Package: molpathR (via r-universe)

November 3, 2026

Title Molecular Pathology Data Platform for Clinical NGS Integration

Description A unified platform for ingesting, linking, querying, and visualizing molecular pathology data from heterogeneous sources including NGS files (FASTQ, BAM, VCF), XML variant interpretation reports, structured pathology PDF reports, pathology and clinical information systems, and survival outcome data. Provides a searchable in-memory database and an interactive Shiny application for clinical exploration and cohort analysis. Raw sequence and alignment readers do not perform variant calling or clinical interpretation.

Version 0.1.0.9001

License MIT + file LICENSE

URL https://github.com/cttir/molpathR, https://cttir.github.io/molpathR/

BugReports https://github.com/cttir/molpathR/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

LazyData true

Depends R (>= 4.1.0)

Imports tibble, dplyr, tidyr, readr, readxl, xml2, jsonlite, pdftools, stringr, rlang, cli, ggplot2, survival, DT, shiny (>= 1.6.0), bslib (>= 0.5.0), plotly, shinyWidgets, stats, utils, tools, methods

Suggests Rsamtools, VariantAnnotation, GenomicRanges, ShortRead, Biostrings, BiocGenerics, IRanges, SummarizedExperiment, testthat (>= 3.0.0), knitr, rmarkdown, pkgdown, withr, shinytest2, writexl

Config/testthat/edition 3

VignetteBuilder knitr

Config/Needs/website CTTIR/themakR

Config/roxygen2/version 8.0.0

Config/pak/sysreqs cmake make libicu-dev libjpeg-dev libuv1-dev libxml2-dev libssl-dev libpoppler-cpp-dev poppler-data libx11-dev zlib1g-dev

Repository https://cttir.r-universe.dev

Date/Publication 2026-11-03 11:54:04 UTC

RemoteUrl https://github.com/CTTIR/molpathR

RemoteRef HEAD

RemoteSha 133e5ecc539f3ca84e84274bbed615228c6f9d6f

Contents

mp_add_data . . . 3
mp_build_db . . . 3
mp_example_db . . . 5
mp_example_files . . . 6
mp_get_patient . . . 7
mp_load_db . . . 7
mp_merge_db . . . 8
mp_plot_cohort_overview . . . 9
mp_plot_mutation_spectrum . . . 9
mp_plot_survival . . . 10
mp_plot_timeline . . . 11
mp_plot_vaf_distribution . . . 11
mp_plot_variant_landscape . . . 12
mp_query_patients . . . 13
mp_query_samples . . . 13
mp_query_variants . . . 14
mp_read_auto . . . 15
mp_read_bam . . . 16
mp_read_fastq . . . 17
mp_read_nexus_clinical . . . 17
mp_read_nexus_pathology . . . 18
mp_read_pdf_report . . . 19
mp_read_survival . . . 20
mp_read_vcf . . . 20
mp_read_xml_report . . . 21
mp_run_app . . . 22
mp_save_db . . . 23
mp_search . . . 23
mp_summary . . . 24
mp_validate_db . . . 25
new_molpath_db . . . 26
new_molpath_parsed . . . 27
print.molpath_db . . . 28
print.molpath_parsed . . . 28
summary.molpath_db . . . 29

summary.molpath_parsed 29
validate_molpath_db 30

Index **31**

`mp_add_data` *Add parsed data to an existing molpath_db*

Description

Appends new data from one or more `molpath_parsed` objects to an existing `molpath_db` database. Records are deduplicated after appending.

Usage

```
mp_add_data(db, parsed)
```

Arguments

`db`	A `molpath_db` object to update.
`parsed`	A single `molpath_parsed` object or a list of `molpath_parsed` objects to add.

Value

An updated `molpath_db` object with the new data merged in.

Examples

```
db <- mp_example_db(n_patients = 5, seed = 1)
new_data <- new_molpath_parsed(
  data = tibble::tibble(patient_id = "P99", os_months = 12,
    os_status = 1L, pfs_months = 6, pfs_status = 1L),
  source_type = "survival", source_file = "new.csv"
)
db <- mp_add_data(db, new_data)
```

`mp_build_db` *Build a molpath_db from parsed data objects*

Description

Constructs a unified `molpath_db` database object from one or more `molpath_parsed` objects. Each parsed object is categorized by its `source_type` and merged into the appropriate database table. Raw FASTQ/BAM reads remain available through their readers but cannot be interpreted as variant calls by this function. Unknown gene annotations, pathogenicity and patient links remain missing. Use `mp_validate_db()` to assess incomplete links.

Usage

```
mp_build_db(...)
```

Arguments

`...`	One or more `molpath_parsed` objects, or lists of `molpath_parsed` objects. All objects are flattened and processed.

