Package 'panpathAudit'

November 20, 2026

Type Package

Title Audit Sequence Preservation in GFA Traversals

Version 0.99.3

Description Audits source FASTA sequences against named paths and coordinate-aware walks in GFA files. It reports exact recovery, divergence coordinates, digests, and optional base-level statistics. Graph topology is not validated.

License MIT + file LICENSE

Encoding UTF-8

Language en-US

Depends R (>= 4.6.0)

Imports methods, Rcpp, S4Vectors

LinkingTo Rcpp

Suggests BiocCheck, Biostrings, BiocStyle, knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

SystemRequirements C++20, zlib

biocViews Alignment, DataImport, GenomeAssembly, Genetics, Sequencing

URL https://github.com/gkanogiannis/panpathAudit

BugReports https://github.com/gkanogiannis/panpathAudit/issues

NeedsCompilation yes

RoxygenNote 8.1.0

Config/testthat/edition 3

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

git_branch devel

git_last_commit fa93189

git_last_commit_date 2026-11-17

Repository Bioconductor 3.24

Date/Publication 2026-11-19

Author Anestis Gkanogiannis [aut, cre] (ORCID:
<https://orcid.org/0000-0002-6441-0688>),
Jack O'Connor [cph] (Author of the bundled BLAKE3 C implementation in src/vendor/blake3 (CC0 license)),
Samuel Neves [cph] (Author of the bundled BLAKE3 C implementation in src/vendor/blake3 (CC0 license)),
Alain Mosnier [cph] (Author of the bundled SHA-256 implementation in src/vendor/sha256 (0BSD license))

Maintainer Anestis Gkanogiannis <anestis@gkanogiannis.com>

Contents

panpathAudit-package 2
auditGFA 3
PanPathAuditResult-class 5

Index **7**

`panpathAudit-package` *Audit sequence preservation in GFA traversals*

Description

Audits FASTA sequences against named GFA paths and walks without claiming to validate graph topology.

For each traversal, **panpathAudit** reports whether the sequence spelled by a GFA `P` path or coordinate-aware `W` walk is identical to its source FASTA record. Every traversal gets SHA-256 and BLAKE3 digests of the source and graph sequences, and divergent traversals also get the coordinates of the first difference. Optionally, the package also computes base-level comparison statistics. Links, overlaps beyond blunt joins, and connectivity are not checked.

Details

The main entry point is `auditGFA()`, which takes one or more FASTA files (plain or gzip-compressed), one GFA file, and an optional mapping between FASTA files and path or walk names. It returns a `PanPathAuditResult` object, whose contents are read with these accessors:

- `auditOutcomes()`: one row per traversal, as a `DataFrame`.
- `auditSummary()`: counts and percentages per outcome status.
- `auditStatistics()`: base-level statistics when `statistics = "comprehensive"`.
- `auditProvenance()`: inputs, settings, and versions used.
- `auditTelemetry()`: memory limit, peak tracked memory, and thread count.

The native engine is a C++ port of the Rust command-line tool `panpath-audit` (https://github.com/gkanogiannis/panpath-audit). See `system.file("PROVENANCE.md", package = "panpathAudit")` for the port, bundled third-party code, and example data.

The vignette gives a worked introduction: `vignette("panpathAudit", package = "panpathAudit")`.

Author(s)

Anestis Gkanogiannis <anestis@gkanogiannis.com>

Maintainer: Anestis Gkanogiannis <anestis@gkanogiannis.com>

References

Gkanogiannis, A. (2026). Panpath-Audit: Sequence-Level Verification of Embedded Pangenome Paths Against Source Assemblies. *Preprints*. doi:10.20944/preprints202608.1872.v1

See Also

`auditGFA` to run an audit; `PanPathAuditResult` for the result class and its accessors.

Package website and issue tracker: https://github.com/gkanogiannis/panpathAudit

Examples

```
fasta <- system.file("extdata", "example.fa", package = "panpathAudit")
gfa <- system.file("extdata", "example.gfa", package = "panpathAudit")
result <- auditGFA(fasta, gfa)
result
summary(result)
```

`auditGFA` *Audit FASTA sequences against GFA traversals*

Description

Compares source sequences with named GFA `P` paths and coordinate-aware `W` walks. Graph topology is not validated.

