

Package: WithinHostExperiment (via r-universe)

September 18, 2026

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

Version 0.99.0

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 (>= 4.5.0)

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

Suggests ggplot2 (>= 3.4.0), patchwork, testthat (>= 3.1.5), knitr, rmarkdown, BiocCheck, covr

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

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

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

Encoding UTF-8

LazyData false

RoxygenNote 7.3.3

VignetteBuilder knitr

Config/testthat/edition 3

Collate 'AllClasses.R' 'AllGenerics.R'

'WithinHostExperiment-package.R' 'adaptive-filter.R'
 'methods.R' 'annotate-gff.R' 'annotate-codon.R' 'bridge.R'
 'bottleneck-exact.R' 'bottleneck-quick.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'

Config/pak/sysreqs

make libbz2-dev liblzma-dev libpng-dev libxml2-dev libssl-dev xz-utils zlib1g-dev

Repository <https://biocstaging.r-universe.dev>

Date/Publication 2026-09-17 00:13:03 UTC

RemoteUrl <https://github.com/BiocStaging/WithinHostExperiment>

RemoteRef HEAD

RemoteSha 28a1a5c5a77399ceaa85502f2893d28071b1457e

Contents

WithinHostExperiment-package	3
adaptiveFilter	6
annotateCodonChange	7
annotateFromGFF	9
asViralBottleneckInput	10
benchmarkCallers	11
buildSFS	12
buildSFSList	13
calcCallerConcordance	14
calcDiversity	15
calcNeutralityTests	16
calcSharedVariants	17
chao1ISNV	18
coerce-WHE	19
compareDiversity	19
compareSFS	20
consensusISNV	21
correctSFSBias	22
detectCrossContamination	23
detectEmergingVariants	25
dndsWithinHost	26
estimateSelectionCoefficient	28
exactBottleneck	29
example_whe	31
exportPairFrequencies	31
fitDiversityModel	32
flagISNV	33

flagPrimerSites	34
flagReplicateDiscordance	35
flagStrandBiasSCI	36
flagTemporalInconsistency	38
fusFs	39
ISNVFilter	40
ISNVFilter-class	41
multitestCorrection	43
neutralityFromSFS	44
permutationTest	45
piISNV	46
plotFrequencySpectrum	47
plotFrequencyTrajectory	48
plotPairScatter	49
plotQCDashboard	50
plotSFS	50
qcLog	51
qcSummary	52
quickBottleneck	53
readWithinHost	54
readWithinHostTable	55
sciStrandBias	56
shannonISNV	57
simpsonISNV	58
tajimaD	58
trackFrequency	60
validateDiversity	61
wattersonISNV	62
WHE-combine	63
WithinHostExperiment	64
WithinHostExperiment-class	64
WithinHostSFS-class	66
Index	68

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 `annotateFromGFF` (positional gene annotation from GFF3 files), `annotateCodonChange` (CDS-aware codon translation and amino acid change prediction from GFF3 + reference FASTA)

Selection analysis `dndsWithinHost`, `estimateSelectionCoefficient`

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`, `validateDiversity`

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 (super-infection) 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. *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 *Annotate codon changes from GFF3 CDS + reference FASTA*

Description

For each SNV in a [WithinHostExperiment](#), determines the affected codon and amino acid change using CDS coordinates from a GFF3 file and the reference genome sequence from a FASTA file. Adds GFF_FEATURE, REF_CODON, ALT_CODON, REF_AA, ALT_AA, and AA_CLASS (Synonymous / Non-synonymous / Nonsense) columns to mcols(rowRanges).

Usage

```
annotateCodonChange(whe, gff, refFasta, nameCol = "gene")
```

annotateFromGFF	<i>Annotate variant sites from a GFF3 gene model</i>
-----------------	--

Description

Reads a GFF3 file and adds gene/feature annotations to `mcols(rowRanges)` of a [WithinHostExperiment](#). This automates the annotation step required by [dndsWithinHost](#), which expects REF_AA and ALT_AA columns.

Usage

```
annotateFromGFF(whe, gff, featureType = "gene", nameCol = "Name")
```

Arguments

<code>whe</code>	A WithinHostExperiment object.
<code>gff</code>	Character scalar. Path to a GFF3 file.
<code>featureType</code>	Character scalar. Which GFF feature type to extract (default: "gene"). Common choices: "gene", "CDS", "mRNA".
<code>nameCol</code>	Character scalar. GFF3 attribute to use as the feature name (default: "Name"). Falls back to "gene", then "ID" if not found.