Details

Source types map to database tables as follows:

- `"vcf"` – non-reference GT calls or explicitly normalized variants
- `"xml_report"` – variants and explicitly identified patients/samples
- `"pdf_report"` – reports
- `"nexus_pathology"` – patients and samples
- `"nexus_clinical"` – clinical
- `"survival"` – survival

Value

A `molpath_db` S3 object containing linked patients, samples, variants, reports, clinical, survival, and metadata tables.

Examples

```
# Build from a molpath_parsed object
surv_parsed <- new_molpath_parsed(
  data = tibble::tibble(
    patient_id = c("P1", "P2"),
    os_months = c(12, 24),
    os_status = c(1L, 0L),
    pfs_months = c(6, 18),
    pfs_status = c(1L, 0L)
  ),
  source_type = "survival",
  source_file = "test.csv"
)
db <- mp_build_db(surv_parsed)
```

`mp_example_db` *Create a synthetic molecular pathology database*

Description

Generates a realistic but entirely synthetic `molpath_db` object containing patients, samples, variants, reports, clinical data, and survival data. The data set is designed to mirror the structure of a real-world molecular pathology database with plausible clinical correlations, including diagnosis-specific mutation profiles, TNM staging, and survival outcomes.

Usage

```
mp_example_db(n_patients = 150, seed = 42)
```

Arguments

`n_patients`	Integer. Number of patients to generate. Default `150`.
`seed`	Integer. Random seed for reproducibility. Default `42`. The caller's random-number state is restored, including on error.

Value

A `molpath_db` object (S3 list) containing:

patients A `tibble` with columns patient_id, age, sex, diagnosis, diagnosis_date.

samples A `tibble` with columns sample_id, patient_id, sample_type, date, source_file.

variants A `tibble` with columns sample_id, gene, variant, variant_type, classification, vaf, chromosome, position, ref_allele, alt_allele.

reports A `tibble` with columns sample_id, report_type, report_date, summary_text, source_file.

clinical A `tibble` with columns patient_id, parameter, value, date, source.

survival A `tibble` with columns patient_id, os_months, os_status, pfs_months, pfs_status.

Examples

```
db <- mp_example_db()
db$patients
nrow(db$variants)

# Smaller data set for quick tests
db_small <- mp_example_db(n_patients = 20, seed = 123)
```

`mp_example_files` *Write synthetic example files to a directory*

Description

Creates a set of realistic mock files that can be used to test the **molpathR** import pipeline. Files include VCF, XML, text-based PDF-like reports, a survival CSV, and a tiny FASTQ file.

Usage

```
mp_example_files(
  dir = tempdir(),
  seed = 42,
  reference_date = as.Date("2025-01-01")
)
```

Arguments

`dir`	Character. Directory where files are written. Default `tempdir()`.
`seed`	Integer. Random seed for reproducibility. Default `42`. The caller's random-number state is restored, including on error.
`reference_date`	Date. Fixed anchor for synthetic report dates; default 2025-01-01. File contents do not depend on the day they are generated.

Value

A named list of file paths:

vcf Character vector of 5 VCF file paths.

xml Character vector of 5 XML report file paths.

pdf Character vector of 5 text-based mock PDF report paths.

survival Path to the survival CSV file.

fastq Path to the sample FASTQ file.

Examples

```
files <- mp_example_files()
readLines(files$vcf[1], n = 5)
```

mp_get_patient *Get all data for a single patient*

Description

Returns a named list containing all data layers for the given patient, including samples, variants (via sample linkage), reports, clinical data, and survival data.

Usage

```
mp_get_patient(db, patient_id)
```

Arguments

db	A molpath_db object.
patient_id	A single patient ID string.

Value

A named list with elements: patient, samples, variants, reports, clinical, survival.

Examples

```
db <- mp_example_db(n_patients = 20, seed = 1)
pat <- mp_get_patient(db, db$patients$patient_id[1])
pat$samples
```

mp_load_db *Load a molpath_db from disk*

Description

Reads an RDS file, verifies it contains a valid molpath_db object, and prints a summary of the loaded database. Known legacy zero-column empty tables and one-row metadata tibbles are normalized explicitly. Malformed schemas or values are rejected; incomplete foreign-key links remain visible through mp_validate_db() rather than being invented.

Usage

```
mp_load_db(path)
```

Arguments

path	Character string giving the file path to an RDS file containing a molpath_db object.

Value

A molpath_db object.

