample_id`, `n_sites_assessed`, `detection_limit` and one column per requested index.

References

Lauring AS (2020). Within-Host Viral Diversity: A Window into Viral Evolution. *Annu Rev Virol* 7:63-81. doi:10.1146/annurevvirology010320061642

Examples

```
vcf <- system.file("extdata", "test_donor.vcf",
    package = "WithinHostExperiment")
## the genome length comes from the reference, recorded on import
ref <- Biostrings::readDNAStringSet(system.file("extdata",
    "test_ref.fa", package = "WithinHostExperiment"))
whe <- readWithinHost(vcf,
    colData = S4Vectors::DataFrame(sample_id = "donor"),
    caller = "ivar", genome = ref)
calcDiversity(whe)
```

---

calcNeutralityTests *Compute neutrality test statistics*

Description

Generic for computing neutrality tests from within-host variant data. Methods should return a `DataFrame` with one row per sample.

Calculates neutrality test statistics per sample from a `WithinHostExperiment`. Implements Tajima's D (Tajima 1989) and Fu's Fs (Fu 1997), adapted for deep sequencing iSNV data.

Usage

```
calcNeutralityTests(x, ...)

## S4 method for signature 'WithinHostExperiment'
calcNeutralityTests(
  x,
  indices = c("tajimaD", "fusFs"),
  usePassedOnly = TRUE,
  ...
)
```

Arguments

x	A WithinHostExperiment.
...	Not used; supplying anything is an error.
indices	Character vector. Which tests to compute: "tajimaD", "fusFs" (default: both).
usePassedOnly	Logical. If TRUE (default), only use variants where qcPass == TRUE.

Details

No genome length is needed. Both statistics depend on the nucleotide diversity summed over the genome, $\hat{\pi}L$, which is computed directly from the frequencies, so a per-site value would only be multiplied back by the length it was divided by.

Fu's Fs sets the number of haplotypes to n_sites + 1, the infinite-sites maximum, as fusFs explains. It therefore carries no haplotype information, depends only on n_sites, the effective sample size and $\hat{\pi}L$, and is pushed below 0 under neutrality, increasingly so as θ grows (about -7 at $\theta = 5$ with $n = 100$, near 0 at θ of 1 or less); a negative value is not evidence of expansion.

Value

A DataFrame of neutrality statistics.

A DataFrame with one row per sample and columns sample_id, n_sites (the number of segregating sites each statistic rests on, that is, the sites at which the sample's frequency after QC lies strictly between 0 and 1; a frequency of 0, or of 1, a fixed difference from the reference, is not segregating within the host) and one column per requested test. With n the effective sample size, the sample's median depth capped at 100, a value is NA in four cases (see tajimaD and fusFs): Tajima's D when n_sites is below three, or when n is below four; Fu's Fs when n_sites is 0, or when n_sites + 1 exceeds n, which is the case for every sample with 100 or more segregating sites. For each case that occurs, one warning names how many samples it affected.

References

Tajima F (1989). Statistical method for testing the neutral mutation hypothesis by DNA polymorphism. *Genetics* 123:585-595.

Fu YX (1997). Statistical tests of neutrality of mutations against population growth, hitchhiking and background selection. *Genetics* 147:915-925.

Examples

```
vcf <- system.file("extdata", "test_donor.vcf",
    package = "WithinHostExperiment")
whe <- readWithinHost(vcf,
    colData = S4Vectors::DataFrame(sample_id = "donor"),
    caller = "ivar")
calcNeutralityTests(whe)
```

`calcSharedVariants` *Calculate shared variant statistics*

Description

Counts, for one transmission pair, the variant sites present in both hosts, in the donor alone and in the recipient alone, using the calls that passed QC.

Usage

```
calcSharedVariants(whe, pairId, minDepth = NULL)
```

Arguments

`whe`	A `WithinHostExperiment`.
`pairId`	Character. Which pair to analyse.
`minDepth`	Integer. The recipient depth at or above which an absent donor variant counts as absent at adequate depth. Defaults to the strictest `minDepth` of the filters applied with `flagISNV`, or to 100 when none was applied.

Details

A host carries a variant at a site when its frequency there is known and above 0, as in `exportPairFrequencies`; a recorded frequency of 0 is a site that was assessed and found without the variant.

A donor variant absent from the recipient means one thing when the recipient was sequenced deeply enough at that site to have shown it, and another when it was not. The donor-only count is therefore split by the recipient's read depth at the site, from the `totalDepth` assay: `n_donor_only_adequate_depth` where that depth is at least `minDepth`, `n_donor_only_low_depth` where it is lower, and `n_donor_only_unknown_depth` where it is missing. Depth is missing at every site the recipient's caller did not report when the object was read from VCF files, which list only the sites with a call, so with such input most donor-only sites fall in the last group; a depth matrix from a coverage file, or a table listing every assessed site with its depth and a frequency of 0 where the recipient had no alternative reads (`readWithinHostTable` with `format = "generic"` and `aaf = 0`), is needed to tell the first two apart. A fourth group, `n_donor_only_recipient_failed_qc`, holds the sites where the recipient did make a call, at a frequency above 0, but QC removed it, for example a call above `maxFreq` in a recipient where the variant rose towards fixation: such a site is not absent, so it is kept out of the three depth groups. Only a donor-only site at adequate depth is evidence of loss in transmission.

Value

A named list: `n_donor_only` (the four groups below added together), `n_donor_only_adequate_depth`, `n_donor_only_low_depth`, `n_donor_only_unknown_depth`, `n_donor_only_recipient_failed_qc`, `n_recipient_only`, `n_shared`, `n_total`, `sharing_rate` (shared sites over all sites) and `min_depth`, the threshold used.

Examples

```
vcf1 <- system.file("extdata", "test_donor.vcf",
    package = "WithinHostExperiment")
vcf2 <- system.file("extdata", "test_recipient.vcf",
    package = "WithinHostExperiment")
```

```
whe <- readWithinHost(c(vcf1, vcf2),
    colData = S4Vectors::DataFrame(
        sample_id = c("donor", "recipient"),
        role = c("donor", "recipient"),
        pair_id = c("p1", "p1")),
    caller = "ivar")
calcSharedVariants(whe, pairId = "p1")
```

`chao1ISNV` *Chao1 richness estimator for iSNV data*

Description

Estimates total species (variant) richness using the Chao1 estimator (Chao 1984):

$$\hat{S}_{Chao1} = S_{obs} + \frac{f_1^2}{2f_2}$$

where S_{obs} is the observed number of variants, f_1 is the number of singletons (variants with `altCount == 1`), and f_2 is the number of doubletons (variants with `altCount == 2`). When $f_2 = 0$, the bias-corrected form is used: $S_{obs} + f_1(f_1 - 1)/2$.

Usage

```
chao1ISNV(counts, detected)
```

Arguments

`counts`	Integer vector. Alternative allele read counts (`altCount`) for each variant.
`detected`	Integer scalar. Number of observed variants (S_{obs}).

Value

Numeric scalar. Chao1 richness estimate.

References

Chao A (1984). Nonparametric estimation of the number of classes in a population. *Scand J Statist* 11:265-270.

Examples

```
chao1ISNV(c(1, 1, 2, 5, 10), detected = 5)
```

`checkDiversityPlausibility` *Check diversity estimates against heuristic plausibility ranges*

Description

Compares observed diversity metrics against wide plausibility ranges for common pathogens, as a check on units and orders of magnitude. The ranges are heuristic: they are the package's own, set generously enough that a correctly computed estimate on a typical data set falls inside them, and they are not taken from the literature or from any study. Each is listed in Details. A value inside its range is therefore no evidence that an estimate is correct, and one outside it most often points to a unit, depth or QC problem rather than to biology.

Usage

```
checkDiversityPlausibility(
  div_df,
  pathogen = c("sars_cov_2", "influenza", "hiv", "tb", "generic")
)
```

Arguments

`div_df`	A `DataFrame` or `data.frame` with columns `metric` and `value`, or containing diversity columns such as `pi` and `shannon` (as returned by `calcDiversity`), with an optional `sample_id`.
`pathogen`	Character scalar. Pathogen whose plausibility range is used. One of `"sars_cov_2"`, `"influenza"`, `"hiv"`, `"tb"`, or `"generic"`.

Details

The ranges, per site for π and on the scale `calcDiversity` returns for Shannon entropy:

pathogen	pi	shannon
`sars_cov_2`	1e-6 to 1e-3	0 to 15
`influenza`	1e-5 to 5e-3	0 to 20
`hiv`	1e-4 to 5e-2	0 to 30
`tb`	1e-9 to 1e-4	0 to 5
`generic`	0 to 1	0 to 100

A wide table, as `calcDiversity` returns, is checked row by row, one row per sample and metric; a long table with `metric` and `value` columns is checked as given. Metrics without a range are left out.

Value

A `DataFrame` with one row per sample and metric and columns:

sample_id Sample identifier, taken from `div_df` when it has that column, otherwise the row number; `NA` for long input.

metric Diversity metric name.

observed Observed value.

expected_min Lower bound of the plausibility range.

expected_max Upper bound of the plausibility range.

in_range Logical; whether the observed value falls within the range, and `FALSE` when the value is missing.

Examples

```
div <- S4Vectors::DataFrame(sample_id = c("a", "b"),
    pi = c(1e-4, 0.3), shannon = c(5.0, 7.5))
checkDiversityPlausibility(div, pathogen = "sars_cov_2")
```

`coerce-WHE` *Convert between WithinHostExperiment, data.frame, VRanges and VCF*

Description

`as(x, "data.frame")` gives one row per site and sample with an observed frequency; `as(x, "VRanges")` gives the same rows as a `VRanges`. In the other direction, `as(vr, "WithinHostExperiment")` builds the object from the `refDepth`, `altDepth` and `totalDepth` of a `VRanges`, one sample per `sampleNames` level, and `as(vcf, "WithinHostExperiment")` from a VCF read with `VariantAnnotation::readVcf()` choosing the field layout from the header as `readWithinHost(caller = "auto")` does and taking every sample the file holds. Coercion to and from a `VRanges` keeps the whole `seqinfo`: every recorded sequence and its length, including sequences without a call, since the genome length that `calcDiversity` divides by sums over all of them. The VCF coercion keeps the lengths of the file's `##contig` lines.

Value

A `data.frame`, a `VRanges` or a `WithinHostExperiment`.

See Also

`readWithinHost` for VCF files.

Examples

```
data(example_whe)
df <- as(example_whe, "data.frame")
head(df)

vr <- as(example_whe, "VRanges")
vr
as(vr, "WithinHostExperiment")

vcf <- VariantAnnotation::readVcf(system.file("extdata",
    "test_donor.vcf", package = "WithinHostExperiment"))
as(vcf, "WithinHostExperiment")
```

compareDiversity *Compare diversity between sample groups*

Description

Compare diversity between sample groups

Usage

```
compareDiversity(whe, groupBy, index = "shannon")
```

Arguments

whe	A WithinHostExperiment.
groupBy	Character. Column name in colData to group by.
index	Character. Which index to compare (default: "shannon"). "pi" and "watterson" need the genome length recorded in seqlengths(rowRanges(whe)); see calcDiversity.

Value

A DataFrame with test results.

Examples

```
vcf <- system.file("extdata", "test_donor.vcf",
    package = "WithinHostExperiment")
whe <- readWithinHost(vcf,
    colData = S4Vectors::DataFrame(sample_id = "donor"),
    caller = "ivar")
## Single-group, returns diversity table:
compareDiversity(whe, groupBy = "sample_id", index = "shannon")
```

compareSFS *Compare two within-host SFS objects*

Description

Computes summary statistics and a chi-squared test for homogeneity between two WithinHostSFS objects (e.g., two hosts or two timepoints). A spectrum built before QC can be compared with one built after it only when every filter applied used maxFreq = 1; otherwise their caps differ and the function stops.

Usage

```
compareSFS(sfs1, sfs2)
```

Arguments

sfs1	A WithinHostSFS object.
sfs2	A WithinHostSFS object with the same breaks, folding, polarisation and maxFreq as sfs1.

Details

The test compares raw variant counts in matching bins, so the two spectra must share their breaks, their folding, their polarisation and their QC frequency cap (the `maxFreq` slot), and neither may be bias-corrected: the counts of a corrected spectrum are rescaled and no longer multinomial. The function stops, naming the difference, when any of these does not hold. Bins where both spectra are empty are dropped, and when only two bins remain `chisq.test` applies Yates' continuity correction. No test is run, and the statistic, p-value and diagnostics are all `NA`, when either spectrum is empty or fewer than two bins hold variants. The chi-squared approximation is unreliable when an expected count falls below 5; instead of a warning, that condition is reported in `chisq_approx_ok`, and the p-value should then be read with caution.

Value

A named list with components:

chisq_stat Chi-squared statistic, `NA` when no test is run.

chisq_p Chi-squared p-value, `NA` when no test is run.

df Degrees of freedom, `NA` when no test is run.

min_expected Smallest expected count under homogeneity, or `NA` when either spectrum is empty or fewer than two bins remain, and no test is run.

chisq_approx_ok `TRUE` when every expected count is at least 5, the condition under which `chisq.test` does not warn; `NA` alongside the other `NA` diagnostics when no test is run.

n1 Total variants in sfs1.

n2 Total variants in sfs2.

proportions1 Proportion in each bin, sfs1.

proportions2 Proportion in each bin, sfs2.

Examples

```
data(example_whe)
whe1 <- example_whe
sfs1 <- buildSFS(whe1, genomeLength = 1000L)
sfs2 <- buildSFS(whe1, sampleIdx = 2L, genomeLength = 1000L)
compareSFS(sfs1, sfs2)
```

`ConcordanceRule` *Create a ConcordanceRule object*

Description

Says how the technical replicates of a sample must agree, for use as the `replicateConc` of an `ISNVFilter`.

Usage

```
ConcordanceRule(freqTolerance = 0.05, requireBothDetected = TRUE)
```

Arguments

freqTolerance	Numeric. The largest difference in allele frequency allowed between two replicates that both made the call (default: 0.05); Inf compares no frequencies.
requireBothDetected	Logical. Whether a site called in one replicate of a pair but not in the other fails (default: TRUE).

Details

Neither argument is converted from another type: a factor, a string or a logical freqTolerance is an error, and so is a requireBothDetected that is not TRUE or FALSE, as in flagReplicateDiscordance. An integer tolerance is stored as a double.

Value

A ConcordanceRule object.

See Also

flagISNV, which applies the rule, and flagReplicateDiscordance.

Examples

```
rule <- ConcordanceRule(freqTolerance = 0.05)
rule
ISNVFilter(replicateConc = rule)
```

ConcordanceRule-class *ConcordanceRule: agreement required between technical replicates*

Description

An S4 class saying how the technical replicates of a sample must agree for their calls to pass. It is held in the replicateConc slot of an ISNVFilter, and flagISNV applies it after the per-sample thresholds.

Usage

```
## S4 method for signature 'ConcordanceRule'
show(object)
```

Arguments

object	A ConcordanceRule object.

Details

Replicates are the samples that share a value of the replicate_group column of colData. Every pair of replicates in a group is compared, and a site that fails for any pair is flagged in every sample of the group. A call is a non-missing altFreq, taken before any threshold is applied, so the rule flags what flagReplicateDiscordance flags with the same two settings. Replicate intersection, which keeps only the calls made in both replicates, is ConcordanceRule(freqTolerance = Inf). Frequencies lie in [0, 1], so no two differ by more than 1: a rule with freqTolerance of 1 or more, Inf included, and requireBothDetected = FALSE would flag nothing and is refused.