Details

The function overlaps variant positions with GFF features of the specified type (default: "gene") and adds a GFF_FEATURE column containing the gene name or ID. For coding-region annotation (amino acid changes), a reference FASTA is additionally required; this function provides the positional overlap only.

The GFF3 file is parsed with base R (no external dependency). Only lines matching `featureType` in column 3 are used. Attribute parsing follows the GFF3 specification (key=value pairs separated by semicolons).

For full codon-aware annotation (predicting amino acid changes), use dedicated tools such as SnpEff or ANNOVAR upstream, then import the annotated VCF via [readWithinHost](#).

Value

A [WithinHostExperiment](#) with a GFF_FEATURE column added to `mcols(rowRanges)`.

Examples

```
## Create a minimal GFF3 for demonstration
gff_file <- tempfile(fileext = ".gff3")
writeLines(c(
  "##gff-version 3",
  "test_segment\t.\tgene\t1\t500\t.\t+\t.\tName=geneA",
  "test_segment\t.\tgene\t501\t1000\t.\t+\t.\tName=geneB"
), gff_file)
```

```
vcf <- system.file("extdata", "test_donor.vcf",
  package = "WithinHostExperiment")
whe <- readWithinHost(vcf,
  colData = S4Vectors::DataFrame(sample_id = "donor"),
  caller = "ivar")
whe <- annotateFromGFF(whe, gff_file)
S4Vectors::mcols(SummarizedExperiment::rowRanges(whe))$GFF_FEATURE
```

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

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	<i>Benchmark variant callers against truth or each other</i>
------------------	--

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

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

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.

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.

Usage

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

Arguments

whe	A WithinHostExperiment .
sampleIdx	Integer or character. Which sample (column) to use. Default: 1.
nBins	Integer. Number of frequency bins (default: 20).
fold	Logical. If TRUE, fold the SFS to minor allele frequency (0 to 0.5). Default: TRUE.
usePassedOnly	Logical. If TRUE, only use QC-passed variants. Default: TRUE.
threshold	Numeric. Detection threshold (default: 0.03). The first bin starts at this value, not 0.
genomeLength	Integer (optional). Reference genome length.

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

<code>whe</code>	A <code>WithinHostExperiment</code> .
<code>nBins</code>	Integer. Number of frequency bins (default: 20).
<code>fold</code>	Logical. If TRUE, fold the SFS to minor allele frequency (0 to 0.5). Default: TRUE.
<code>usePassedOnly</code>	Logical. If TRUE, only use QC-passed variants. Default: TRUE.
<code>threshold</code>	Numeric. Detection threshold (default: 0.03). The first bin starts at this value, not 0.
<code>genomeLength</code>	Integer (optional). Reference genome length.

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).

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	<i>Compute within-host diversity indices</i>
---------------	--

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"),
  genomeLength = NULL,
  usePassedOnly = TRUE,
  ...
)

```

Arguments

<code>x</code>	A WithinHostExperiment .
<code>...</code>	Additional arguments (currently unused).
<code>indices</code>	Character vector. Which indices to compute: "shannon", "pi", "watterson", "richness", "simpson", "chao1" (default: all six).
<code>genomeLength</code>	Integer (optional). Required for "pi" and "watterson". 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.
<code>usePassedOnly</code>	Logical. If TRUE (default), only use variants where <code>qcPass == TRUE</code> .

Value

A `DataFrame` of diversity metrics.

A `DataFrame` with one row per sample and columns for each 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")
whe <- readWithinHost(vcf,
  colData = S4Vectors::DataFrame(sample_id = "donor"),
  caller = "ivar")
calcDiversity(whe, genomeLength = 1000L)
```

calcNeutralityTests	<i>Compute neutrality test statistics</i>
---------------------	---

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,
  genomeLength = NULL,
  indices = c("tajimaD", "fusFs"),
  usePassedOnly = TRUE,
  ...
)
```

Arguments

x	A WithinHostExperiment .
...	Additional arguments (currently unused).
genomeLength	Integer scalar. Reference genome length in bp. Required.
indices	Character vector. Which tests to compute: "tajimaD", "fusFs" (default: both).
usePassedOnly	Logical. If TRUE (default), only use variants where qcPass == TRUE.

Value

A `DataFrame` of neutrality statistics.

A `DataFrame` with one row per sample and columns `sample_id` plus one column per requested test.

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, genomeLength = 1000L)
```

calcSharedVariants	<i>Calculate shared variant statistics</i>
--------------------	--

Description

Calculate shared variant statistics

Usage

```
calcSharedVariants(whe, pairId)
```

Arguments

whe	A <code>WithinHostExperiment</code> .
pairId	Character. Which pair to analyse.