Examples

```
db <- mp_example_db(n_patients = 5, seed = 1)
tmp <- tempfile(fileext = ".rds")
mp_save_db(db, tmp)
db2 <- mp_load_db(tmp)
unlink(tmp)
```

---

mp_merge_db *Merge two molpath_db objects*

Description

Combines all tables from two molpath_db objects into a single unified database. Records are deduplicated after merging. Metadata source files are combined as a union. Identical primary-key rows are deduplicated and missing attributes may be filled from the other source. Conflicting observed patient, sample or survival attributes require explicit reconciliation and cause an error.

Usage

```
mp_merge_db(db1, db2)
```

Arguments

db1	A molpath_db object.
db2	A molpath_db object.

Value

A new molpath_db object containing merged data from both inputs.

Examples

```
db1 <- mp_example_db(n_patients = 5, seed = 1)
db2 <- db1
db1$survival <- db1$survival[0, ]
db_combined <- mp_merge_db(db1, db2)
```

mp_plot_cohort_overview

Plot cohort overview

Description

Returns a list of four ggplot objects summarising the database: diagnosis distribution, age by sex, sample types, and top mutated genes. Arrange with `patchwork` or `cowplot` as needed.

Usage

```
mp_plot_cohort_overview(db)
```

Arguments

db	A `molpath_db` object.

Value

A named list of four ggplot2::ggplot objects: `diagnosis`, `age`, `sample_types`, `top_genes`.

Examples

```
db <- mp_example_db(n_patients = 20, seed = 1)
plots <- mp_plot_cohort_overview(db)
plots$diagnosis
```

mp_plot_mutation_spectrum

Plot mutation spectrum

Description

Bar chart of SNV substitution types (C>A, C>G, C>T, T>A, T>C, T>G).

Usage

```
mp_plot_mutation_spectrum(db, sample_id = NULL)
```

Arguments

db	A `molpath_db` object.
sample_id	Optional character vector of sample IDs to include. If `NULL`, aggregate across all samples.

Value

A ggplot2::ggplot object.

Examples

```
db <- mp_example_db(n_patients = 20, seed = 1)
mp_plot_mutation_spectrum(db)
```

`mp_plot_survival` *Plot Kaplan-Meier survival curves*

Description

Creates Kaplan-Meier curves from the survival table. Optionally stratifies by a clinical variable or gene mutation status.

Usage

```
mp_plot_survival(db, group_by = NULL, type = c("os", "pfs"))
```

Arguments

`db`	A `molpath_db` object.
`group_by`	Optional character string. If it matches a column in the patients table (e.g., `"diagnosis"`), stratify by that variable. If it matches a gene name present in the variants table, stratify by presence of a recorded variant. Absence of a record does not establish a negative assay or wild-type genotype.
`type`	Character; `"os"` for overall survival or `"pfs"` for progression-free survival.

Value

A ggplot2::ggplot object.

Examples

```
db <- mp_example_db(n_patients = 50, seed = 1)
mp_plot_survival(db, type = "os")
mp_plot_survival(db, group_by = "diagnosis", type = "os")
```

mp_plot_timeline *Plot patient timeline*

Description

Displays a timeline of events for a single patient: sample collections, report dates, and clinical measurements.

Usage

```
mp_plot_timeline(db, patient_id)
```

Arguments

db	A molpath_db object.
patient_id	A single patient ID.

Value

A ggplot2::ggplot object.

Examples

```
db <- mp_example_db(n_patients = 20, seed = 1)
pid <- db$patients$patient_id[1]
mp_plot_timeline(db, pid)
```

mp_plot_vaf_distribution *Plot variant allele frequency distribution*

Description

Histogram or density plot of variant allele frequencies (VAF), optionally filtered by gene.

Usage

```
mp_plot_vaf_distribution(db, gene = NULL)
```

Arguments

db	A molpath_db object.
gene	Optional character vector of gene symbols to include.

Value

A ggplot2::ggplot object.

Examples

```
db <- mp_example_db(n_patients = 20, seed = 1)
mp_plot_vaf_distribution(db)
mp_plot_vaf_distribution(db, gene = "TP53")
```

`mp_plot_variant_landscape`

Plot variant landscape (oncoplot)

Description

Creates a tile plot showing variant classifications across samples and genes.

Usage

```
mp_plot_variant_landscape(db, genes = NULL, top_n = 20)
```

Arguments

db	A molpath_db object.
genes	Character vector of genes to display. If NULL, the top top_n most frequently mutated genes are shown.
top_n	Integer. Number of top genes to show when genes is NULL.

Value

A ggplot2::ggplot object.

Examples

```
db <- mp_example_db(n_patients = 20, seed = 1)
mp_plot_variant_landscape(db, top_n = 10)
```

mp_query_patients *Filter patients in a molpath_db*

Description

Uses tidy evaluation to filter the patients table.