Value

An object of class ConcordanceRule. Objects are created with the constructor ConcordanceRule; the show method prints the two settings and returns NULL invisibly.

Slots

freqTolerance Numeric scalar. The largest difference in altFreq allowed between two replicates that both made the call (default: 0.05); a difference equal to it, to within rounding error, passes. Inf compares no frequencies.

requireBothDetected Logical scalar. If TRUE (default), a site called in one replicate of a pair but not in the other fails.

See Also

ConcordanceRule for the constructor, ISNVFilter and flagISNV for applying it, flagReplicateDiscordance for the same check on its own.

Examples

```
# Frequencies within 5 points, and called in both replicates
ConcordanceRule()

# Replicate intersection only
ConcordanceRule(freqTolerance = Inf)
```

---

consensusISNV *Multi-caller consensus iSNV detection*

Description

Takes a named list of WithinHostExperiment objects (one per variant caller) and identifies variant sites detected by multiple callers. Sites are matched by genomic position (seqnames + start) and alternative allele. Only sites detected by at least min_callers with a frequency spread no greater than freq_tolerance are retained.

Usage

```
consensusISNV(whe_list, min_callers = 2L, freq_tolerance = 0.05)
```

Arguments

whe_list	A named list of WithinHostExperiment objects, one per variant caller (e.g., list(ivar = whe1, lofreq = whe2)).
min_callers	Integer scalar. Minimum number of callers that must detect a site (default: 2L).
freq_tolerance	Numeric scalar. Maximum allowed range (max - min) of allele frequencies across callers for a site to pass (default: 0.05); a range equal to it, to within rounding error, passes.

Details

When technical replicates are unavailable, using a multi-caller consensus is recommended for reducing false positive iSNV calls (Roder et al. 2023).

Value

A WithinHostExperiment containing only consensus sites, with a column for every sample of the inputs. mcols(rowRanges) includes:

n_callers Number of callers detecting the site; a caller that lists the same site and allele twice counts once.

caller_names Comma-separated caller names.

freq_spread Range of the allele frequencies of the calls, across callers.

Where a site passes in more than one sample, these three columns describe one of those samples, not all of them.

The altFreq assay contains the mean frequency of the calls for each consensus site and NA where the site did not pass in that sample, so a sample with no consensus site is a column of NA. Every call enters the mean and the spread, so a caller that lists a site twice is weighted twice there, though it counts once in n_callers. A sample held by fewer than min_callers inputs cannot reach consensus; it is kept as a column of NA, with a warning naming it, since that column would otherwise read as a sample without iSNVs. The constructor adds a qcPass assay that is TRUE throughout. Read counts, depths and the QC history are not carried over, so run flagISNV on the individual callers' objects rather than on the consensus.

The seqinfo() of the result merges those of the inputs; a sequence whose length, circularity or genome differs between inputs is an error. Each sample's colData row comes from the first input that holds the sample, so host, role and transmission-pair columns carry over.

References

Roder AE et al. (2023). Optimized quantification of intra-host viral diversity in SARS-CoV-2 and influenza virus sequence data. *mBio* 14:e01046-23. doi:10.1128/mbio.0104623

Examples

```
gr <- GenomicRanges::GRanges("seg1",
    IRanges::IRanges(c(100, 200, 300), width = 1))
S4Vectors::mcols(gr)$ref <- c("A", "C", "G")
S4Vectors::mcols(gr)$alt <- c("T", "G", "A")
whe1 <- WithinHostExperiment(
    assays = list(altFreq = matrix(c(0.10, 0.20, 0.30), ncol = 1)),
    rowRanges = gr, colData = S4Vectors::DataFrame(sample_id = "S1"))
gr2 <- GenomicRanges::GRanges("seg1",
    IRanges::IRanges(c(100, 200, 400), width = 1))
S4Vectors::mcols(gr2)$ref <- c("A", "C", "T")
S4Vectors::mcols(gr2)$alt <- c("T", "G", "C")
whe2 <- WithinHostExperiment(
    assays = list(altFreq = matrix(c(0.12, 0.18, 0.15), ncol = 1)),
    rowRanges = gr2, colData = S4Vectors::DataFrame(sample_id = "S1"))
consensus <- consensusISNV(list(caller1 = whe1, caller2 = whe2))
consensus
```

consensusSequences *Consensus sequence of each sample*

Description

Builds, for every sample of a WithinHostExperiment, the reference sequence with each substitution present at or above minFreq written in, and returns the sequences as a DNAStringSet. Its purpose is comparison between hosts: the within-host ratio of pNpSWithinHost describes polymorphism inside one host, whereas dN/dS between hosts is a divergence between their consensus sequences, which the models of MSA2dist::dnastring2kaks() estimate.

Usage

```
consensusSequences(x, ref, ...)

## S4 method for signature 'WithinHostExperiment,DNAStringSet'
consensusSequences(
  x,
  ref,
  minFreq = 0.5,
  usePassedOnly = TRUE,
  seqname = NULL,
  region = NULL,
  ...
)

## S4 method for signature 'WithinHostExperiment,character'
consensusSequences(x, ref, ...)

## S4 method for signature 'WithinHostExperiment,XString'
consensusSequences(x, ref, ...)
```

Arguments

x	A WithinHostExperiment with ref and alt columns in mcols(rowRanges(x)).
ref	A DNAStringSet holding the reference sequence the variants were called against, for example from Biostrings::readDNAStringSet(). Sequence names are cut at the first space.
...	Not used; supplying anything is an error.
minFreq	Numeric scalar in (0, 1]. The frequency at or above which a substitution is written in (default 0.5).
usePassedOnly	Logical. If TRUE (default), only sites with qcPass == TRUE in that sample are written in.
seqname	The sequence of ref to build; required when ref holds more than one sequence and region does not name one.
region	Optional GRanges on seqname, all on one strand: the part of the sequence to return; see Details.

Details

Only substitutions are written in: single-nucleotide changes and multi-nucleotide changes whose reference and alternative alleles have the same length. Insertions and deletions are not applied, with a warning, so every consensus has the length of the reference and stays aligned to it and to every other sample's consensus; that is what `MSA2dist::dnastring2kaks()` needs. Where a deletion or insertion is fixed in a host, the returned sequence therefore differs from that host's true consensus.

A site enters a sample's consensus when its `altFreq` is at least `minFreq` and, with `usePassedOnly = TRUE`, it passed QC. With `minFreq` at or below 0.5 two alleles can qualify at one position; the more frequent is written in, the first in the object on a tie, and a warning reports how many were set aside. A site whose `ref` allele differs from the reference at that position is never written in, with a warning, since it was called against another sequence.

`region` restricts the output to part of the sequence, such as one coding sequence: its ranges are taken in order along the sequence and joined, and the result is reverse-complemented when they lie on the minus strand, so a coding sequence split into parts comes back as one sequence read in its own direction. Positions are those of the reference, since no length-changing edit is made.

Value

A `DNAStringSet` with one sequence per sample, named by sample. Its `mcols()` hold `n_substitutions`, the number of substitutions written into the returned part of each sequence.

See Also

`pNpSWithinHost` for the within-host ratio; `MSA2dist::dnastring2kaks()` for dN/dS between sequences.

Examples

```
## A coding sequence of 15 codons and three hosts, each carrying one
## synonymous and one nonsynonymous change at high frequency:
## host1 6 G>A (CTG -> CTA) and 7 A>G (AAA -> GAA),
## host2 12 G>A (GGG -> GGA) and 13 T>C (TTT -> CTT),
## host3 18 C>A (CCC -> CCA) and 19 A>C (AAA -> CAA)
ref <- Biostrings::DNAStringSet(c(
    seg1 = "ATGCTGAAAGGGTTTCCCAAACTGATGGTTCTGAAAGGGTTTTAA"))
gr <- GenomicRanges::GRanges("seg1",
    IRanges::IRanges(c(6, 7, 12, 13, 18, 19), width = 1))
S4Vectors::mcols(gr)$ref <- c("G", "A", "G", "T", "C", "A")
S4Vectors::mcols(gr)$alt <- c("A", "G", "A", "C", "A", "C")
freqs <- cbind(host1 = c(0.95, 0.91, 0.02, 0.03, 0.01, 0.02),
               host2 = c(0.03, 0.02, 0.88, 0.93, 0.04, 0.01),
               host3 = c(0.01, 0.04, 0.02, 0.01, 0.97, 0.90))
whe <- WithinHostExperiment(
    assays = list(altFreq = freqs),
    rowRanges = gr,
    colData = S4Vectors::DataFrame(sample_id = colnames(freqs),
                                   row.names = colnames(freqs)))
cons <- consensusSequences(whe, ref)
cons

## dN/dS between the hosts' consensus coding sequences: every pair
## differs by two synonymous and two nonsynonymous changes
if (requireNamespace("MSA2dist", quietly = TRUE)) {
    kaks <- suppressMessages(MSA2dist::dnastring2kaks(cons, model = "Li"))
```

```
    kaks[, c("seq1", "seq2", "Ka", "Ks", "Ka/Ks")]
}
```

`correctSFSBias` *Correct ascertainment bias in the within-host SFS*

Description

Deep sequencing imposes a detection threshold (e.g., 3%): sites with true frequency below this threshold are unobservable. This inflates the apparent fraction of moderate-frequency variants relative to rare variants.

Usage

```
correctSFSBias(sfs, method = c("truncation", "binomial"))
```

Arguments

`sfs`	A `WithinHostSFS` object.
`method`	Character. Correction method: `"truncation"` (default) rescales the whole spectrum for the unobservable part of the frequency range; `"binomial"` models the probability of detecting a variant at each frequency given the read depth.

Details

This function applies a correction based on the expected number of unobserved sites under a neutral beta distribution model: given the observed SFS and depth-dependent detectability, it estimates the missing mass in the lowest frequency bins.

The `"truncation"` method assumes a uniform density of variant frequencies and multiplies every bin by $f_{max}/(f_{max} - \text{threshold})$, the reciprocal of the observable fraction of the frequency range. It therefore rescales the spectrum without changing its shape. The `"binomial"` method uses the stored depth to compute $P(\text{detect}|\nu, n) = P(\text{Alt} \geq k_{\min}|n, \nu)$ where $k_{\min} = \lceil n \times \text{threshold} \rceil$, and reweights each bin by the inverse of its detectability. Detectability is floored at 0.01, so a bin is inflated at most one hundredfold.

Either way the corrected counts stay whole numbers of sites, as `fusFs` needs a whole number of segregating sites. The corrected total is rounded once, and its sites are shared out among the bins by largest remainder, ties going to the lower bins; rounding each bin on its own would leave every bin of a few sites unchanged. `nSites` is that total.

Value

A corrected `WithinHostSFS` with `corrected = TRUE`. Counts are no longer raw variant counts, and `nSites` changes accordingly, which also changes the number of segregating sites used by `neutralityFromSFS`. Use uncorrected spectra for `compareSFS`.

Examples

```
vcf <- system.file("extdata", "test_donor.vcf",
    package = "WithinHostExperiment")
whe <- readWithinHost(vcf,
    colData = S4Vectors::DataFrame(sample_id = "donor"),
    caller = "ivar")
sfs <- buildSFS(whe, genomeLength = 1000L)
sfs_corr <- correctSFSBias(sfs, method = "binomial")
sfs_corr
```

`detectCrossContamination` *Detect potential cross-contamination between samples*

Description

Identifies iSNV sites where a high-frequency variant in one sample appears at suspiciously low frequency in another sample from the same sequencing run, consistent with index hopping or sample cross-contamination.

Usage

```
detectCrossContamination(
  whe,
  runCol = NULL,
  sourceMinFreq = 0.2,
  sinkMaxFreq = 0.05,
  sinkMinFreq = 0.005,
  usePassedOnly = TRUE
)
```

Arguments

`whe`	A `WithinHostExperiment` with multiple samples.
`runCol`	Character scalar (optional). Column name in `colData` identifying the sequencing run/batch. If `NULL`, all samples are compared against each other. A name that is not a column of `colData` is an error. Samples whose run is `NA` are left out of every comparison, with a warning giving their number.
`sourceMinFreq`	Numeric scalar. Minimum frequency in the "source" sample for a variant to be considered a potential contaminant origin (default: 0.20).
`sinkMaxFreq`	Numeric scalar. Maximum frequency in the "sink" sample for a variant to be flagged as potential contamination (default: 0.05).
`sinkMinFreq`	Numeric scalar. Minimum frequency in the "sink" sample (default: 0.005). Variants below this are likely noise, not contamination.
`usePassedOnly`	Logical. If `TRUE` (default for source), only consider QC-passed variants in the source sample. A source call that `flagISNV` failed for being above its `maxFreq` (0.5 by default) is still used when it has the read depth, alternative reads and strand balance that every filter in `qcFilters(whe)` asks for: a high frequency is the signature looked for, and the frequency ceiling is there for the iSNV analyses. Other QC steps do not record why a call failed, so a call above `maxFreq` that one of them also flagged is used too.

Details

Index hopping occurs when adapters are exchanged between library molecules during pooled sequencing, causing reads from one sample to be misassigned to another. The signature: a variant at high frequency (e.g., 80%) in sample A appears at very low frequency (e.g., 0.5-2%) in sample B sequenced on the same flow cell, at a ratio consistent with the expected hopping rate (typically 0.1-2% for Illumina platforms).

This function does not require external metadata beyond what is in `colData`. For finer control, use the output `DataFrame` to implement study-specific contamination criteria.

Value

A `DataFrame` with columns:

source_sample Sample where the variant is at high frequency.

sink_sample Sample where the variant appears at suspiciously low frequency.

chrom Chromosome/segment.

position Genomic position.

ref Reference allele.

alt Alternative allele.

source_freq Frequency in the source sample.

sink_freq Frequency in the sink sample.

freq_ratio sink_freq / source_freq – lower values are more suspicious.

run Sequencing run (if `runCol` provided).

References

Bendall EE et al. (2022). SARS-CoV-2 genomic diversity in households highlights the challenges of sequence-based transmission inference. *mSphere* 7:e00400-22. doi:10.1128/msphere.0040022

Costello M et al. (2018). Characterization and remediation of sample index swaps by non-redundant dual indexing on massively parallel sequencing platforms. *BMC Genomics* 19:332. doi:10.1186/s1286401847030

Examples

```
gr <- GenomicRanges::GRanges("seg",
    IRanges::IRanges(c(100, 200, 300), width = 1))
S4Vectors::mcols(gr)$ref <- c("A", "C", "G")
S4Vectors::mcols(gr)$alt <- c("T", "G", "A")
## Sample A has pos 100 at 80%; sample B has same pos at 1%
freq <- matrix(c(0.80, 0.05, 0.30,   # sample A
                 0.01, 0.40, 0.00),   # sample B
               nrow = 3, ncol = 2)
whe <- WithinHostExperiment(
    assays = list(altFreq = freq),
    rowRanges = gr,
    colData = S4Vectors::DataFrame(
        sample_id = c("A", "B"),
        run = c("run1", "run1")))
detectCrossContamination(whe, runCol = "run")
```

`detectEmergingVariants` *Detect emerging variants across time points*

Description

Identifies variants whose allele frequency crosses a threshold between consecutive time points within each host. A variant is considered emerging if it was below `freqThreshold` at one time point and increased by at least `minIncrease` at the next. Replicate samples taken at one time point are combined first: the variant's frequency there is the mean of the replicates' calls, so replicates never form an interval of their own and the result does not depend on the order of the columns. Intervals join the consecutive time points at which the variant has a call in the host; a time point without a call for it is passed over, since no call is not a frequency of 0, so a variant first called at a later time point is not reported.