Value

A named list: `n_donor_only`, `n_recipient_only`, `n_shared`, `n_total`, `sharing_rate`.

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

<code>counts</code>	Integer vector. Alternative allele read counts (<code>altCount</code>) for each variant.
<code>detected</code>	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)
```

coerce-WHE*Coerce WithinHostExperiment to data.frame or VRanges*

Description

Coerce WithinHostExperiment to data.frame or VRanges

Value

A data.frame or [VRanges](#) object containing the variant data from the WithinHostExperiment object.

Examples

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

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

compareDiversity*Compare diversity between sample groups*

Description

Compare diversity between sample groups

Usage

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

Arguments

whe	A WithinHostExperiment .
groupBy	Character. Column name in colData to group by.
index	Character. Which index to compare (default: "shannon").
genomeLength	Integer (optional). Required if index is "pi" or "watterson".

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., before vs. after QC, or two timepoints).

Usage

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

Arguments

sfs1 A [WithinHostSFS](#) object.
sfs2 A [WithinHostSFS](#) object (same binning).

Details

The test compares raw variant counts, so apply it to uncorrected spectra: the counts of a bias-corrected spectrum are rescaled and no longer multinomial. Bins where both spectra are empty are dropped, and when only two bins remain [chisq.test](#) applies Yates' continuity correction. 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.

chisq_p Chi-squared p-value.

df Degrees of freedom.

min_expected Smallest expected count under homogeneity, or NA when 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)
```

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).

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. `mcols(rowRanges)` includes:

n_callers Number of callers detecting the site.
caller_names Comma-separated caller names.
freq_spread Range of allele frequencies across callers.

The `altFreq` assay contains the mean frequency across callers for each consensus site. It is the only assay of the result: read counts, depths and `qcPass` are not carried over, so run [flagISNV](#) on the individual callers' objects rather than on the consensus.

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
```

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 <code>WithinHostSFS</code> 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.

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

<code>whe</code>	A <code>WithinHostExperiment</code> with multiple samples.
<code>runCol</code>	Character scalar (optional). Column name in <code>colData</code> identifying the sequencing run/batch. If <code>NULL</code> , all samples are compared against each other.
<code>sourceMinFreq</code>	Numeric scalar. Minimum frequency in the "source" sample for a variant to be considered a potential contaminant origin (default: 0.20).
<code>sinkMaxFreq</code>	Numeric scalar. Maximum frequency in the "sink" sample for a variant to be flagged as potential contamination (default: 0.05).
<code>sinkMinFreq</code>	Numeric scalar. Minimum frequency in the "sink" sample (default: 0.005). Variants below this are likely noise, not contamination.
<code>usePassedOnly</code>	Logical. If <code>TRUE</code> (default for source), only consider QC-passed variants in the source sample.

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.

Usage

```

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

```

Arguments

<code>whe</code>	A WithinHostExperiment with longitudinal samples.
<code>hostCol</code>	Character scalar. Column name in <code>colData</code> for host identifier (default: "host_id").
<code>timeCol</code>	Character scalar. Column name in <code>colData</code> for time point (default: "timepoint").
<code>freqThreshold</code>	Numeric scalar. Frequency threshold for emergence (default: 0.05).
<code>minIncrease</code>	Numeric scalar. Minimum frequency increase between consecutive time points (default: 0.05).
<code>usePassedOnly</code>	Logical. If TRUE (default), only use QC-passed variants.

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.

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)
```

dndsWithinHost

Within-host dN/dS ratio

Description

Estimates the ratio of nonsynonymous to synonymous divergence among the iSNVs of each sample with the counting method of Nei and Gojobori (1986). The numbers of synonymous (S) and nonsynonymous (N) sites are counted over the complete codons of every CDS in gff, using the reference sequence in refFasta and the standard genetic code. With S_d synonymous and N_d nonsynonymous iSNVs present in a sample,

$$p_S = S_d/S, \quad p_N = N_d/N,$$

each proportion is corrected for multiple hits with Jukes and Cantor (1969), $d = -\frac{3}{4} \log(1 - \frac{4}{3}p)$, and d_N/d_S is their ratio. Under neutrality the expected ratio is 1; values below 1 suggest purifying selection and values above 1 diversifying selection, but within-host ratios computed from few iSNVs are highly variable and should be read with their counts.