Usage

```
auditGFA(
  fasta,
  gfa,
  mapping = NULL,
  statistics = c("basic", "comprehensive"),
  threads = 1L,
  memoryMiB = 1024L,
  alignmentMaxCells = 10000000
)
```

Arguments

`fasta` The source sequences, as one of:

- a `character()` vector of paths to one or more FASTA files, plain or gzip-compressed, with no `NA` or empty strings;
- a `Biostrings::DNAStringSet` of sequences already in memory, treated as one source; every sequence must be named;
- a `list` (or `Biostrings::DNAStringSetList`) in which each element is a single path or a `DNAStringSet`, one element per source.

Records are matched to GFA traversals by name, after applying `mapping`. As in a FASTA header, a record's name is the first word of its name, so a sequence named `"chr1 assembled"` matches `chr1`. Each `DNAStringSet` is written to a temporary FASTA file in `tempdir()` for the duration of the call, so it needs that much free disk space; it is shown as `<DNAStringSet name>` (or its position) in provenance and diagnostics. **Biostrings** is only needed for `DNAStringSet` input.

`gfa` `character(1)`. Path to one GFA file, plain or gzip-compressed.

`mapping` `NULL` (default), or a `data.frame` or `DataFrame` with one row per source in `fasta` (one per path, `DNAStringSet`, or list element), in the same order. It says how each FASTA file's record names map to GFA path or walk names. Columns:

- `mode` Required. One of `"exact"` (use the record name as is), `"pansn"` (`sample#haplotype#name`), or `"prefix"` (`prefix` followed by the record name).
- `sample` Sample name. Required for `"pansn"` rows.
- `haplotype` Haplotype number, given as digits (e.g. `"1"` or `1`). Required for `"pansn"` rows.
- `prefix` Non-empty prefix string. Required for `"prefix"` rows.

Columns that a row's mode does not use may be omitted or left as `""` or `NA`. With `NULL`, every file uses `"exact"`.

`statistics` `character(1)`. `"basic"` (default) reports outcomes and digests only. `"comprehensive"` also aligns every divergent pair and reports base-level statistics; see `auditStatistics`.

`threads` `numeric(1)`. Number of worker threads; a positive whole number. Default `1L`. Results do not depend on the thread count.

`memoryMiB` `numeric(1)`. Limit, in MiB, on the memory used by sequences held at the same time; a positive whole number. Default `1024L`. A single sequence larger than the limit is still audited, on its own, and is counted in `auditTelemetry(x)$oversized_contigs`.

`alignmentMaxCells` `numeric(1)`. Maximum number of alignment matrix cells per divergent pair when `statistics = "comprehensive"`; a positive whole number. Default `1e7`. If the limit is reached, that row gets `alignment_status = "limit_exceeded"` and the audit continues. Ignored when `statistics = "basic"`.

Details

Gzip input is detected from file content. GFA `P` records must use blunt overlaps (`*` or `0M`). `W` identifiers are formed as `sample#haplotype#sequence`; their ranges are zero-based and half-open.

Use `mapping` when FASTA record names differ from the GFA path or walk names, for example when walk names follow the PanSN convention (`sample#haplotype#name`). See the `mapping` argument for the supported modes and columns.

A running audit can be interrupted with Ctrl-C (or Esc in the R GUI). Worker threads are stopped, and an R `interrupt` condition is signalled; no partial result is returned.

Argument, input, and operational failures raise conditions with classes `panpathAudit_argument_error`, `panpathAudit_input_error`, and `panpathAudit_operational_error`. Each inherits from `panpathAudit_error` and carries a `diagnostics` `DataFrame`. Diagnostic positions are one-based.

Value

A PanPathAuditResult.