Usage

```
detectEmergingVariants(
  whe,
  hostCol = "host_id",
  timeCol = "timepoint",
  freqThreshold = 0.05,
  minIncrease = 0.05,
  usePassedOnly = TRUE
)
```

Arguments

`whe`	A `WithinHostExperiment` with longitudinal samples.
`hostCol`	Character scalar. Column name in `colData` for host identifier (default: `"host_id"`). Samples whose host is `NA` are left out, with a warning.
`timeCol`	Character scalar. Column name in `colData` for time point (default: `"timepoint"`), read as in `trackFrequency`: it is put in time order, and samples whose time point is `NA` are left out, with a warning.
`freqThreshold`	Numeric scalar. Frequency threshold for emergence (default: 0.05).
`minIncrease`	Numeric scalar. Minimum frequency increase between consecutive time points (default: 0.05).
`usePassedOnly`	Logical. If TRUE (default), an interval is reported only when the later time point has a call that passed QC, and its frequency is the mean of those calls. The earlier time point is the mean of its passing calls too, or of all its calls when none passed: `flagISNV` fails a frequency below the filter's `minFreq`, and such a frequency is the low start that emergence is measured from. `qc_from` says which. Intervals are formed on every call before QC is applied, so a time point that failed QC, for example above `maxFreq`, is never passed over. With `FALSE` every call is used.

Value

A `DataFrame` with columns:

host_id Host identifier.

variant_key Variant identifier (chrom:pos:alt).

time_from Earlier time point.

time_to Later time point.

freq_from Frequency at the earlier time point.

freq_to Frequency at the later time point.

delta_freq Change in frequency.

qc_from Logical. Whether a call at the earlier time point passed QC (`TRUE` without a `qcPass` assay). A `FALSE` start failed QC, which a frequency below `minFreq` does by design; one that failed for low depth may be set aside.

One row is one interval, between two distinct time points. Rows are ordered by host, then by variant, each in order of first appearance in the object, then by time point.

Examples

```
gr <- GenomicRanges::GRanges("seg1",
    IRanges::IRanges(c(100, 200), width = 1))
S4Vectors::mcols(gr)$ref <- c("A", "C")
S4Vectors::mcols(gr)$alt <- c("T", "G")
freq <- matrix(c(0.02, 0.10, 0.12, 0.15), nrow = 2, ncol = 2)
whe <- WithinHostExperiment(
    assays = list(altFreq = freq),
    rowRanges = gr,
    colData = S4Vectors::DataFrame(
        sample_id = c("S1_t1", "S1_t2"),
        host_id = c("H1", "H1"),
        timepoint = c(1, 2)))
detectEmergingVariants(whe)
```

---

estimateSelectionCoefficient

Estimate selection coefficient from frequency trajectory

Description

Given a vector of allele frequencies observed over equally-spaced time points, estimates the selection coefficient s using the change in logit-transformed frequency per unit time:

$$s = \frac{\Delta \mathrm{logit}(p)}{\Delta t}$$

where $\mathrm{logit}(p) = \log(p/(1-p))$.

Usage

```
estimateSelectionCoefficient(freqs, times = NULL, generation_time = 1)
```

Arguments

`freqs`	Numeric vector. Allele frequencies at successive time points (values must be in (0, 1)).
`times`	Numeric vector (optional). Time values for each observation. If `NULL`, integer time steps 1, 2, ... are used.
`generation_time`	Numeric scalar. Time per generation (default: 1). Used to scale the selection coefficient.

Details

This assumes a simple deterministic selection model. Each consecutive pair of time points yields one estimate of s.

Value

Numeric vector of selection coefficient estimates, one per consecutive time interval. Length is `length(freqs) - 1`.

References

Feder AF et al. (2016). More effective drugs lead to harder selective sweeps in the evolution of drug resistance in HIV-1. *eLife* 5:e10670. doi:10.7554/eLife.10670

Examples

```
freqs <- c(0.05, 0.10, 0.25, 0.50)
estimateSelectionCoefficient(freqs)
estimateSelectionCoefficient(freqs, times = c(0, 3, 7, 14))
```

`exactBottleneck` *Exact bottleneck size estimation*

Description

Estimates transmission bottleneck size using one of three methods:

`exact_bb` Beta-binomial likelihood over the number of transmitted variant virions, which models finite recipient read depth, in the spirit of Sobel Leonard et al. (2017). Requires `altCount` and `totalDepth` assays. Two deviations from the published method are deliberate: when all or none of the transmitted virions carry the variant, the recipient read counts are modelled with a fixed sequencing error probability of `threshold / 2`, and the recipient detection threshold is not integrated out. The likelihood needs the recipient's read depth at each donor variant, so it cannot use a site where that depth is unknown: such a site would add the same constant at every `Nb`. This includes every site that the recipient's VCF file does not list, so with objects read from VCF files a donor variant the recipient lacks usually enters `exact_bb` not at all, and loss in transmission is then scored only by the other estimators. Such sites are left out, with a message, and counted in `n_excluded`, not in `n_variants`; a bootstrap interval (`nboot > 0`) resamples only the sites used. Where the depth is known but the recipient's alternative count is not, as in an iVar table without `ALT_DP`, the count is `round(altFreq * totalDepth)`; where the recipient has no frequency there either, it has no call, and the count is 0.

`wright_fisher` Approximate score in which one generation of binomial drift is combined with a beta kernel of concentration `max(Nb, 2)` around the drifted frequency. It mixes a density for detected variants with probabilities for lost variants, so it is a heuristic rather than a likelihood; prefer `exact_bb` or `presence_absence` for inference. Like `presence_absence`, it uses the donor variants with frequencies from `threshold` to `1 - threshold`.

`presence_absence` Presence-absence likelihood using $P(\text{detect}|Nb, \nu) = 1 - (1 - \nu)^{Nb}$. As in the published method, it uses the donor variants with frequencies from `threshold` to `1 - threshold`, and leaves out, with a message, those above: a donor variant at frequency 1 that the recipient lacks would have probability 0 under every `Nb`.

Usage

```
exactBottleneck(
  whe,
  pairId,
  maxNb = 500L,
  method = c("exact_bb", "wright_fisher", "presence_absence"),
  nboot = 0L,
  threshold = 0.03
)
```

Arguments

`whe`	A `WithinHostExperiment` with transmission pairs defined in `colData`.
`pairId`	Character. Which pair to estimate.
`maxNb`	Integer. Maximum bottleneck size to evaluate (default: 500); a single whole number of at least 1.
`method`	Character. One of `"exact_bb"`, `"wright_fisher"`, or `"presence_absence"`.
`nboot`	Integer. Number of bootstrap replicates for confidence intervals (default: 0, uses a likelihood-ratio interval). Bootstrapping resamples variant sites and uses the random number generator, so set a seed for reproducible intervals.
`threshold`	Numeric. Variant calling threshold (default: 0.03), a single number above 0 and below 0.5: at 0 every recipient frequency counts as present, and from 0.5 no donor variant lies between `threshold` and `1 - threshold`.

Details

Every method stops with an error when its log-likelihood is not finite at any `Nb` of the grid, where it would otherwise return `Nb = 1`.

Value

A named list with components:

`Nb` Numeric. Maximum likelihood estimate of bottleneck size over the grid `1:maxNb`. A warning is issued when the estimate reaches `maxNb`, because the value is then censored by the grid.

`ci` Numeric vector of length 2. 95% confidence interval (likelihood-ratio or bootstrap percentile), also bounded by `maxNb`.

`loglik` Numeric. Log-likelihood at the MLE; for `"wright_fisher"`, the heuristic score at its maximiser.

`method` Character. Method used. For `"wright_fisher"` the `Nb` and `ci` are the maximiser and interval of a heuristic score, and a message says so when the method is run.

`n_variants` Integer. Number of donor variants used.

`n_excluded` Named integer. Donor variants at or above `threshold` that were left out, by reason: `recipient_failed_qc`, sites where QC removed a recipient call at or above `threshold`, which is not an absence (see `asViralBottleneckInput`), left out by every method; `unknown_depth`, sites of unknown recipient depth, which only `exact_bb` leaves out; and `donor_near_fixation`, donor variants above `1 - threshold`, which `presence_absence` and `wright_fisher` leave out, as `quickBottleneck` does.

`pair_id` Character. Pair identifier.

References

Sobel Leonard A et al. (2017). Transmission Bottleneck Size Estimation from Pathogen Deep-Sequencing Data, with an Application to Human Influenza A Virus. *J Virol* 91:e00171-17. doi:10.1128/JVI.0017117

Examples

```
vcf1 <- system.file("extdata", "test_donor.vcf",
    package = "WithinHostExperiment")
vcf2 <- system.file("extdata", "test_recipient.vcf",
    package = "WithinHostExperiment")
whe <- readWithinHost(c(vcf1, vcf2),
    colData = S4Vectors::DataFrame(
        sample_id = c("donor", "recipient"),
        role = c("donor", "recipient"),
        pair_id = c("p1", "p1")),
    caller = "ivar")
exactBottleneck(whe, pairId = "p1", maxNb = 50L,
    method = "presence_absence")
```

`example_whe` *Example WithinHostExperiment dataset*

Description

A small synthetic `WithinHostExperiment` with 3 samples (donor, recipient, independent) containing iSNV data at positions 33 to 945 of a synthetic genome segment. The donor and recipient form a transmission pair (pair_id = "pair_1").

Usage

```
data(example_whe)
```

Format

A `WithinHostExperiment` with 28 variant sites and 3 samples. Assays: altFreq, totalDepth, refCount, altCount, qcPass. The length of the segment, 1000 bases, is recorded in `seqlengths()` from `test_ref.fa`.

Source

Synthetic. Built by `inst/scripts/save_example_data.R` from `test_donor.vcf`, `test_recipient.vcf` and `test_independent.vcf` in `inst/extdata`, read with `readWithinHost`.

Examples

```
data(example_whe)
example_whe
transmissionPairs(example_whe)
```

`exportPairFrequencies` *Export donor-recipient frequency table*

Description

For each transmission pair, creates a table of variant frequencies in the donor and recipient.

Usage

```
exportPairFrequencies(whe, pairId = NULL, usePassedOnly = TRUE)
```

Arguments

`whe`	A `WithinHostExperiment` with `role` and `pair_id` in `colData`.
`pairId`	Character (optional). Specific pair to export. If NULL, exports all pairs.
`usePassedOnly`	Logical. Use only QC-passed variants (default: TRUE).

Value

A `DataFrame` with columns: chrom, position, ref, alt, donor_freq, recipient_freq, shared (logical) and pair_id. A host carries the allele at a site when its frequency there is known and above 0; `shared` is `TRUE` where both hosts carry it. A recorded frequency of 0, such as a site that a table lists without alternative reads, is an assessed site without the allele: it is kept as 0 in its frequency column, and is not shared. A frequency is `NA` where the host has no call, or its call failed QC. Sites that neither host carries are left out.

Examples

```
vcf1 <- system.file("extdata", "test_donor.vcf",
    package = "WithinHostExperiment")
vcf2 <- system.file("extdata", "test_recipient.vcf",
    package = "WithinHostExperiment")
whe <- readWithinHost(c(vcf1, vcf2),
    colData = S4Vectors::DataFrame(
        sample_id = c("donor", "recipient"),
        role = c("donor", "recipient"),
        pair_id = c("p1", "p1")),
    caller = "ivar")
exportPairFrequencies(whe)
```

fitDiversityModel *Fit a linear model to diversity data*

Description

Fits a linear model (lm) relating a diversity metric to one or more predictors. When group_col is supplied it is included as a fixed effect (a full mixed-effects model would require the **nlme** dependency).

Usage

```
fitDiversityModel(div_df, formula, group_col = NULL)
```

Arguments

div_df	A data.frame containing the diversity metrics and predictor columns (e.g.\ output of as.data.frame(calcDiversity(...))).
formula	A formula or character string describing the model (e.g.\ shannon ~ timepoint).
group_col	Character scalar (optional). Column name in div_df representing a grouping variable. Added as a fixed effect if not already present in formula.

Value

A list with components:

model The fitted lm object.

summary_df A data.frame of coefficient estimates, standard errors, t-values, and p-values.

residuals Numeric vector of model residuals.

Examples

```
df <- data.frame(
    shannon = c(1.2, 1.5, 0.8, 1.1, 1.9, 0.7),
    timepoint = c(1, 2, 3, 1, 2, 3),
    group = c("A", "A", "A", "B", "B", "B")
)
res <- fitDiversityModel(df, shannon ~ timepoint)
res_grouped <- fitDiversityModel(df, shannon ~ timepoint,
    group_col = "group")
```

flagISNV *Flag iSNV sites based on quality criteria*

Description

Generic for applying quality control filters to within-host variant data. The default method operates on WithinHostExperiment objects. Other packages can define methods for alternative containers.

Applies quality control filters to a WithinHostExperiment. Instead of removing variants, this function sets the corresponding entries in the qcPass assay to FALSE. Each filtering step is recorded in the qcLog for full reproducibility.

Usage

```
flagISNV(x, filter = ISNVFilter(), ...)

## S4 method for signature 'WithinHostExperiment'
flagISNV(x, filter = ISNVFilter(), ...)
```

Arguments

x	A WithinHostExperiment object.
filter	An ISNVFilter object (default: ISNVFilter()).
...	Must be empty. Any further argument is an error, so that a misspelled argument name cannot be silently ignored.

Details

Filtering steps are applied in order:

1. Minimum read depth (minDepth)
2. Minimum alternative allele frequency (minFreq)
3. Maximum alternative allele frequency (maxFreq)
4. Minimum alternative allele read count (minAltReads)
5. Strand bias (maxStrandBias) – only if fwdAltCount and revAltCount assays exist
6. Replicate concordance (replicateConc) – only if the filter holds a ConcordanceRule

The filter is checked with validObject() first, so a threshold or a rule changed with @<- into one its class refuses is an error rather than a step applied.

Base and mapping quality are not filtered here: the object holds no per-site quality, so those thresholds belong in the variant caller.

The replicate step compares every pair of samples that share a value of the replicate_group column of colData and flags a site that fails the rule in every sample of the group. It compares the calls as made, before the thresholds above, so it flags what flagReplicateDiscordance flags with the same settings, and it records the same two qcLog rows. A rule with no replicate_group column, or with no group of two or more samples, is an error rather than a step that checks nothing. Samples outside any such group are named in a warning and are filtered by the other thresholds only.

Value

The filtered object (same class as input).

A WithinHostExperiment with updated qcPass assay and qcLog.

See Also

ISNVFilter for filter configuration, ConcordanceRule for the replicate rule, passedISNV for extracting passed variants, qcSummary for QC report, flagReplicateDiscordance for replicate QC.

Examples

```
vcf <- system.file("extdata", "test_donor.vcf",
    package = "WithinHostExperiment")
whe <- readWithinHost(vcf,
    colData = S4Vectors::DataFrame(sample_id = "donor"),
    caller = "ivar")
whe <- flagISNV(whe, ISNVFilter(minDepth = 500L))
qcLog(whe)
passedISNV(whe)
```

`flagPrimerSites` *Flag iSNVs in primer binding regions*

Description

For amplicon-based sequencing data, marks variants that fall within primer binding sites in the `qcPass` assay.

Usage

```
flagPrimerSites(whe, primers, action = c("flag", "remove"))
```

Arguments

`whe`	A `WithinHostExperiment`.
`primers`	`GRanges`. Primer binding site coordinates. Can be loaded from a BED file using `rtracklayer::import.bed()`. Their sequence names must match those of the sites: when the two share none, the function stops and prints both, rather than report that no site lies in a primer.
`action`	Character. `"flag"` (default) sets qcPass to FALSE for affected sites, preserving all data for audit. `"remove"` is provided as an explicit convenience for users who require destructive filtering after inspection.