Usage

```
dndsWithinHost(
  whe,
  gff,
  refFasta,
  gffFeatureCol = "GFF_FEATURE",
  refAACol = "REF_AA",
  altAACol = "ALT_AA",
  usePassedOnly = TRUE,
  nameCol = "gene"
)
```

Arguments

whe	A WithinHostExperiment annotated with annotateCodonChange .
gff	Character scalar. Path to the GFF3 file with CDS features used for the annotation.
refFasta	Character scalar. Path to the reference genome FASTA used for the annotation.
gffFeatureCol	Character scalar. Column name in <code>mcols(rowRanges)</code> containing the gene annotation (default: "GFF_FEATURE").
refAACol	Character scalar. Column name in <code>mcols(rowRanges)</code> for the reference amino acid (default: "REF_AA").
altAACol	Character scalar. Column name in <code>mcols(rowRanges)</code> for the alternative amino acid (default: "ALT_AA").
usePassedOnly	Logical. If TRUE (default), only use variants where <code>qcPass == TRUE</code> .
nameCol	Character scalar. GFF3 attribute holding the gene name (default: "gene"), as in annotateCodonChange .

Details

Each iSNV present in a sample (non-missing, non-zero frequency, and passing QC when `usePassedOnly = TRUE`) is counted once, irrespective of its frequency. Amino acid classes come from [annotateCodonChange](#), which should be run with the same `gff` and `refFasta`. Variants outside a CDS, or without an amino acid annotation, are not counted. Nonsense changes count as nonsynonymous. Codons shared by overlapping CDS features of the same gene are counted once.

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 CDS features.

pS, pN Proportions S_d/S and N_d/N .

dNdS Jukes-Cantor corrected d_N/d_S (NA when d_S is zero or a proportion reaches 0.75).

gene, gene_nS, gene_nN, gene_S_sites, gene_N_sites, gene_dNdS 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

Jukes TH, Cantor CR (1969). Evolution of protein molecules. In Munro HN (ed.), *Mammalian Protein Metabolism*, pp. 21-132. Academic Press, New York.

See Also

[annotateCodonChange](#)

Examples

```
ref <- tempfile(fileext = ".fa")
writeLines(c(">seg1", "ATGCTGAAAGGGTAA"), ref)
gff <- tempfile(fileext = ".gff3")
writeLines(c("##gff-version 3",
  "seg1\t.\tCDS\t1\t15\t.\t+\t0\tgene=geneA"), gff)
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"))
whe <- annotateCodonChange(whe, gff, ref)
dndsWithinHost(whe, gff = gff, refFasta = ref)
```

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:

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

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

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](https://doi.org/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	<i>Exact bottleneck size estimation</i>
-----------------	---

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.

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.

presence_absence Presence-absence likelihood using $P(\text{detect} | Nb, \nu) = 1 - (1 - \nu)^{Nb}$.

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).
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).

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.

method Character. Method used.

n_variants Integer. Number of donor variants used.

pair_id Character. Pair identifier.

References

Sobel Leonard A et al. (2017). Transmission Bottleneck Size Estimation from Pathogen Deep-Sequencing Data. *J Virol* 91:e00171-17. doi:[10.1128/JVI.0017117](https://doi.org/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.

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).

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	<i>Fit a linear model to diversity data</i>
-------------------	---

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

<code>x</code>	A WithinHostExperiment object.
<code>filter</code>	An ISNVFilter object (default: <code>ISNVFilter()</code>).
<code>...</code>	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

The minBaseQual, minMapQual and replicateConc slots of the filter are recorded for provenance but not applied here; a non-default value raises a warning. Use [flagReplicateDiscordance](#) for replicate concordance.