Examples

```
fasta <- system.file("extdata", "example.fa", package = "panpathAudit")
gfa <- system.file("extdata", "example.gfa", package = "panpathAudit")
result <- auditGFA(fasta, gfa)
auditOutcomes(result)
```

PanPathAuditResult-class

Audit result class

Description

Stores traversal outcomes, summary counts, optional comprehensive statistics, provenance, and memory telemetry.

Usage

```
auditOutcomes(x)
auditSummary(x)
auditStatistics(x)
auditProvenance(x)
auditTelemetry(x)
```

Arguments

x	A PanPathAuditResult.

Value

auditOutcomes() returns a DataFrame. Other accessors return SimpleList objects. as.data.frame() returns the outcomes as a base data frame.

Outcome columns

identifier Mapped path or walk identifier.

source_index One-based FASTA index.

status IDENTICAL, DIVERGENT, MISSING_PATH, or MISSING_SOURCE.

record_type P or W.

source_start_0based, source_end_0based_exclusive Walk source range.

source_length, graph_length, addressed_source_bases Sequence and addressed-range sizes.

divergence_kind BASE, END, or RANGE_LENGTH.

source_position_1based, graph_position_1based First divergent base position.

segment, orientation Segment and traversal strand at the divergence.

traversal_position_1based, segment_position_1based Divergence coordinates within the traversal and segment.

range_length, walk_length Expected and reconstructed sizes for a walk range mismatch.

source_sha256, addressed_source_sha256, graph_sha256 SHA-256 digests.

source_blake3, addressed_source_blake3, graph_blake3 BLAKE3 digests.

`source_ambiguous_bases`, `graph_ambiguous_bases`, `ambiguous_columns` IUPAC ambiguity counts.

`length_delta` Graph length minus addressed source length.

`matched`, `substituted`, `source_only`, `graph_only` Aligned base counts.

`not_embedded`, `missing_path_bases`, `missing_source_bases` Unrepresented base counts.

`unaligned_divergent_source`, `unaligned_divergent_graph` Divergent bases not aligned because of a resource limit.

`edit_distance` Edit distance when alignment completed.

`alignment_status` `not_applicable`, `not_required`, `complete`, `limit_exceeded`, or `memory_limit_exceeded`.

Columns that do not apply contain `NA`.

Accessor structures

`auditSummary()` contains `counts`, `percentages`, and `total`. `auditStatistics()` is empty in basic mode; otherwise it contains `global`, `by_source`, and `by_record_type` aggregates. Each aggregate reports sequence and base totals, ambiguity counts, edit and alignment totals, base-category counts, and status counts.

`auditProvenance()` records package and reference versions, normalized input paths, mapping and resource options, compression, GFA version and record counts, overlap interpretation, and the false `topology_validated` flag.

`auditTelemetry()` contains `limit_bytes`, `peak_tracked_bytes`, `oversized_contigs`, and `threads`.

Examples

```
fasta <- system.file("extdata", "example.fa", package = "panpathAudit")
gfa <- system.file("extdata", "example.gfa", package = "panpathAudit")
result <- auditGFA(fasta, gfa)
summary(result)
```

Index

∗ **package**
panpathAudit-package, 2

as.data.frame,PanPathAuditResult-method
*(*PanPathAuditResult-class*)*, 5
auditGFA, *2*, *3*, 3
auditOutcomes, *2*
auditOutcomes
*(*PanPathAuditResult-class*)*, 5
auditProvenance, *2*
auditProvenance
*(*PanPathAuditResult-class*)*, 5
auditStatistics, *2*, *4*
auditStatistics
*(*PanPathAuditResult-class*)*, 5
auditSummary, *2*
auditSummary
*(*PanPathAuditResult-class*)*, 5
auditTelemetry, *2*
auditTelemetry
*(*PanPathAuditResult-class*)*, 5

panpathAudit *(*panpathAudit-package*)*, 2
panpathAudit-package, 2
PanPathAuditResult, *2–4*
PanPathAuditResult-class, 5

show,PanPathAuditResult-method
*(*PanPathAuditResult-class*)*, 5
summary,PanPathAuditResult-method
*(*PanPathAuditResult-class*)*, 5