Value

A `WithinHostExperiment`.

Examples

```
library(GenomicRanges)
vcf <- system.file("extdata", "test_donor.vcf",
    package = "WithinHostExperiment")
whe <- readWithinHost(vcf,
    colData = S4Vectors::DataFrame(sample_id = "donor"),
    caller = "ivar")
primers <- GRanges("test_segment", IRanges::IRanges(
    start = c(40, 880), width = c(25, 30)))
whe <- flagPrimerSites(whe, primers)
qcLog(whe)
```

`flagReplicateDiscordance` *Flag iSNVs with replicate discordance*

Description

Identifies variant sites where technical replicates disagree and flags them in the `qcPass` assay.

Usage

```
flagReplicateDiscordance(whe, freqTolerance = 0.05, requireBothDetected = TRUE)
```

Arguments

`whe`	A `WithinHostExperiment` with `replicate_group` and `replicate_id` columns in `colData`.
`freqTolerance`	Numeric scalar. Maximum allowable difference in `altFreq` between replicates (default: 0.05).
`requireBothDetected`	Logical scalar. If `TRUE` (default), sites detected in only one replicate are also flagged. A `freqTolerance` of 1 or more with `requireBothDetected = FALSE` would flag nothing, since frequencies lie in [0, 1], and is an error, as it is for a `ConcordanceRule`.

Details

For each replicate group, the function compares `altFreq` between replicates and flags sites where the difference exceeds `freqTolerance`; a difference equal to it, to within rounding error, passes. Sites are flagged in ALL samples of the replicate group.

Roder et al. (2023) reported that the choice of variant caller and the use of replicate sequencing had the largest effects on single-nucleotide variant discovery. An alternative approach is replicate intersection (requiring detection in both replicates); use `requireBothDetected = TRUE` with `freqTolerance = Inf` to achieve this.

Value

A `WithinHostExperiment` with updated `qcPass` and `qcLog`.

References

Roder AE et al. (2023). Optimized quantification of intra-host viral diversity in SARS-CoV-2 and influenza virus sequence data. *mBio* 14:e01046-23. doi:10.1128/mbio.0104623

See Also

`flagISNV` for standard QC filtering.

Examples

```
rep1 <- system.file("extdata", "test_donor_rep1.vcf",
    package = "WithinHostExperiment")
rep2 <- system.file("extdata", "test_donor_rep2.vcf",
    package = "WithinHostExperiment")
whe <- readWithinHost(c(rep1, rep2),
    colData = S4Vectors::DataFrame(
        sample_id = c("rep1", "rep2"),
        replicate_group = c("donor_reps", "donor_reps"),
        replicate_id = c("1", "2")),
    caller = "ivar")
whe <- flagReplicateDiscordance(whe, freqTolerance = 0.05)
qcLog(whe)
```

`flagStrandBiasSCI` *Flag strand-biased iSNVs using the SCI metric*

Description

Applies Strand Coverage Imbalance filtering to a `WithinHostExperiment`. Sites whose absolute SCI score exceeds `maxSCI` are flagged in the `qcPass` assay.

Usage

```
flagStrandBiasSCI(whe, maxSCI = 1.5)
```

Arguments

`whe`	A `WithinHostExperiment` object. Must contain the assays `fwdAltCount`, `revAltCount`, `fwdTotalCount`, and `revTotalCount`.
`maxSCI`	Numeric scalar. Maximum allowable absolute SCI score (default: 1.5). Sites with `|SCI| > maxSCI` are flagged.

Details

Unlike the simple fold-difference strand bias in `flagISNV`, this function uses the SCI score, which normalises by the global strand coverage ratio (see `sciStrandBias`). This is preferred for amplicon-based sequencing where forward and reverse strand coverage is inherently unbalanced.

The global strand ratio is computed for each sample over the sites that sample called, that is, the sites where its four count assays are not `NA`. Sites the sample did not call keep their `qcPass` value.

Value

A `WithinHostExperiment` with updated `qcPass` and `qcLog`.

References

Mostefai F et al. (2024). Refining SARS-CoV-2 intra-host variation by leveraging large-scale sequencing data. *NAR Genomics and Bioinformatics* 6:lqae145. doi:10.1093/nargab/lqae145

See Also

sciStrandBias for the underlying computation, flagISNV for standard QC.

Examples

```
library(SummarizedExperiment)
library(GenomicRanges)
gr <- GRanges("seg", IRanges::IRanges(start = 1:3, width = 1))
assays <- list(
    altFreq      = matrix(c(0.1, 0.05, 0.2), ncol = 1),
    totalDepth   = matrix(c(1000L, 1000L, 1000L), ncol = 1),
    qcPass       = matrix(TRUE, nrow = 3, ncol = 1),
    fwdAltCount  = matrix(c(50L, 2L, 30L), ncol = 1),
    revAltCount  = matrix(c(45L, 40L, 28L), ncol = 1),
    fwdTotalCount = matrix(c(500L, 500L, 500L), ncol = 1),
    revTotalCount = matrix(c(480L, 480L, 480L), ncol = 1)
)
whe <- WithinHostExperiment(
    rowRanges = gr, assays = assays,
    colData = S4Vectors::DataFrame(sample_id = "s1"))
whe <- flagStrandBiasSCI(whe, maxSCI = 1.5)
qcLog(whe)
```

---

flagTemporalInconsistency *Flag temporally inconsistent iSNVs in longitudinal data*

Description

For longitudinal samples from the same host, identifies iSNVs that appear at only one timepoint (transient) versus those detected at multiple consecutive timepoints (persistent). Transient iSNVs are more likely to be sequencing artefacts; persistent ones are more likely genuine within-host variants.

Usage

```
flagTemporalInconsistency(
  whe,
  hostCol = "host_id",
  timeCol = "timepoint",
  minTimepoints = 2L,
  minConsecutive = NULL,
  usePassedOnly = TRUE
)
```

Arguments

whe	A WithinHostExperiment with longitudinal samples from the same host.
hostCol	Character scalar. Column name in colData for host identifier (default: "host_id").

`timeCol`	Character scalar. Column name in `colData` for time point (default: `"timepoint"`). It must hold numbers, dates, date-times, time differences or a factor whose levels are in time order; text is read as numbers when every value is one, and is otherwise an error, since text such as `"day10"` and `"day2"` has no time order. A blank or `NA` is a missing timepoint.
`minTimepoints`	Integer scalar. Minimum number of distinct timepoints at which an iSNV must be detected to be considered persistent (default: 2). Replicate samples taken at one timepoint count as one timepoint.
`minConsecutive`	Integer scalar. Minimum number of **consecutive** timepoints required, counted over the distinct timepoints at which the host was sampled. If `NULL` (default), any `minTimepoints` detections suffice regardless of consecutiveness. Set to 2 or 3 for stricter filtering.
`usePassedOnly`	Logical. If `TRUE` (default), only consider QC-passed entries; a cell whose `qcPass` is `NA` counts as not passed.

Details

This check applies only where a host was sampled at more than one timepoint. Detections are counted per distinct timepoint: a site called in any sample taken at a timepoint, including any of its technical replicates, is detected at that timepoint, so replicates alone cannot make a site persistent. A host sampled at a single timepoint, however many replicates it has, is not classified. Samples whose timepoint is `NA` are not counted, with a warning, but are flagged with the rest of their host. Transient sites are flagged by setting `qcPass` to `FALSE` where it was `TRUE`; `NA` cells are left as they are.

The biological rationale: genuine within-host variants arise from viral replication and should persist (or change smoothly) across consecutive timepoints. An iSNV that appears once at 5% and is absent at all other timepoints is more likely a sequencing or library preparation artefact.

The function records the filtering step in `qcLog` with step name `"temporal_consistency"`.

Value

A `WithinHostExperiment` with updated `qcPass` assay and a new column `temporal_class` in `mcols(rowRanges)`. Sites are classified within each host; a transient site is flagged only in the samples of the hosts where it is transient. The site-level `temporal_class` is `"persistent"` if the site is persistent in at least one host, `"transient"` if it is transient in every host where it was detected, and `NA` where it was never detected.

References

Farjo M et al. (2024). Within-host evolutionary dynamics and tissue compartmentalization during acute SARS-CoV-2 infection. *J. Virol.* 98:e01618-23. doi:10.1128/jvi.0161823

Examples

```
gr <- GenomicRanges::GRanges("seg1",
    IRanges::IRanges(c(100, 200, 300), width = 1))
S4Vectors::mcols(gr)$ref <- c("A", "C", "G")
S4Vectors::mcols(gr)$alt <- c("T", "G", "A")
freq <- matrix(c(0.1, NA,  0.2,   # var1: 2 timepoints
                 0.05, NA,  NA,    # var2: 1 timepoint (transient)
                 0.15, 0.2, 0.1),  # var3: 3 timepoints
               nrow = 3, ncol = 3)
whe <- WithinHostExperiment(
```

```
    assays = list(altFreq = freq),
    rowRanges = gr,
    colData = S4Vectors::DataFrame(
        sample_id = c("t1", "t2", "t3"),
        host_id = c("H1", "H1", "H1"),
        timepoint = c(1, 2, 3)))
whe <- flagTemporalInconsistency(whe, minTimepoints = 2L)
S4Vectors::mcols(SummarizedExperiment::rowRanges(whe))$temporal_class
```

`fusFs` *Fu's Fs statistic*

Description

Computes Fu's Fs (Fu 1997), a test of neutrality based on the probability of observing at least the observed number of haplotypes given the estimate of theta from nucleotide diversity. Under the infinite-sites model:

$$F_s = \ln\left(\frac{S'}{1 - S'}\right)$$

where S' is the probability of observing a number of haplotypes greater than or equal to the observed value, given $\theta = \hat{\pi}$.

Usage

```
fusFs(S, n, theta_pi)
```

Arguments

`S`	Integer scalar. Number of segregating sites; the haplotype count is taken to be $S + 1$ (see Details).
`n`	Numeric scalar. Sample size. As in `tajimaD`, do **not** use raw read depth: use an effective sample size that reflects the number of independently sampled genomes.
`theta_pi`	Numeric scalar. Theta estimated from nucleotide diversity ($\hat{\pi} \times L$).

Details

The haplotype count is not observed. Per-site calls cannot say how many distinct haplotypes a sample holds, so the function sets that number to $S + 1$. Under infinite sites without recombination this is the largest number of haplotypes S segregating sites can produce, and it is usually far above the number present. The value therefore carries no haplotype information: it depends only on S, n and `theta_pi`, the summaries Tajima's D already contrasts, and under neutrality it is pushed below 0, increasingly so as θ grows. With $n = 100$ and S and `theta_pi` at their neutral expectations, it is -1.46 at $\theta = 2$ ($S = 10$), -7.00 at $\theta = 5$ ($S = 26$) and -25.0 at $\theta = 10$ ($S = 52$), where the expected numbers of haplotypes are 8.4, 15.7 and 24.4 rather than 11, 27 and 53; at $\theta = 1$ ($S = 5$) it is -0.37, and at very small θ it can be positive (+0.55 at $\theta = 0.2$, $S = 1$). A negative value is therefore not evidence of population expansion or hitchhiking, and no fixed threshold calibrates it.

The probability is computed exactly, for every `n`, from Ewens' sampling formula, $P(K = k) = |s(n, k)|\theta^k / \theta_{(n)}$, with the unsigned Stirling numbers of the first kind built in log space. S' is held below `1 - .Machine$double.eps`, so the value cannot exceed about 36.

Value

Numeric scalar. Fu's Fs, or `NA` with a warning in three cases: at $S = 0$, where $S' = 1$ and Fs is infinite; at `theta_pi = 0` with $S > 0$, where there is no diversity to condition on; and when $S + 1 > n$, since the statistic uses $S + 1$ as the number of haplotypes, which cannot exceed the number of sequences sampled.

References

Fu YX (1997). Statistical tests of neutrality of mutations against population growth, hitchhiking and background selection. *Genetics* 147:915-925.

Examples

```
fusFs(S = 10, n = 100, theta_pi = 5)
```

`ISNVFilter` *Create an ISNVFilter object*

Description

Constructs a quality control filter configuration for use with `flagISNV`.

Usage

```
ISNVFilter(
  minDepth = 100L,
  minFreq = 0.03,
  maxFreq = 0.5,
  minAltReads = 10L,
  maxStrandBias = 10,
  replicateConc = NULL
)
```

Arguments

`minDepth`	Integer. Minimum read depth (default: 100).
`minFreq`	Numeric. Minimum alt allele frequency (default: 0.03).
`maxFreq`	Numeric. Maximum alt allele frequency (default: 0.50).
`minAltReads`	Integer. Minimum alt allele reads (default: 10).
`maxStrandBias`	Numeric. Max strand bias fold-diff (default: 10).
`replicateConc`	A `ConcordanceRule` saying how technical replicates must agree, which `flagISNV` applies after the other thresholds, or `NULL` (default) for none.

Value

An `ISNVFilter` object.

Examples

```
ISNVFilter()
ISNVFilter(minDepth = 500L, minFreq = 0.05)
ISNVFilter(replicateConc = ConcordanceRule(freqTolerance = 0.05))
```

ISNVFilter-class *ISNVFilter: Quality control filter for iSNV analysis*

Description

An S4 class holding parameters for filtering intra-host single nucleotide variants. Used by flagISNV.

Usage

```
## S4 method for signature 'ISNVFilter'
show(object)
```

Arguments

object	An ISNVFilter object.

Details

Default values are based on commonly used thresholds in the within-host pathogen diversity literature:

minDepth = 100 Minimum coverage for variant detection. This is a permissive floor: Grubaugh et al. (2019, Genome Biology) measured iSNVs above 3% accurately from at least 1,000 RNA copies sequenced to at least 400x.

minFreq = 0.03 3% minor allele frequency threshold, the lower limit of accurate iSNV measurement in Grubaugh et al. (2019); Popa et al. (2020, Science Translational Medicine) used 2%.

maxFreq = 0.50 Variants above 50% are considered consensus-level changes, not intra-host variants.

minAltReads = 10 Minimum supporting reads to reduce false positives from sequencing errors.

maxStrandBias = 10 Maximum fold-difference between forward and reverse strand alt allele counts. Excessive strand bias suggests systematic error.

Value

An object of class ISNVFilter. Objects are created with the constructor ISNVFilter; the show method prints the filter thresholds and returns NULL invisibly.

Slots

minDepth Integer scalar. Minimum read depth at iSNV site (default: 100).

minFreq Numeric scalar. Minimum alternative allele frequency (default: 0.03).

maxFreq Numeric scalar. Maximum alternative allele frequency; variants above this are excluded as near-fixed (default: 0.50).

minAltReads Integer scalar. Minimum reads supporting the alternative allele (default: 10).

maxStrandBias Numeric scalar. Maximum strand bias as fold-difference between strands (default: 10.0).

replicateConc A ConcordanceRule saying how technical replicates must agree, or NULL (default) for no replicate check. flagISNV applies it after the other thresholds.

References

Roder AE et al. (2023). Optimized quantification of intra-host viral diversity in SARS-CoV-2 and influenza virus sequence data. *mBio* 14:e01046-23. doi:10.1128/mbio.0104623

McCrone JT, Lauring AS (2016). Measurements of intrahost viral diversity are extremely sensitive to systematic errors in variant calling. *J Virol* 90:6884-6895. doi:10.1128/JVI.0066716

Grubaugh ND et al. (2019). An amplicon-based sequencing framework for accurately measuring intrahost virus diversity using PrimalSeq and iVar. *Genome Biology* 20:8. doi:10.1186/s13059018-16187

Mostefai F et al. (2024). Refining SARS-CoV-2 intra-host variation by leveraging large-scale sequencing data. *NAR Genomics and Bioinformatics* 6:lqae145. doi:10.1093/nargab/lqae145

See Also

`ISNVFilter` for the constructor, `flagISNV` for applying filters.