Value

The filtered object (same class as input).

A [WithinHostExperiment](#) with updated qcPass assay and qcLog.

See Also

[ISNVFilter](#) for filter configuration, [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 <code>rtracklayer::import.bed()</code> .
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.

Details

For each replicate group, the function compares altFreq between replicates and flags sites where the difference exceeds freqTolerance. 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 > \text{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.

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,
  coldData = 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").
minTimepoints	Integer scalar. Minimum number of timepoints at which an iSNV must be detected to be considered persistent (default: 2).
minConsecutive	Integer scalar. Minimum number of consecutive timepoints required. 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.

Details

This check applies only where a host was sampled more than once. 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 F_s (Fu 1997), a test of neutrality based on the probability of observing at least the observed number of alleles 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 alleles greater than or equal to the observed value, given $\theta = \hat{\pi}$. A large negative F_s indicates an excess of alleles (rare variants), consistent with population expansion or genetic hitchhiking.

Usage

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

Arguments

<code>S</code>	Integer scalar. Number of segregating sites (alleles minus 1).
<code>n</code>	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.
<code>theta_pi</code>	Numeric scalar. Theta estimated from nucleotide diversity ($\hat{\pi} \times L$).

Details

The probability is approximated using Ewens' sampling formula for the expected number of alleles.

Value

Numeric scalar. Fu's F_s , or NA with a warning when $S + 1 > n$: 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,
  minBaseQual = 20L,
  minMapQual = 20L,
  replicateConc = FALSE
)
```

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).
minBaseQual	Integer. Minimum base quality (default: 20). Recorded for provenance; flagISNV does not apply it.
minMapQual	Integer. Minimum mapping quality (default: 20). Recorded for provenance; flagISNV does not apply it.
replicateConc	Logical. Require replicate concordance (default: FALSE). Recorded for provenance; apply it with flagReplicateDiscordance .

Value

An [ISNVFilter](#) object.

Examples

```
ISNVFilter()
ISNVFilter(minDepth = 500L, minFreq = 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).

`minBaseQual` Integer scalar. Minimum mean base quality (default: 20). Recorded for provenance only: `flagISNV` does not apply it, because the object model has no per-site quality assay. Filter on quality in the variant caller instead.

`minMapQual` Integer scalar. Minimum mean mapping quality (default: 20). Recorded for provenance only, as for `minBaseQual`.

`replicateConc` Logical scalar. Whether replicate concordance is required (default: `FALSE`). Recorded for provenance only; apply it with `flagReplicateDiscordance`.

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	<i>Multiple testing correction for p-values</i>
---------------------	---

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	<i>Compute all neutrality statistics from a WithinHostSFS</i>
-------------------	---

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: <code>min(100, meanDepth)</code> 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 is defined as:

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

where $\hat{\theta}_H = \frac{2}{n(n-1)} \sum_i i^2 \xi_i$ and ξ_i is the count of sites with i derived alleles. A negative H indicates an excess of high-frequency variants.

Value

A named list with components:

- S** Number of segregating sites.
- n** Effective sample size used.
- pi** Nucleotide diversity (per site).
- theta_W** Watterson's theta.
- theta_H** Fay and Wu's theta_H.
- tajimaD** Tajima's D.
- fusFs** Fu's Fs.
- fayWuH** Fay and Wu's H.
- sfs_corrected** Whether the input SFS was bias-corrected.

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	<i>Permutation test for two-group comparison</i>
-----------------	--

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 (π) 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.
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.

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").
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	<i>Plot donor-recipient frequency scatter</i>
-----------------	---

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	<i>QC dashboard plot</i>
-----------------	--------------------------

Description

Creates a publication-quality four-panel dashboard showing frequency distribution, read depth, QC filtering waterfall, and per-sample pass/fail summary.

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	<i>Plot a within-host site frequency spectrum</i>
---------	---

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)
}
```

 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)
```

Description

Produces per-sample summary statistics from a [WithinHostExperiment](#).

Usage

```
qcSummary(whe)
```

Arguments

whe A [WithinHostExperiment](#).

Value

A `DataFrame` with columns: `sample_id`, `total_sites`, `passed_sites`, `flagged_sites`, `pass_rate`, `median_depth`, `median_freq`.

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).
threshold Numeric. Variant calling threshold (default: 0.03).

Value

A named list: `Nb` (MLE), `ci` (95% CI), `loglik`, `n_variants`, `pair_id`.

References

Sobel Leonard A et al. (2017). Transmission Bottleneck Size Estimation from Pathogen Deep-Sequencing Data. *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	<i>Import iSNV data from VCF files</i>
----------------	--

Description

Reads one or more VCF files produced by variant callers (LoFreq, Freebayes, or generic) and constructs a [WithinHostExperiment](#).

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", "auto" (default), or "generic". See Details for what "ivar" means.
genome	Character scalar (optional). Path to reference FASTA (stored in metadata, not currently used for annotation).

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.

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")`.

Records with several alternative alleles are split into one site per allele. Per-allele values (INFO AF; FORMAT AO, AF, ALT_DP, ALT_FREQ and ALT_QUAL) are taken for that allele, and per-site values such as total depth and reference counts are shared.

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
```

<code>readWithinHostTable</code>	<i>Import iSNV data from tabular files</i>
----------------------------------	--

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

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

Value

A `WithinHostExperiment`.

Examples

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

sciStrandBias	<i>Strand Coverage Imbalance (SCI) score</i>
---------------	--

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.
- 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:

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

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.

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 (values in (0, 1); 0, 1, 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](https://doi.org/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:

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

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 (values in (0, 1); 0, 1, 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

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 toward 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 ($n = \min(\text{depth}, 100)$).

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.

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	<i>Track variant frequency trajectories over time</i>
----------------	---

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").
usePassedOnly	Logical. If TRUE (default), only use variants where qcPass == TRUE.

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 variant passed QC at that time point (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)

```

validateDiversity	<i>Check diversity estimates against plausibility ranges</i>
-------------------	--

Description

Compares observed diversity metrics against wide plausibility ranges for common pathogens, as a sanity check on units and orders of magnitude rather than a reference interval. The bounds are the package's own heuristics, not estimates from a specific study, and are deliberately generous: π is compared on a per-site scale and Shannon entropy on the scale returned by [calcDiversity](#). A value inside the range is not evidence that an estimate is correct.

Usage

```

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

```

Arguments

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

Value

A `DataFrame` with columns:

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.

Examples

```

div <- S4Vectors::DataFrame(pi = 1e-4, shannon = 5.0)
validateDiversity(div, pathogen = "sars_cov_2")

```

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). `combineCols` combines objects horizontally (adding samples); all objects must have the same variant sites (rows). 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

<code>x</code>	A <code>WithinHostExperiment</code> .
<code>...</code>	Further <code>WithinHostExperiment</code> objects to combine.
<code>use.names</code>	Accepted for compatibility with the generic. Sites are combined in the order given, not matched by 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.

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](#).

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	<i>WithinHostSFS: Site Frequency Spectrum for within-host variants</i>
---------------------	--

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); if FALSE, unfolded (derived allele frequency).

sampleId Character scalar. Sample identifier.

genomeLength Integer scalar. Reference genome length.

nSites Integer scalar. Total number of variant sites used to build the SFS.

`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.