Examples

```
# Default filter
ISNVFilter()

# Stringent filter
ISNVFilter(minDepth = 500L, minFreq = 0.05)
```

`multitestCorrection` *Multiple testing correction for p-values*

Description

Thin wrapper around `p.adjust` for convenient multiple testing correction of p-value vectors.

Usage

```
multitestCorrection(pvalues, method = "BH")
```

Arguments

`pvalues`	Numeric vector of raw p-values.
`method`	Character scalar. Correction method passed to `p.adjust` (default: `"BH"` for Benjamini-Hochberg).

Value

Numeric vector of adjusted p-values (same length as `pvalues`).

Examples

```
raw_p <- c(0.001, 0.04, 0.03, 0.5)
multitestCorrection(raw_p)
multitestCorrection(raw_p, method = "bonferroni")
```

`neutralityFromSFS` *Compute all neutrality statistics from a WithinHostSFS*

Description

Computes Tajima's D, Fu's Fs, and Fay and Wu's H from a single `WithinHostSFS` object. This is the recommended interface for neutrality testing: build the SFS once, then derive all statistics from it.

Usage

```
neutralityFromSFS(sfs, n = NULL)
```

Arguments

`sfs`	A `WithinHostSFS` object.
`n`	Integer. Effective sample size for neutrality tests. For deep sequencing data, do **not** use raw depth. Default: `min(100, meanDepth)`, where the spectrum's `meanDepth` slot holds the median depth across sites, as recommended in the `tajimaD` documentation.

Details

Fay and Wu's H measures an excess of high-frequency derived variants, which can indicate positive selection or genetic hitchhiking (Fay and Wu 2000). It needs the frequency of the derived allele at each site, so the function stops on a folded spectrum (`polarization = "none"`): folding maps a derived allele at 0.9 and one at 0.1 to the same bin. For Tajima's D and Fu's Fs alone, which need no polarisation, use `calcNeutralityTests`. An unfolded spectrum built by `buildSFS` has `polarization = "reference"`: its frequencies are those of the alternative allele, so H here assumes that the reference allele is ancestral at every site, which holds only as far as the reference is close to the population the host was infected from. H is defined as:

$$H = \hat{\pi} - \hat{\theta}_H$$

where $\hat{\pi} = \frac{2}{n(n-1)} \sum_i i(n-i)\xi_i$ and $\hat{\theta}_H = \frac{2}{n(n-1)} \sum_i i^2 \xi_i$ are Fay and Wu's estimators on the same classes, and ξ_i is the count of sites with i derived alleles. Each bin stands for the class i nearest its midpoint, `round(mid * n)`, held within 1 to $n-1$. On matching classes both estimators have the same expectation under neutrality, so H is centred on 0 when every class from 1 to $n-1$ is observed and each bin holds one class; the detection threshold, which removes the lowest classes, and wider bins move it (with the default threshold of 0.03 and $n = 100$, leaving out classes 1 and 2 shifts it by about -0.04θ), and $\hat{\pi}$ carries the $n/(n-1)$ correction that `calcNeutralityTests` applies. A negative H indicates an excess of high-frequency variants.

Because H weights the high-frequency classes most, it is uninterpretable when QC has removed them. The function therefore stops on a spectrum whose `maxFreq` slot is below 1, as it is after `flagISNV` with the default `ISNVFilter`, which drops every call above 0.5: H could not then reveal an excess of high-frequency variants, whatever the data. Flag the unfiltered object, before any filter with a lower `maxFreq`, with `ISNVFilter(maxFreq = 1)`, then `buildSFS(fold = FALSE)`.

The per-site values need the spectrum's genome length, which `buildSFS` takes from the object; a spectrum without one is an error rather than a set of per-genome values labelled per site.

Value

A named list with components:

S Number of segregating sites.

n Effective sample size used.

pi Nucleotide diversity per site, Fay and Wu's $\hat{\pi}$ on the classes i; this is the value passed to `tajimaD` and, times the genome length, to `fusFs`.

theta_W Watterson's theta per site.

theta_H Fay and Wu's theta_H per site.

tajimaD Tajima's D.

fusFs Fu's Fs, with the number of haplotypes set to $S + 1$, the infinite-sites maximum. As `fusFs` explains, it then carries no haplotype information, depends only on S, n and $\hat{\pi}$, and is pushed below 0 under neutrality, increasingly so as θ grows, so a negative value is not evidence of expansion.

fayWuH Fay and Wu's H.

sfs_corrected Whether the input SFS was bias-corrected.

polarization The input's polarisation, `"reference"` or `"outgroup"`: which allele `theta_H` and `fayWuH` treat as ancestral.

References

Fay JC, Wu CI (2000). Hitchhiking under positive Darwinian selection. *Genetics* 155:1405-1413.

Examples

```
vcf <- system.file("extdata", "test_donor.vcf",
    package = "WithinHostExperiment")
whe <- readWithinHost(vcf,
    colData = S4Vectors::DataFrame(sample_id = "donor"),
    caller = "ivar")
sfs <- buildSFS(whe, genomeLength = 1000L, fold = FALSE)
neutralityFromSFS(sfs)
```

---

permutationTest *Permutation test for two-group comparison*

Description

Non-parametric permutation test comparing a summary statistic between two groups. The observed test statistic is compared to a null distribution obtained by randomly permuting group labels.

Usage

```
permutationTest(values, groups, n_perm = 999L, statistic = "mean_diff")
```

Arguments

values	Numeric vector. The measured values (e.g.\ a diversity index).
groups	Factor or character vector (same length as values). Must contain exactly two unique levels.
n_perm	Integer scalar. Number of permutations (default: 999). Permutation uses the random number generator, so set a seed for a reproducible p-value.
statistic	Character scalar. The test statistic to compute: "mean_diff" (default) for difference in group means.

Value

A list with components:

observed Numeric scalar. The observed test statistic.

p_value Numeric scalar. Two-sided permutation p-value.

n_perm Integer. Number of permutations performed.

Examples

```
set.seed(42)
vals <- c(rnorm(10, 2), rnorm(10, 3))
grps <- rep(c("A", "B"), each = 10)
permutationTest(vals, grps, n_perm = 499)
```

---

piISNV *Nucleotide diversity (pi) from iSNV frequencies*

Description

Estimates nucleotide diversity following Nei & Li (1979), adapted for deep sequencing allele frequencies:

$$\pi = \frac{n}{n-1} \cdot \frac{2}{L} \sum_i p_i(1-p_i)$$

where p_i is the alternative allele frequency at site i, L is the genome length, and n is the number of reads sampled at a site, for which mean read depth stands in. The $n/(n-1)$ correction removes the finite-sample bias of estimating a frequency from n draws (Nei 1987, Eq. 10.5). This is a different use of "sample size" from the one in tajimaD, where reads must not stand in for independent lineages. When meanDepth is not provided, the uncorrected estimator is returned (equivalent to infinite sample size).

Usage

```
piISNV(freq, genomeLength, meanDepth = NULL)
```

Arguments

freq	Numeric vector. Alternative allele frequencies, between 0 and 1; anything else is an error. NA values are dropped.
genomeLength	Integer scalar. Reference genome length in bp.
meanDepth	Numeric scalar (optional). Mean read depth for finite-sample correction. If NULL, no correction is applied.

Value

Numeric scalar. Nucleotide diversity.

References

Nei M, Li WH (1979). Mathematical model for studying genetic variation in terms of restriction endonucleases. *Proc Natl Acad Sci USA* 76:5269-5273. doi:10.1073/pnas.76.10.5269

Nei M (1987). *Molecular Evolutionary Genetics.* Columbia University Press, New York. Chapter 10.

Examples

```
piISNV(c(0.1, 0.3, 0.5), genomeLength = 13588)
piISNV(c(0.1, 0.3, 0.5), genomeLength = 13588, meanDepth = 1000)
```

---

plotFrequencySpectrum *Plot iSNV frequency spectrum*

Description

Creates a publication-quality histogram or density plot of alternative allele frequencies, optionally faceted by sample or metadata group. Only calls are drawn: a recorded frequency of 0 is a site assessed without the allele and is left out. A site at frequency 1 is drawn unless QC has removed it.

Usage

```
plotFrequencySpectrum(
  whe,
  type = c("histogram", "density"),
  facetBy = NULL,
  usePassedOnly = TRUE,
  threshold = NULL,
  freqRange = NULL
)
```

Arguments

whe	A WithinHostExperiment.
type	Character. "histogram" (default) or "density".
facetBy	Character (optional). Column name in colData to facet by.
usePassedOnly	Logical. Only show QC-passed variants (default: TRUE).
threshold	Numeric scalar (optional). Variant calling threshold to display as reference line. Default NULL (no line shown). Set to e.g. 0.03 for a 3% threshold.
freqRange	Numeric vector of length 2 (optional). Frequency range for the x-axis as proportions, e.g. c(0, 0.5). Default NULL uses the data range.

Value

A ggplot object.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
    vcf <- system.file("extdata", "test_donor.vcf",
        package = "WithinHostExperiment")
    whe <- readWithinHost(vcf,
        colData = S4Vectors::DataFrame(sample_id = "d"),
        caller = "ivar")
    plotFrequencySpectrum(whe, threshold = 0.03)
}
```

`plotFrequencyTrajectory` *Plot allele frequency trajectories over time*

Description

Creates a ggplot2 spaghetti plot of allele frequency over time for each variant within a specified host. Each line represents one variant, coloured with the Okabe-Ito palette (Wong 2011); with more than eight variants the colours repeat.

Usage

```
plotFrequencyTrajectory(
  whe,
  hostId,
  hostCol = "host_id",
  timeCol = "timepoint",
  usePassedOnly = TRUE
)
```

Arguments

`whe`	A `WithinHostExperiment` with longitudinal samples.
`hostId`	Character scalar. Host identifier to plot.
`hostCol`	Character scalar. Column name in `colData` for host identifier (default: `"host_id"`).
`timeCol`	Character scalar. Column name in `colData` for time point (default: `"timepoint"`), read as in `trackFrequency`. Numbers given as text are drawn in numeric order.
`usePassedOnly`	Logical. If TRUE (default), only use QC-passed variants.

Value

A ggplot object.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
    gr <- GenomicRanges::GRanges("seg1",
        IRanges::IRanges(c(100, 200), width = 1))
    S4Vectors::mcols(gr)$ref <- c("A", "C")
    S4Vectors::mcols(gr)$alt <- c("T", "G")
    freq <- matrix(c(0.05, 0.10, 0.15, 0.20), nrow = 2, ncol = 2)
```

```
    whe <- WithinHostExperiment(
        assays = list(altFreq = freq),
        rowRanges = gr,
        colData = S4Vectors::DataFrame(
            sample_id = c("S1_t1", "S1_t2"),
            host_id = c("H1", "H1"),
            timepoint = c(1, 2)))
    plotFrequencyTrajectory(whe, hostId = "H1")
}
```

`plotPairScatter` *Plot donor-recipient frequency scatter*

Description

Creates a publication-quality scatter plot comparing variant frequencies between a donor and recipient, highlighting shared and unique variants with distinct shapes and colours.

Usage

```
plotPairScatter(whe, pairId, threshold = 0.03)
```

Arguments

`whe`	A `WithinHostExperiment` with transmission pairs.
`pairId`	Character. Which pair to plot.
`threshold`	Numeric. Variant calling threshold shown as reference lines (default: 0.03).

Value

A ggplot object.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
    vcf1 <- system.file("extdata", "test_donor.vcf",
        package = "WithinHostExperiment")
    vcf2 <- system.file("extdata", "test_recipient.vcf",
        package = "WithinHostExperiment")
    whe <- readWithinHost(c(vcf1, vcf2),
        colData = S4Vectors::DataFrame(
            sample_id = c("donor", "recipient"),
            role = c("donor", "recipient"),
            pair_id = c("p1", "p1")),
        caller = "ivar")
    plotPairScatter(whe, pairId = "p1")
}
```

plotQCDashboard *QC dashboard plot*

Description

Creates a publication-quality four-panel dashboard showing frequency distribution, read depth, QC filtering waterfall, and per-sample pass/fail summary. Panels a and d draw and count calls, frequencies above 0, as `qcSummary` does: a recorded 0 is a site assessed without the allele.

Usage

```
plotQCDashboard(whe)
```

Arguments

whe	A `WithinHostExperiment`.

Value

A patchwork composite ggplot object.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE) &&
    requireNamespace("patchwork", quietly = TRUE)) {
    vcf <- system.file("extdata", "test_donor.vcf",
        package = "WithinHostExperiment")
    whe <- readWithinHost(vcf,
        colData = S4Vectors::DataFrame(sample_id = "d"),
        caller = "ivar")
    whe <- flagISNV(whe, ISNVFilter())
    plotQCDashboard(whe)
}
```

plotSFS *Plot a within-host site frequency spectrum*

Description

Creates a publication-quality bar plot of a `WithinHostSFS` object.

Usage

```
plotSFS(sfs, normalize = FALSE)
```

Arguments

sfs	A `WithinHostSFS` object, or a named list of `WithinHostSFS` objects for overlay.
normalize	Logical. If `TRUE`, plot proportions instead of counts. Default: `FALSE`.

Value

A ggplot object.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
    vcf <- system.file("extdata", "test_donor.vcf",
        package = "WithinHostExperiment")
    whe <- readWithinHost(vcf,
        colData = S4Vectors::DataFrame(sample_id = "donor"),
        caller = "ivar")
    sfs <- buildSFS(whe, genomeLength = 1000L)
    plotSFS(sfs)
}
```

---

pNpSWithinHost *Within-host pN/pS ratio*

Description

Computes, for each sample, the ratio of nonsynonymous to synonymous polymorphism per site among its iSNVs. This is a within-host polymorphism ratio, pN/pS: it counts variants segregating inside one host, not substitutions fixed between two sequences, so it is not the dN/dS of a between-sequence comparison and carries no multiple-hit correction. Sites are counted with the method of Nei and Gojobori (1986). For dN/dS between hosts or between consensus sequences, use the models in the **MSA2dist** package.

Usage

```
pNpSWithinHost(x, cds, ref, ...)

## S4 method for signature 'WithinHostExperiment,TxDb,DNAStringSet'
pNpSWithinHost(x, cds, ref, usePassedOnly = TRUE, ...)

## S4 method for signature 'WithinHostExperiment,GRangesList,DNAStringSet'
pNpSWithinHost(x, cds, ref, usePassedOnly = TRUE, geneCol = NULL, ...)

## S4 method for signature 'WithinHostExperiment,GRanges,DNAStringSet'
pNpSWithinHost(
  x,
  cds,
  ref,
  usePassedOnly = TRUE,
  txCol = "ID",
  geneCol = "gene",
  ...
)

## S4 method for signature 'WithinHostExperiment,ANY,character'
pNpSWithinHost(x, cds, ref, ...)
```

```
## S4 method for signature 'WithinHostExperiment,ANY,XString'
pNpSWithinHost(x, cds, ref, ...)