See Also

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

Index

- * **package**
 - WithinHostExperiment-package, 3
- .ISNVFilter (ISNVFilter-class), 41
- .WithinHostExperiment
 - (WithinHostExperiment-class), 64
- .WithinHostSFS (WithinHostSFS-class), 66
- [,WithinHostExperiment,ANY,ANY,ANY-method
 - (WithinHostExperiment-class), 64
- adaptiveFilter, 4, 6
- annotateCodonChange, 4, 7, 27, 28
- annotateFromGFF, 4, 9
- asViralBottleneckInput, 4, 10, 53
- benchmarkCallers, 4, 11
- buildSFS, 4, 12, 13, 66, 67
- buildSFSList, 4, 13
- calcCallerConcordance, 4, 11, 14
- calcDiversity, 4, 15, 61
- calcDiversity,WithinHostExperiment-method
 - (calcDiversity), 15
- calcNeutralityTests, 4, 15, 16, 59, 62
- calcNeutralityTests,WithinHostExperiment-method
 - (calcNeutralityTests), 16
- calcSharedVariants, 4, 17
- chao1ISNV, 4, 18
- chisq.test, 20
- coerce,VRanges,WithinHostExperiment-method
 - (coerce-WHE), 19
- coerce,WithinHostExperiment,data.frame-method
 - (coerce-WHE), 19
- coerce,WithinHostExperiment,VRanges-method
 - (coerce-WHE), 19
- coerce-WHE, 19
- combineCols, 63
- combineCols,WithinHostExperiment-method
 - (WHE-combine), 63
- combineRows, 63
- combineRows,WithinHostExperiment-method
 - (WHE-combine), 63
- compareDiversity, 4, 19
- compareSFS, 4, 20, 23
- consensusISNV, 4, 21
- correctSFSBias, 4, 22
- detectCrossContamination, 4, 23
- detectEmergingVariants, 4, 25
- dndsWithinHost, 4, 9, 26
- estimateSelectionCoefficient, 4, 5, 28
- exactBottleneck, 4, 29
- example_whe, 31
- exportPairFrequencies, 4, 31
- fitDiversityModel, 4, 32
- flagISNV, 4, 7, 21, 33, 36, 37, 40–43, 65, 66
- flagISNV,WithinHostExperiment-method
 - (flagISNV), 33
- flagPrimerSites, 4, 34
- flagReplicateDiscordance, 4, 34, 35, 41, 42
- flagStrandBiasSCI, 4, 36
- flagTemporalInconsistency, 4, 38
- fusFs, 4, 39
- ISNVFilter, 4, 6, 7, 33, 34, 40, 41–43, 52, 65
- ISNVFilter-class, 41
- lm, 32
- multitestCorrection, 43
- neutralityFromSFS, 4, 23, 44, 67
- p.adjust, 43
- passedISNV, 4, 34, 66
- passedISNV (qcLog), 51

- passedISNV, WithinHostExperiment-method
(qcLog), [51](#)
- permutationTest, [45](#)
- piISNV, [4](#), [46](#)
- plotFrequencySpectrum, [4](#), [47](#)
- plotFrequencyTrajectory, [4](#), [48](#)
- plotPairScatter, [4](#), [49](#)
- plotQCDashboard, [4](#), [50](#)
- plotSFS, [4](#), [50](#)

- qcFilters (qcLog), [51](#)
- qcFilters, WithinHostExperiment-method
(qcLog), [51](#)
- qcFilters<- (qcLog), [51](#)
- qcFilters<- , WithinHostExperiment-method
(qcLog), [51](#)
- qcLog, [51](#)
- qcLog, WithinHostExperiment-method
(qcLog), [51](#)
- qcLog<- (qcLog), [51](#)
- qcLog<- , WithinHostExperiment-method
(qcLog), [51](#)
- qcSummary, [4](#), [34](#), [52](#)
- quickBottleneck, [4](#), [53](#)

- readWithinHost, [4](#), [9](#), [31](#), [54](#), [66](#)
- readWithinHostTable, [4](#), [55](#), [55](#), [66](#)

- sciStrandBias, [4](#), [37](#), [56](#)
- shannonISNV, [4](#), [57](#)
- show, ISNVFilter-method
(ISNVFilter-class), [41](#)
- show, WithinHostExperiment-method
(WithinHostExperiment-class),
[64](#)
- show, WithinHostSFS-method
(WithinHostSFS-class), [66](#)
- simpsonISNV, [4](#), [58](#)

- tajimaD, [4](#), [15](#), [40](#), [44](#), [46](#), [58](#), [62](#)
- trackFrequency, [4](#), [60](#)
- transmissionPairs, [4](#), [66](#)
- transmissionPairs (qcLog), [51](#)
- transmissionPairs, WithinHostExperiment-method
(qcLog), [51](#)

- validateDiversity, [4](#), [61](#)
- VRanges, [19](#)

- wattersonISNV, [4](#), [62](#)
- WHE-combine, [63](#)
- WithinHostExperiment, [7–17](#), [19](#), [21](#), [24](#), [25](#),
[27](#), [30–39](#), [47–55](#), [60](#), [63](#), [64](#), [64](#), [65](#)
- WithinHostExperiment-class, [64](#)
- WithinHostExperiment-package, [3](#)
- WithinHostSFS, [12](#), [13](#), [20](#), [22](#), [23](#), [44](#), [50](#)
- WithinHostSFS (WithinHostSFS-class), [66](#)
- WithinHostSFS-class, [66](#)