## S4 method for signature 'WithinHostExperiment,character,DNAStringSet'
pNpSWithinHost(x, cds, ref, ...)
```

Arguments

x	A WithinHostExperiment with ref and alt columns in mcols(rowRanges(x)).
cds	The coding sequences, as for annotateCodonChange: a TxDb, a GRanges of CDS records, or a GRangesList with one element per transcript.
ref	A DNAStringSet with the reference sequence of each seqlevel carrying coding sequences, for example from Biostrings::readDNAStringSet().
...	Not used; supplying anything is an error.
usePassedOnly	Logical. If TRUE (default), only iSNVs with qcPass == TRUE are counted.
txCol, geneCol	As for annotateCodonChange: the transcript and gene columns of a GRanges or GRangesList cds.

Details

With n_S synonymous and n_N nonsynonymous iSNVs present in a sample, and S synonymous and N nonsynonymous sites in the coding sequences,

$$p_S = n_S/S, \quad p_N = n_N/N, \quad \mathrm{pN/pS} = p_N/p_S.$$

Under the assumptions of the counting model (every substitution equally likely, no selection) the expected ratio is 1. A ratio below 1 is what purifying selection against amino acid changes would give, but a within-host ratio rests on few iSNVs, is highly variable, and should always be read with its counts. It is NA when no synonymous iSNV is present.

Classification. The counts use only the codons and the genetic code that the sites are counted over. This function calls annotateCodonChange on x with the same cds and ref, so annotation columns already present in x are not used, and takes from it, for each transcript reading an iSNV, the reference and variant codon. A reading counts only when its reference codon is a sense codon: an iSNV in a stop codon, including the loss of a stop codon, is counted in neither n_S nor n_N, since a stop codon carries no sites. The reading is then classified by translating both codons with the standard genetic code: synonymous when the amino acid is kept, counting towards n_S, and nonsynonymous otherwise, a change to a stop codon included, counting towards n_N. The initiator rule is not applied, so ATG to CTG or TTG at the first base of a coding sequence is nonsynonymous here, as ATG carries no synonymous site, although annotateCodonChange reports it as synonymous.

The totals take each iSNV by its first counted reading, in the order of collapse = "longest": the longest transcript that reads it in a sense codon, and where that transcript reads the nucleotide twice, the first of those readings in a sense codon. Each gene's counts take it by the first counted reading of that gene's own transcripts, so an iSNV in a codon two genes share counts in both genes, as the codon's sites do, but once in the total. Sites outside every coding sequence, insertions and deletions, and codons holding an ambiguous base are not counted. An iSNV is present in a sample when its frequency is above zero and, with usePassedOnly = TRUE, it passed QC; it is counted once whatever its frequency.

Sites. S and N are summed over the distinct codons of the genome that some coding sequence reads: a codon is identified by the genomic positions of its three bases, so where two coding sequences read the same bases in the same frame (ORF1a inside ORF1ab in SARS-CoV-2) the shared codons are counted once, while coding sequences that overlap in different frames (ORF7a and

ORF7b) contribute their codons separately. Each sense codon carries three sites, split between synonymous and nonsynonymous under the standard genetic code; stop codons and codons with an ambiguous base carry none, so $S + N$ is three times the number of distinct sense codons. When coding sequences of different transcripts overlap, a warning names them with the nucleotides and codons they share. The sites of a gene are summed in the same way over the codons of that gene's own transcripts, so a codon shared by two genes is counted in each of them but once in the total.

Relation to MSA2dist. `MSA2dist::codonmat2pnps()` and `MSA2dist::dnastring2kaks()` compare pairs of aligned sequences codon by codon, so they cannot take a set of per-site iSNV calls, each present in a sample at its own frequency; that is why the sites are counted here. Codon for codon the counts agree: each of the 61 sense codons carries the same Nei-Gojobori synonymous and nonsynonymous sites here as in `MSA2dist::codonmat2pnps()`. They differ only at stop codons, which carry no sites here but a third or two thirds of a synonymous site there, so over a coding sequence holding stop codons the two totals differ.

Value

A `DataFrame` with one row per sample and gene carrying at least one counted iSNV (one row with `gene = NA` when a sample has none):

sample_id Sample identifier.

nS, nN Synonymous and nonsynonymous iSNVs in the sample.

S_sites, N_sites Synonymous and nonsynonymous sites over all coding sequences.

pS, pN The proportions n_S/S and n_N/N.

pNpS p_N/p_S; `NA` when $n_S = 0$.

gene, gene_nS, gene_nN, gene_S_sites, gene_N_sites, gene_pNpS The same quantities for each gene.

References

Nei M, Gojobori T (1986). Simple methods for estimating the numbers of synonymous and nonsynonymous nucleotide substitutions. *Mol Biol Evol* 3:418-426. doi:10.1093/oxfordjournals.molbev.a040410

See Also

`annotateCodonChange` for the reference and variant codons of each iSNV, from which the classification here is made.

Examples

```
## A 15-nucleotide coding sequence, ATG CTG AAA GGG TAA
ref <- Biostrings::DNAStringSet(c(seg1 = "ATGCTGAAAGGGTAA"))
cds <- GenomicRanges::GRanges("seg1", IRanges::IRanges(1, 15),
    strand = "+", ID = "cds-A", gene = "geneA")
## 1 A>G is ATG -> GTG (nonsynonymous); 6 G>A is CTG -> CTA (synonymous)
gr <- GenomicRanges::GRanges("seg1",
    IRanges::IRanges(c(1, 6), width = 1))
S4Vectors::mcols(gr)$ref <- c("A", "G")
S4Vectors::mcols(gr)$alt <- c("G", "A")
whe <- WithinHostExperiment(
    assays = list(altFreq = matrix(c(0.1, 0.2), ncol = 1)),
    rowRanges = gr,
    colData = S4Vectors::DataFrame(sample_id = "S1"))
if (requireNamespace("txdbmaker", quietly = TRUE)) {
```

```
    pNpSWithinHost(whe, cds, ref)
}
```

---

qcLog *Access the QC audit log*

Description

Returns the quality control log recording all filtering steps applied to a `WithinHostExperiment`.

Returns a subset of the `WithinHostExperiment` containing only variant sites where at least one sample passed QC.

Identifies donor-recipient pairs based on `role` and `pair_id` columns in `colData`.

Usage

```
qcLog(x)

qcFilters(x)

passedISNV(x)

transmissionPairs(x)

qcLog(x) <- value

qcFilters(x) <- value

## S4 method for signature 'WithinHostExperiment'
qcLog(x)

## S4 method for signature 'WithinHostExperiment'
qcFilters(x)

## S4 replacement method for signature 'WithinHostExperiment'
qcLog(x) <- value

## S4 replacement method for signature 'WithinHostExperiment'
qcFilters(x) <- value

## S4 method for signature 'WithinHostExperiment'
passedISNV(x)

## S4 method for signature 'WithinHostExperiment'
transmissionPairs(x)
```

Arguments

x	A `WithinHostExperiment` object.
value	A list replacement value.

Value

A DataFrame with columns: step, parameter, value, n_flagged, n_passed, timestamp.

A list of ISNVFilter objects.

A WithinHostExperiment (subset of rows).

A DataFrame with columns: donor (sample_id), recipient (sample_id), pair_id.

The modified WithinHostExperiment.

The modified WithinHostExperiment.

Examples

```
gr <- GenomicRanges::GRanges("seg", IRanges::IRanges(1:3, width = 1))
whe <- WithinHostExperiment(
    assays = list(altFreq = matrix(runif(3), ncol = 1)),
    rowRanges = gr,
    colData = S4Vectors::DataFrame(sample_id = "S1"))
qcLog(whe)
gr <- GenomicRanges::GRanges("seg", IRanges::IRanges(1:3, width = 1))
whe <- WithinHostExperiment(
    assays = list(altFreq = matrix(runif(3), ncol = 1)),
    rowRanges = gr,
    colData = S4Vectors::DataFrame(sample_id = "S1"))
passedISNV(whe)
```

---

qcSummary *Generate a QC summary report*

Description

Produces per-sample summary statistics from a WithinHostExperiment.

Usage

```
qcSummary(whe)
```

Arguments

whe	A WithinHostExperiment.

Value

A DataFrame with one row per sample and columns:

sample_id Sample identifier.

total_sites The sample's calls: sites where its frequency is above 0. A recorded 0 is a site assessed without the allele and is not counted.

passed_sites, flagged_sites The calls whose qcPass is TRUE and FALSE.

pass_rate passed_sites / total_sites, NA without calls.

median_depth, median_freq The median read depth and the median frequency of the calls.

Examples

```
vcf <- system.file("extdata", "test_donor.vcf",
    package = "WithinHostExperiment")
whe <- readWithinHost(vcf,
    colData = S4Vectors::DataFrame(sample_id = "donor"),
    caller = "ivar")
whe <- flagISNV(whe, ISNVFilter(minDepth = 500L))
qcSummary(whe)
```

`quickBottleneck` *Quick bottleneck size estimation (beta-binomial approximate MLE)*

Description

Estimates transmission bottleneck size using the approximate beta-binomial method of Sobel Leonard et al. (2017), which does not model sequencing depth explicitly. For the exact beta-binomial method (which accounts for finite read depth) or other approaches (presence-absence, KL divergence, Wright-Fisher), use the **ViralBottleneck** package (Zheng et al. 2025, *Virus Evolution* 11:veaf071) via `asViralBottleneckInput`.

Usage

```
quickBottleneck(whe, pairId, maxNb = 200L, threshold = 0.03)
```

Arguments

`whe`	A `WithinHostExperiment` with transmission pairs defined in `colData`.
`pairId`	Character. Which pair to estimate.
`maxNb`	Integer. Maximum bottleneck size to evaluate (default: 200); a single whole number of at least 1.
`threshold`	Numeric. Variant calling threshold (default: 0.03), a single number above 0 and below 0.5.

Details

As in the published method, the donor variants used have frequencies from `threshold` to `1 - threshold`; those above are left out, with a message, since a donor variant at frequency 1 that the recipient lacks has probability 0 under every `Nb`. The function stops with an error when the log-likelihood is not finite at any `Nb` of the grid, where it would otherwise return `Nb = 1`.

Value

A named list: Nb (MLE), ci (95% CI), loglik, n_variants, n_excluded, pair_id. `n_excluded` is a named integer, as from `exactBottleneck`, of the donor variants at or above `threshold` left out: `recipient_failed_qc`, sites where QC removed a recipient call at or above `threshold`, which is not an absence (see `asViralBottleneckInput`); `unknown_depth`, always 0 here, since this method does not use depths; and `donor_near_fixation`, donor variants above `1 - threshold`.

References

Sobel Leonard A et al. (2017). Transmission Bottleneck Size Estimation from Pathogen Deep-Sequencing Data, with an Application to Human Influenza A Virus. *J Virol* 91:e00171-17. doi:10.1128/JVI.0017117

Examples

```
vcf1 <- system.file("extdata", "test_donor.vcf",
    package = "WithinHostExperiment")
vcf2 <- system.file("extdata", "test_recipient.vcf",
    package = "WithinHostExperiment")
whe <- readWithinHost(c(vcf1, vcf2),
    colData = S4Vectors::DataFrame(
        sample_id = c("donor", "recipient"),
        role = c("donor", "recipient"),
        pair_id = c("p1", "p1")),
    caller = "ivar")
quickBottleneck(whe, pairId = "p1", maxNb = 50L)
```

`readWithinHost` *Import iSNV data from VCF files*

Description

Reads one or more VCF files produced by variant callers (iVar through a VCF converter, LoFreq, Freebayes, or others) with `VariantAnnotation::readVcf()` and constructs a `WithinHostExperiment`. A VCF already read into R is converted with `as(vcf, "WithinHostExperiment")`, and a `VRanges` with `as(vr, "WithinHostExperiment")`; see `coerce-WHE`.

Usage

```
readWithinHost(vcfFiles, colData, caller = "auto", genome = NULL)
```

Arguments

`vcfFiles`	Character vector. Paths to VCF files (one per sample).
`colData`	`DataFrame` or `data.frame`. Sample metadata. Must contain `sample_id`. May also contain `host_id`, `role`, `pair_id`, `replicate_group`, `replicate_id`, etc.
`caller`	Character scalar. Which field layout the VCF uses: `"ivar"`, `"lofreq"`, `"freebayes"`, `"auto"` (default) or `"generic"`. See Details.
`genome`	Optional `DNAStringSet` holding the reference sequence, as returned by `Biostrings::readDNAStrin` The length of each sequence is recorded in `seqlengths(rowRanges())`, whether or not a call lies on it. It must name the sequences the calls lie on and be long enough to hold them; see Details.

Details

Each VCF file corresponds to one sample. The `colData` argument provides sample metadata and must contain a `sample_id` column whose length matches the number of files. A file holding more than one sample is an error; read it with `readVcf()` and convert the object instead.

`caller = "ivar"` refers to a VCF carrying iVar's per-sample fields (`REF_DP`, `ALT_DP`, `ALT_FREQ`, `ALT_QUAL`). `ivar variants` itself writes a TSV, not a VCF, so such files come from a converter, for instance `ivar_variants_to_vcf.py` in the nf-core/viralrecon pipeline. An iVar TSV is read directly by `readWithinHostTable(format = "ivar")`.

Caller. With `caller = "auto"` the layout is chosen from the VCF header: a header naming iVar, LoFreq or Freebayes (as the `##source` line of each does) selects that layout, and any other gives `"generic"`. The layouts read: iVar, FORMAT `ALT_FREQ`, `ALT_DP` and `REF_DP`, with depth their sum (INFO `DP` where they are missing); LoFreq, INFO `AF` and `DP`, with counts derived from them; Freebayes, FORMAT `AO`, `RO` and `DP` (INFO `DP` where FORMAT has none), with frequency `AO / DP`; generic, the first of those field names that the file carries.

Several alternative alleles. A record with several alternative alleles is split into one site per allele with `VariantAnnotation::expand()`. A per-allele value is split correctly only when the header declares it `Number=A`, as the VCF specification requires; `readVcf()` keeps just the first value of a field declared otherwise. When such a record carries one of the per-allele fields read here (INFO `AF`; FORMAT `AO`, `AF`, `ALT_DP`, `ALT_FREQ`, `ALT_QUAL`) under another `Number`, the function stops rather than give the second allele the first allele's value. Correct the header, or split the records first (for example with `bcftools norm -m-`). On a biallelic record a field declared `Number=1` or `Number=.` is read as it is. A field declared `Number=R` stops the import whatever the records, since it gives the reference allele's value first and that value would be read as the alternative allele's; declare it `Number=A`, with one value per alternative allele.

Sequence lengths. Lengths are recorded in `seqlengths(rowRanges())` from the VCF header's `##contig` lines and from `genome`; two sources that disagree are an error. Every sequence a header line or `genome` names is recorded, including those without a call, since the genome length sums over all of them. Functions that need the genome length read it from there, so `genome` is checked before anything is recorded: it is an error when it names none of the sequences the calls lie on, or when a call lies past the end of its sequence, since either way it is not the reference the calls were made against. A sequence the calls lie on that `genome` does not name is left without a length, with a warning.

What is not read. The `FILTER` column is not read, so a record the caller rejected (LoFreq's `sb_fdr` or `min_snvqual`, for instance) is imported as a call, and `flagISNV` judges it by its depth, frequency and read-count thresholds alone. To drop such records, read the file with `readVcf()`, keep those whose `FILTER` is `PASS` or `.`, write them to a file and import that:

```
vcf <- VariantAnnotation::readVcf(path)
keep <- VariantAnnotation::filt(vcf) %in% c("PASS", ".")
passed <- tempfile(fileext = ".vcf")
VariantAnnotation::writeVcf(vcf[keep, ], passed)
whe <- readWithinHost(passed, colData, caller = "lofreq")
```

This keeps every argument of `readWithinHost()`. Converting the subset directly, with `as(vcf[keep, ], "WithinHostExperiment")`, also works, but the caller is then taken from the `##source` header line (the generic layout when it names none), the sample names from the VCF, and no `genome` can be given.

Per-strand counts, such as LoFreq's `DP4` or Freebayes's `SAF` and `SAR`, are not imported. The strand-bias step of `flagISNV()` runs only when the assays `fwdAltCount` and `revAltCount` are present, so on an imported object it is skipped and leaves no `qcLog` row, although `qcFilters()`

records the filter's maxStrandBias; flagStrandBiasSCI stops for want of those assays and of fwdTotalCount and revTotalCount. Assays added with assay(x, "fwdAltCount") <- and so on before flagISNV() are used.

Insertions and deletions are imported as rows of width 1 at their POS, with ref and alt as the VCF writes them (CTA and C for a deletion of two bases), and nothing tells them apart from substitutions: they pass flagISNV() like any other call and enter the diversity statistics. An iVar TSV read by readWithinHostTable() writes the same deletion in its own notation, so the two routes do not match it by allele. To keep substitutions only, subset to rows whose ref and alt are one base each:

```
rr <- SummarizedExperiment::rowRanges(whe)
whe <- whe[nchar(rr$ref) == 1L & nchar(rr$alt) == 1L, ]
```

When technical replicates are not available, Roder et al. (2023, *mBio* 14:e01046-23) recommend using a combination of multiple variant callers with stringent cutoffs to reduce false positives. To implement multi-caller consensus, import VCFs from each caller separately and intersect the resulting objects using standard GRanges overlap operations.

Value

A WithinHostExperiment.

Examples

```
vcf <- system.file("extdata", "test_donor.vcf",
    package = "WithinHostExperiment")
whe <- readWithinHost(vcf,
    colData = S4Vectors::DataFrame(sample_id = "donor"),
    caller = "ivar")
whe

## The same file read into R first, then converted
v <- VariantAnnotation::readVcf(vcf)
as(v, "WithinHostExperiment")

## Only the records the caller passed (FILTER is PASS or ".")
keep <- VariantAnnotation::filt(v) %in% c("PASS", ".")
as(v[keep, ], "WithinHostExperiment")
```

---

readWithinHostTable *Import iSNV data from tabular files*

Description

Reads pre-computed iSNV data from TSV or CSV files and constructs a WithinHostExperiment.

Usage

```
readWithinHostTable(files, format = c("ivar", "generic"), colData = NULL)
```

Arguments

files	Character vector. Paths to files (one per sample).
format	Character. "ivar" for iVar TSV output (columns: REGION, POS, REF, ALT, REF_DP, ALT_DP, ALT_FREQ, TOTAL_DP, ...), or "generic" for CSV with columns: chrom, pos, ref, alt, aaf, depth.
colData	DataFrame (optional). Sample metadata with sample_id. If NULL, sample IDs are derived from file basenames.

Details

Only the columns named under format are read.

iVar's PASS column, the outcome of its own test of each call, is not read, so a row with PASS FALSE is imported as a call, and flagISNV judges it by its depth, frequency and read-count thresholds alone. To drop such rows, remove them from the file first, reading every column as text so that the file is written back unchanged:

```
calls <- read.delim(path, colClasses = "character")
passed <- tempfile(fileext = ".tsv")
write.table(calls[calls$PASS == "TRUE", ], passed, sep = "\t",
    quote = FALSE, row.names = FALSE)
whe <- readWithinHostTable(passed, format = "ivar")
```

iVar's per-strand counts (REF_RV, ALT_RV) are not imported. The strand-bias step of flagISNV() runs only when the assays fwdAltCount and revAltCount are present, so on an imported object it is skipped and leaves no qcLog row, although qcFilters() records the filter's maxStrandBias; flagStrandBiasSCI stops for want of those assays.

Insertions and deletions are imported as rows of width 1 at their position, with the caller's allele notation (iVar writes +TTG for an insertion and -T for a deletion), and nothing tells them apart from substitutions: they pass flagISNV() like any other call and enter the diversity statistics. A VCF read by readWithinHost writes the same one-base deletion as, for instance, AT to A, so the two routes do not match it by allele. To keep substitutions only, subset to rows whose ref and alt are one base each:

```
rr <- SummarizedExperiment::rowRanges(whe)
whe <- whe[nchar(rr$ref) == 1L & nchar(rr$alt) == 1L, ]
```

Value

A WithinHostExperiment.

Examples

```
tsv <- system.file("extdata", "test_donor_ivar.tsv",
    package = "WithinHostExperiment")
whe <- readWithinHostTable(tsv, format = "ivar")
whe

## Only the rows iVar passed
calls <- read.delim(tsv, colClasses = "character")
passed <- tempfile(fileext = ".tsv")
write.table(calls[calls$PASS == "TRUE", ], passed, sep = "\t",
    quote = FALSE, row.names = FALSE)
readWithinHostTable(passed, format = "ivar")
```

`sciStrandBias` *Strand Coverage Imbalance (SCI) score*

Description

Computes a strand bias score that normalises the per-site forward/reverse alternative-allele ratio by the global strand coverage ratio, making it suitable for amplicon data where strand coverage is inherently unequal. Mostefai et al. (2024) showed that strand bias filters which ignore this imbalance can discard genuine variants; their own metric is a per-strand binomial likelihood.

Usage

```
sciStrandBias(fwd_alt, rev_alt, fwd_total, rev_total)
```

Arguments

`fwd_alt`	Numeric vector. Forward-strand alternative allele read counts. All four count vectors must be numeric and hold non-negative, finite values or `NA`; `NA` marks a site with no call (see Details).
`rev_alt`	Numeric vector. Reverse-strand alternative allele read counts.
`fwd_total`	Numeric vector. Forward-strand total read counts.
`rev_total`	Numeric vector. Reverse-strand total read counts.

Details

The SCI is defined as:

$$\mathrm{SCI} = \log_2\left(\frac{f_{\mathrm{alt}}/r_{\mathrm{alt}}}{F_{\mathrm{total}}/R_{\mathrm{total}}}\right)$$

where F_{total} and R_{total} are the sums of forward and reverse total counts across all sites. A pseudo-count of 1 is added to strand counts to avoid division by zero.

A site where any of the four counts is `NA`, as in a sample that made no call there, scores `NA` and is left out of the sums, so the global ratio is taken over the sites with all four counts.

Value

A numeric vector of SCI scores (log2 scale). Values near zero indicate no strand bias; large absolute values indicate strand-specific enrichment or depletion.

References

Mostefai F et al. (2024). Refining SARS-CoV-2 intra-host variation by leveraging large-scale sequencing data. *NAR Genomics and Bioinformatics* 6:lqae145. doi:10.1093/nargab/lqae145

Examples

```
sci <- sciStrandBias(
    fwd_alt   = c(50, 2, 30),
    rev_alt   = c(45, 40, 28),
    fwd_total = c(500, 500, 500),
    rev_total = c(480, 480, 480)
)
sci
```

shannonISNV *Shannon entropy from iSNV frequencies*

Description

Calculates per-site biallelic Shannon entropy, summed across all iSNV sites:

$$H = -\sum_i [p_i \log(p_i) + (1 - p_i) \log(1 - p_i)]$$

where p_i is the alternative allele frequency at site i. This formulation treats each iSNV as an independent biallelic locus, following McCrone & Lauring (2016) and Popa et al. (2020). Higher values indicate greater within-host diversity.

Usage

```
shannonISNV(freq)
```

Arguments

`freq`	Numeric vector. Alternative allele frequencies between 0 and 1; anything else is an error. Values of 0 and 1, which carry no diversity, and NA are excluded.

Value

Numeric scalar. Shannon entropy.

References

Lauring AS (2020). Within-Host Viral Diversity: A Window into Viral Evolution. *Annu Rev Virol* 7:63-81. doi:10.1146/annurevvirology010320061642

Examples

```
shannonISNV(c(0.05, 0.15, 0.30, 0.45))
shannonISNV(0.5) # log(2)
```

simpsonISNV *Simpson diversity index for iSNV frequencies*

Description

Calculates Simpson diversity summed over biallelic iSNV sites:

$$D = \sum_i \left[1 - p_i^2 - (1 - p_i)^2\right] = \sum_i 2p_i(1 - p_i)$$

where p_i is the alternative allele frequency at site i. Each site contributes the probability that two alleles drawn at random at that site differ (at most 0.5), so the index is the expected number of differing sites. Higher values indicate greater diversity.

Usage

```
simpsonISNV(freq)
```

Arguments

freq	Numeric vector. Alternative allele frequencies between 0 and 1; anything else is an error. Values of 0 and 1, which carry no diversity, and NA are excluded.

Value

Numeric scalar. Simpson diversity index.

Examples

```
simpsonISNV(c(0.05, 0.15, 0.30, 0.45))
simpsonISNV(0.5) # maximum per-site diversity = 0.5
```

---

tajimaD *Tajima's D statistic*

Description

Computes Tajima's D (Tajima 1989) from summary statistics of a within-host pathogen population. The statistic measures the difference between nucleotide diversity ($\hat{\pi}$) and Watterson's estimator ($\hat{\theta}_W = S/a_1$), normalised by its expected standard deviation under neutrality:

$$D = \frac{\hat{\pi} - S/a_1}{\sqrt{e_1 S + e_2 S(S-1)}}$$

A significantly negative D indicates an excess of rare variants (e.g., population expansion or purifying selection), while a positive D suggests balancing selection or population contraction.

Usage

```
tajimaD(S, n, pi_hat, genomeLength)
```

Arguments

S	Integer scalar. Number of segregating sites.
n	Numeric scalar. Sample size. For deep sequencing data, do **not** use raw read depth directly – the classical Tajima's D assumes independent lineages, whereas sequencing reads oversample a much smaller viral population. Use an effective sample size (e.g., n = 50 or n = 100) that reflects the number of independently sampled viral genomes, or cap the depth at a biologically meaningful value. See Details.
pi_hat	Numeric scalar. Nucleotide diversity estimate (per-site).
genomeLength	Integer scalar. Reference genome length in bp.

Details

Fewer than three segregating sites: with one segregating site D is a monotone function of that site's frequency, and with two it is a function of two frequencies; neither describes a frequency spectrum, and at `S = 0` the statistic is 0/0. The function therefore returns `NA`, with a warning, whenever `S < 3`. Report `S` alongside every D, as `calcNeutralityTests` does in its `n_sites` column. With two or three sequences the variance of D is zero ($c_1 = c_2 = 0$), so `n` must round to at least four; below that D is likewise `NA`.

Deep sequencing caveat: The classical Tajima (1989) formula was designed for independent haplotype samples (e.g., Sanger-sequenced clones or individual genomes). In deep sequencing, reads are not independent lineages; they are technical replicates of a viral population whose effective size is much smaller than the read depth. Using raw read depth (e.g., n = 2000) inflates the harmonic number $a_1 = \sum 1/i$, making θ_W artificially small and biasing D towards positive values.

We recommend capping `n` at a biologically meaningful effective sample size. Lauring (2020 *Annu Rev Virol*) suggests n = 100 as a conservative proxy. The high-level `calcNeutralityTests` method applies this cap automatically (`n = min(depth, 100)`).

Interpretation warning: Deep-sequencing D values are *not* directly comparable to classical population-genetic D computed from independently sampled haplotypes. We recommend reporting D as a *descriptive statistic* of the site frequency spectrum (e.g., "the majority of samples show negative D, consistent with an excess of rare variants"), rather than as a formal neutrality test with p-value interpretation. For rigorous neutrality testing on deep sequencing data, consider simulation-based approaches that account for the read-sampling process.

Value

Numeric scalar. Tajima's D, or `NA` with a warning when `S` is below three or `n` rounds to fewer than four.

References

Tajima F (1989). Statistical method for testing the neutral mutation hypothesis by DNA polymorphism. *Genetics* 123:585-595.

Lauring AS (2020). Within-Host Viral Diversity: A Window into Viral Evolution. *Annu Rev Virol* 7:63-81. doi:10.1146/annurevvirology010320061642

Examples

```
tajimaD(S = 10, n = 1000, pi_hat = 0.001, genomeLength = 10000)
```

`trackFrequency` *Track variant frequency trajectories over time*

Description

Extracts per-variant allele frequency trajectories across longitudinal time points within each host. Returns a long-format `DataFrame` suitable for downstream analysis or plotting.

Usage

```
trackFrequency(
  whe,
  hostCol = "host_id",
  timeCol = "timepoint",
  usePassedOnly = TRUE
)
```

Arguments

whe	A WithinHostExperiment with longitudinal samples from the same host.
hostCol	Character scalar. Column name in colData containing the host/patient identifier (default: "host_id").
timeCol	Character scalar. Column name in colData containing the time point (default: "timepoint"). It must hold numbers, dates, date-times, time differences or a factor whose levels are in time order; text is read as numbers when every value is one, and is otherwise an error, as in flagTemporalInconsistency. Samples whose time point is NA are left out, with a warning.
usePassedOnly	Logical. If TRUE (default), only use variants where qcPass == TRUE. With FALSE every call is returned and qc_passed marks those that failed.

Value

A DataFrame with columns:

host_id Host identifier.

variant_key Unique variant identifier (chrom:pos:alt).

timepoint Time point value.

frequency Alternative allele frequency at that time point.

qc_passed Logical. Whether the call passed QC at that time point; an NA qcPass counts as not passed (TRUE if no QC assay present).

Examples

```
gr <- GenomicRanges::GRanges("seg1",
    IRanges::IRanges(c(100, 200), width = 1))
S4Vectors::mcols(gr)$ref <- c("A", "C")
S4Vectors::mcols(gr)$alt <- c("T", "G")
freq <- matrix(c(0.05, 0.10, 0.15, 0.20), nrow = 2, ncol = 2)
whe <- WithinHostExperiment(
    assays = list(altFreq = freq),
    rowRanges = gr,
    colData = S4Vectors::DataFrame(
        sample_id = c("S1_t1", "S1_t2"),
        host_id = c("H1", "H1"),
        timepoint = c(1, 2)))
trackFrequency(whe)
```

`wattersonISNV` *Watterson's theta from iSNV count*

Description

Estimates Watterson's theta (Watterson 1975):

$$\hat{\theta}_W = S/(a_n \cdot L)$$

where S is the number of segregating sites, $a_n = \sum_{i=1}^{n-1} 1/i$ is the harmonic number, and L is the genome length, with n the number of sampled viral genomes.

Usage

```
wattersonISNV(nSites, genomeLength, meanDepth)
```

Arguments

`nSites`	Integer scalar. Number of segregating (iSNV) sites.
`genomeLength`	Integer scalar. Genome length in bp.
`meanDepth`	Numeric scalar. Sample size n: an effective number of sampled genomes, not raw read depth. See Details.

Details

As `tajimaD` sets out, raw read depth should **not** be passed as n for deep sequencing data: reads are technical replicates of a much smaller viral population, and a depth-sized n inflates a_n and makes θ_W correspondingly small. Pass an effective sample size, or a depth capped at one, as `calcNeutralityTests` does.

Value

Numeric scalar. Watterson's theta estimate.

References

Watterson GA (1975). On the number of segregating sites in genetical models without recombination. *Theor Popul Biol* 7:256-276. doi:10.1016/00405809(75)900209

Examples

```
## n is an effective sample size, not the raw depth
wattersonISNV(15, genomeLength = 13588, meanDepth = 100)
```

WHE-combine *Combine WithinHostExperiment objects by rows or columns*

Description

Methods for the combineRows and combineCols generics. combineRows combines objects vertically (adding variant sites); all objects must have the same samples (columns). Samples are matched by their column names: an object that holds them in another order is put in the first object's order, and one that holds other samples is an error naming them. The colData and the metadata are the first object's. combineCols combines objects horizontally (adding samples); all objects must have the same variant sites (rows). Sites are matched by sequence name, position, ref and alt: an object that holds them in another order is put in the first object's order, and one that holds other sites is an error counting them. Objects read from separate files usually hold different sites; read the files together with readWithinHost instead, which takes the union. QC logs and filters from all objects are merged.

Usage

```
## S4 method for signature 'WithinHostExperiment'
combineRows(x, ...)

## S4 method for signature 'WithinHostExperiment'
combineCols(x, ..., use.names = TRUE)
```

Arguments

x	A WithinHostExperiment.
...	Further WithinHostExperiment objects to combine.
use.names	Accepted for compatibility with the generic. Sites are matched by sequence name, position and alleles, not by row name.

Value

A WithinHostExperiment.

Examples

```
data(example_whe)
whe1 <- example_whe[1:10, ]
whe2 <- example_whe[11:nrow(example_whe), ]
merged <- combineRows(whe1, whe2)
nrow(merged) == nrow(example_whe)

whe_d <- example_whe[, 1]
whe_r <- example_whe[, 2]
merged <- combineCols(whe_d, whe_r)
ncol(merged)
```

WithinHostExperiment *Create a WithinHostExperiment from components*

Description

Constructs a WithinHostExperiment from assay matrices, genomic ranges, and sample metadata.

Usage

```
WithinHostExperiment(assays, rowRanges, colData, metadata = list())
```

Arguments

assays	Named list of matrices. Must include at least altFreq. If qcPass is not provided, it is automatically created as an all-TRUE matrix.
rowRanges	GRanges. Genomic coordinates for each variant site.
colData	DataFrame or data.frame. Sample metadata. Must contain a sample_id column.
metadata	Named list (optional). Additional metadata such as reference_genome, genome_length, calling_info.

Value

A WithinHostExperiment object.

Examples

```
library(GenomicRanges)
gr <- GRanges("seg1", IRanges::IRanges(c(100, 200, 300), width = 1))
whe <- WithinHostExperiment(
    assays = list(altFreq = matrix(c(0.1, 0.2, 0.3), ncol = 1)),
    rowRanges = gr,
    colData = S4Vectors::DataFrame(sample_id = "S1"))
whe
```

WithinHostExperiment-class *WithinHostExperiment: Container for within-host pathogen variant data*

Description

Extends RangedSummarizedExperiment-class to hold intra-host single nucleotide variant (iSNV) data with integrated quality control audit logging.

Subsetting preserves the qcLog and qcFilters slots. These slots are experiment-level metadata (not row/column-dependent) and are carried forward unchanged.

Usage

```
## S4 method for signature 'WithinHostExperiment'
show(object)

## S4 method for signature 'WithinHostExperiment,ANY,ANY,ANY'
x[i, j, ..., drop = TRUE]
```

Arguments

object	A WithinHostExperiment object.
x	A WithinHostExperiment.
i	Row indices (variant sites).
j	Column indices (samples).
...	Additional arguments passed to the parent method.
drop	Logical (ignored for SummarizedExperiment).

Value

A WithinHostExperiment.

Slots

qcLog DataFrame. Audit trail of QC filtering steps.

qcFilters list. History of ISNVFilter objects that have been applied. Validity checks each one, so a filter saved by an earlier version of the package, or changed with @<- into an invalid one, is reported, and the object cannot be subset or combined until the filter is rebuilt with ISNVFilter and stored with qcFilters<-.

Standard assays

altCount Integer matrix. Alternative allele read counts.

refCount Integer matrix. Reference allele read counts.

totalDepth Integer matrix. Total read depth.

altFreq Numeric matrix. Alt allele frequencies (0-1).

qcPass Logical matrix. TRUE = passed QC, FALSE = flagged. Initialized to all TRUE; updated by flagISNV. No package function writes NA, but one set by hand counts as not passed wherever an analysis uses only passing calls.

Assays must be ordinary R matrices. Out-of-memory backends such as DelayedArray or HDF5Array are not supported: the validity method refuses a qcPass assay that is not a base logical matrix, and flagISNV combines assays with base R operators.

Expected colData columns

sample_id Character. Unique sample identifier (required).

host_id Character. Host/patient identifier (optional).

replicate_group Character. Technical replicate group identifier (optional).

replicate_id Character. Replicate number within group (optional).

role Character. Sample role: "donor", "recipient", or "independent" (optional).

pair_id Character. Transmission pair identifier (optional).

See Also

readWithinHost and readWithinHostTable for constructing from files. flagISNV for quality control. passedISNV for extracting QC-passed variants. transmissionPairs for extracting donor-recipient pairs.

Examples

```
data(example_whe)
example_whe
```

WithinHostSFS-class *WithinHostSFS: Site Frequency Spectrum for within-host variants*

Description

An S4 class representing the site frequency spectrum (SFS) of within-host variants from deep sequencing data. The SFS bins variant allele frequencies into discrete categories and stores metadata needed for population-genetic inference.

Usage

```
## S4 method for signature 'WithinHostSFS'
show(object)
```

Arguments

object	A WithinHostSFS object.

Value

An object of class WithinHostSFS, normally created by buildSFS; the show method prints a summary of the spectrum and returns NULL invisibly.

Slots

counts Integer vector. Number of iSNVs in each frequency bin. Length equals length(breaks) - 1.

breaks Numeric vector. Bin boundaries. Bins are right-closed, and the lowest break is included in the first bin. Default: 20 equal bins from the detection threshold to the maximum frequency.

folded Logical scalar. If TRUE, the SFS is folded (minor allele frequency, 0 to 0.5); if FALSE, unfolded: the frequency of the alternative allele, 0 to 1.

polarization Character scalar: which allele is treated as ancestral. "none" for a folded spectrum, which carries no polarisation; "reference" for an unfolded spectrum built by buildSFS, whose frequencies are those of the alternative allele, so the reference allele stands in for the ancestral one; "outgroup" for a spectrum whose frequencies are of alleles derived relative to an outgroup, which this package does not build but accepts. Statistics that need derived-allele frequencies, such as Fay and Wu's H, refuse "none".

sampleId Character scalar. Sample identifier.

genomeLength Integer scalar. Reference genome length.

`nSites` Integer scalar. Number of variant sites counted in the spectrum, that is, with a frequency between the lowest and the highest break; `buildSFS` counts only segregating sites, whose frequency lies strictly between 0 and 1.

`meanDepth` Numeric scalar. Typical sequencing depth; `buildSFS` stores the median depth across sites.

`threshold` Numeric scalar. Detection threshold applied (e.g., 0.03). Sites below this are unobservable.

`corrected` Logical scalar. Whether ascertainment bias correction has been applied.

`maxFreq` Numeric scalar, above 0 and at most 1 (default 1). The highest alternative allele frequency a call could have kept through QC: `buildSFS` stores the smallest `maxFreq` among the filters applied with `flagISNV` when it uses QC-passed calls only, and 1 otherwise. Below 1, the spectrum holds no call above it, so the high-frequency classes are empty by construction; `neutralityFromSFS` refuses such an unfolded spectrum, and `compareSFS` compares only spectra with the same value.

See Also

`buildSFS` for constructing from a `WithinHostExperiment`, `neutralityFromSFS` for computing all neutrality tests from the SFS.

Examples

```
data(example_whe)
## a folded spectrum of the first sample, in ten bins from 3% to 50%;
## the summary reports its polarisation, "none" for a folded spectrum,
## and the genome length, read from seqlengths(example_whe)
sfs <- buildSFS(example_whe, sampleIdx = 1L, nBins = 10L)
sfs
```

Index

∗ **package**
WithinHostExperiment-package, 3
.ConcordanceRule
*(*ConcordanceRule-class*)*, 27
.ISNVFilter *(*ISNVFilter-class*)*, 50
.WithinHostExperiment
*(*WithinHostExperiment-class*)*, 76
.WithinHostSFS *(*WithinHostSFS-class*)*, 78
[,WithinHostExperiment,ANY,ANY,ANY-method
*(*WithinHostExperiment-class*)*, 76

adaptiveFilter, *4*, 6
annotateCodonChange, *4*, 7, *11*, *18*, *60*, *61*
annotateCodonChange,WithinHostExperiment,ANY,character-method
*(*annotateCodonChange*)*, 7
annotateCodonChange,WithinHostExperiment,ANY,XString-method
*(*annotateCodonChange*)*, 7
annotateCodonChange,WithinHostExperiment,character,DNAStringSet-method
*(*annotateCodonChange*)*, 7
annotateCodonChange,WithinHostExperiment,GRanges,DNAStringSet-method
*(*annotateCodonChange*)*, 7
annotateCodonChange,WithinHostExperiment,GRangesList,DNAStringSet-method
*(*annotateCodonChange*)*, 7
annotateCodonChange,WithinHostExperiment,TxDb,DNAStringSet-method
*(*annotateCodonChange*)*, 7
annotateGenes, *4*, *9*, *10*, 10
annotateGenes,WithinHostExperiment,GRanges-method
*(*annotateGenes*)*, 10
annotateGenes,WithinHostExperiment,TxDb-method
*(*annotateGenes*)*, 10
asViralBottleneckInput, *4*, 12, *39*, *64*

benchmarkCallers, *4*, 13
buildSFS, *4*, 14, *15*, *52*, *78*, *79*
buildSFSList, *4*, 15

calcCallerConcordance, *4*, *13*, *14*, 16
calcDiversity, *4*, *15*, *16*, 18, *23–25*
calcDiversity,WithinHostExperiment-method
*(*calcDiversity*)*, 18
calcNeutralityTests, *4*, *14*, *18*, 19, *52*, *72*, *74*
calcNeutralityTests,WithinHostExperiment-method
*(*calcNeutralityTests*)*, 19
calcSharedVariants, *4*, *13*, 21
chao1ISNV, *4*, 22
checkDiversityPlausibility, *4*, 23
chisq.test, *26*
coerce,VCF,WithinHostExperiment-method
*(*coerce-WHE*)*, 24
coerce,VRanges,WithinHostExperiment-method
*(*coerce-WHE*)*, 24
coerce,WithinHostExperiment,data.frame-method
*(*coerce-WHE*)*, 24
coerce,WithinHostExperiment,VRanges-method
*(*coerce-WHE*)*, 24
coerce-WHE, 24
combineCols, *75*
combineCols,WithinHostExperiment-method
*(*WHE-combine*)*, 75
combineRows, *75*
combineRows,WithinHostExperiment-method
*(*WHE-combine*)*, 75
compareDiversity, *4*, 25
compareSFS, *4*, 25, *32*, *79*
ConcordanceRule, 26, *27*, *28*, *42*, *44*, *49*, *50*
ConcordanceRule-class, 27
consensusISNV, *4*, 28
consensusSequences, *4*, 30
consensusSequences,WithinHostExperiment,character-method
*(*consensusSequences*)*, 30
consensusSequences,WithinHostExperiment,DNAStringSet-method
*(*consensusSequences*)*, 30
consensusSequences,WithinHostExperiment,XString-method
*(*consensusSequences*)*, 30
correctSFSBias, *4*, 32

detectCrossContamination, *4*, 33
detectEmergingVariants, *4*, 35

estimateSelectionCoefficient, *4*, 36
exactBottleneck, *4*, *13*, 37, *64*
example_whe, 39
exportPairFrequencies, *4*, *21*, 40

fitDiversityModel, *4*, 41

flagISNV, *4*, *7*, *15, 16*, *21*, *27–29*, *33*, *35*, 41, *44–46*, *49–52*, *66*, *68*, *77–79*
flagISNV,WithinHostExperiment-method *(*flagISNV*)*, 41
flagPrimerSites, *4*, 43
flagReplicateDiscordance, *4*, *27, 28*, *42*, 44
flagStrandBiasSCI, *4*, 45, *67, 68*
flagTemporalInconsistency, *4*, 46, *73*
fusFs, *4*, *20*, *32*, 48, *53*

ISNVFilter, *4*, *6, 7*, *15, 16*, *26–28*, *42*, *49*, 49, *50–52*, *63*, *77*
ISNVFilter-class, 50

lm, *41*

multitestCorrection, 51

neutralityFromSFS, *4*, *15, 16*, *32*, 52, *79*

p.adjust, *51*
passedISNV, *4*, *42*, *78*
passedISNV *(*qcLog*)*, 62
passedISNV,WithinHostExperiment-method *(*qcLog*)*, 62
permutationTest, 53
piISNV, *4*, 54
plotFrequencySpectrum, *4*, 55
plotFrequencyTrajectory, *4*, 56
plotPairScatter, *4*, 57
plotQCDashboard, *4*, 58
plotSFS, *4*, 58
pNpSWithinHost, *4*, *9*, *30, 31*, 59
pNpSWithinHost,WithinHostExperiment,ANY,character-method *(*pNpSWithinHost*)*, 59
pNpSWithinHost,WithinHostExperiment,ANY,XString-method *(*pNpSWithinHost*)*, 59
pNpSWithinHost,WithinHostExperiment,character,DNAStringSet-method *(*pNpSWithinHost*)*, 59
pNpSWithinHost,WithinHostExperiment,GRanges,DNAStringSet-method *(*pNpSWithinHost*)*, 59
pNpSWithinHost,WithinHostExperiment,GRangesList,DNAStringSet-method *(*pNpSWithinHost*)*, 59
pNpSWithinHost,WithinHostExperiment,TxDb,DNAStringSet-method *(*pNpSWithinHost*)*, 59

qcFilters *(*qcLog*)*, 62
qcFilters,WithinHostExperiment-method *(*qcLog*)*, 62
qcFilters<- *(*qcLog*)*, 62
qcFilters<-,WithinHostExperiment-method *(*qcLog*)*, 62
qcLog, 62
qcLog,WithinHostExperiment-method *(*qcLog*)*, 62
qcLog<- *(*qcLog*)*, 62
qcLog<-,WithinHostExperiment-method *(*qcLog*)*, 62
qcSummary, *4*, *42*, *58*, 63
quickBottleneck, *4*, *13*, *39*, 64

readWithinHost, *4*, *18*, *24*, *40*, 65, *68*, *75*, *78*
readWithinHostTable, *4*, *21*, *66*, 67, *78*

sciStrandBias, *4*, *45, 46*, 69
shannonISNV, *4*, 70
show,ConcordanceRule-method *(*ConcordanceRule-class*)*, 27
show,ISNVFilter-method *(*ISNVFilter-class*)*, 50
show,WithinHostExperiment-method *(*WithinHostExperiment-class*)*, 76
show,WithinHostSFS-method *(*WithinHostSFS-class*)*, 78
simpsonISNV, *4*, 70

tajimaD, *4*, *18*, *20*, *48*, *52–54*, 71, *74*
trackFrequency, *4*, *35*, *56*, 72
transmissionPairs, *4*, *78*
transmissionPairs *(*qcLog*)*, 62
transmissionPairs,WithinHostExperiment-method *(*qcLog*)*, 62

VRanges, *24*

wattersonISNV, *4*, 74
wNE-combine, 75
WithinHostExperiment, *7, 8*, *11–21*, *24, 25*, *28–30*, *33*, *35*, *38–47*, *55–58*, *60*, *62–65*, *67, 68*, *73*, *75, 76*, 76, *77*
WithinHostExperiment-class, 76
WithinHostExperiment-package, 3
WithinHostSFS, *14–16*, *25*, *32*, *52*, *58*
WithinHostSFS *(*WithinHostSFS-class*)*, 78
WithinHostSFS-class, 78