Usage

```
mp_query_patients(db, ...)
```

Arguments

db	A molpath_db object.
...	Filter expressions passed to dplyr::filter().

Value

A tibble of matching patients.

Examples

```
db <- mp_example_db(n_patients = 20, seed = 1)
mp_query_patients(db, diagnosis == "Melanoma")
mp_query_patients(db, age > 60, sex == "F")
```

mp_query_samples *Filter samples in a molpath_db*

Description

Uses tidy evaluation to filter the samples table.

Usage

```
mp_query_samples(db, ...)
```

Arguments

db	A molpath_db object.
...	Filter expressions passed to dplyr::filter().

Value

A tibble of matching samples.

Examples

```
db <- mp_example_db(n_patients = 20, seed = 1)
mp_query_samples(db, sample_type == "FFPE")
```

`mp_query_variants` *Filter variants in a molpath_db*

Description

Filter the variants table by gene, classification, VAF range, or variant type.

Usage

```
mp_query_variants(
  db,
  genes = NULL,
  classification = NULL,
  min_vaf = NULL,
  max_vaf = NULL,
  variant_type = NULL
)
```

Arguments

`db`	A `molpath_db` object.
`genes`	Character vector of gene symbols to include. `NULL` means all.
`classification`	Character vector of classifications to include. `NULL` means all.
`min_vaf`	Minimum variant allele frequency (inclusive). `NULL` for no lower bound.
`max_vaf`	Maximum variant allele frequency (inclusive). `NULL` for no upper bound.
`variant_type`	Character vector of variant types to include (e.g., `"SNV"`, `"Indel"`, `"CNV"`, `"Fusion"`). `NULL` means all.

Value

A tibble of matching variants.

Examples

```
db <- mp_example_db(n_patients = 20, seed = 1)
mp_query_variants(db, genes = c("TP53", "KRAS"))
mp_query_variants(db, classification = "Pathogenic", min_vaf = 0.1)
```

`mp_read_auto` *Automatically read a molecular pathology file*

Description

Detect the file type from its extension and/or content, then dispatch to the appropriate parser.

Usage

```
mp_read_auto(path, source_name = NULL)
```

Arguments

`path`	Character scalar. Path to the file.
`source_name`	Optional original filename for an extensionless upload. Dispatch uses its basename, and the result retains that basename as source identity. A disposable copy is removed after reading.

Details

Extension mapping:

- `.vcf`, `.vcf.gz`: `mp_read_vcf`
- `.fastq`, `.fq`, `.fastq.gz`, `.fq.gz`: `mp_read_fastq`
- `.bam`: `mp_read_bam`
- `.xml`: `mp_read_xml_report`
- `.pdf`: `mp_read_pdf_report`
- `.xlsx`, `.xls`, `.csv`: tries survival/clinical format detection

Value

A `molpath_parsed` object produced by the appropriate parser.

Examples

```
f <- system.file("extdata", "example.vcf", package = "molpathR")
if (nzchar(f)) {
  result <- mp_read_auto(f)
  print(result)
}
```

`mp_read_bam` *Read a BAM file*

Description

Parse a BAM (Binary Alignment Map) file into a tidy tibble.

Usage

```
mp_read_bam(path, regions = NULL)
```

Arguments

`path`	Character scalar. Path to a `.bam` file.
`regions`	Character scalar or `NULL`. An optional genomic region string such as `"chr1:1000-2000"` to restrict the query.

Details

Uses `Rsamtools::scanBam()` when installed. Otherwise it calls `samtools view`; if neither is available an informative error is raised. Backend decoding errors are reported rather than triggering another reader.

Value

A `molpath_parsed` object whose `data` slot is a tibble with columns `qname`, `flag`, `rname`, `pos`, `mapq`, `cigar`, `seq`, and `qual`.

Examples

```
bam_file <- system.file("extdata", "example.bam", package = "molpathR")
if (nzchar(bam_file)) {
  result <- mp_read_bam(bam_file)
  print(result)
}
```

`mp_read_fastq` *Read a FASTQ file*

Description

Parse a FASTQ file into a tidy tibble of sequencing reads.

Usage

```
mp_read_fastq(path, n = NULL)
```

Arguments

`path`	Character scalar. Path to a `.fastq`, `.fq`, `.fastq.gz`, or `.fq.gz` file.
`n`	Integer or `NULL`. If specified, only the first `n` records are returned.

Details

Uses ShortRead when installed, streaming the first requested records rather than sampling. Otherwise a strict 4-line text parser is used. Invalid or incomplete records raise an error; backend parse errors are not hidden.

Value

A `molpath_parsed` object whose `data` slot is a tibble with columns `read_id`, `sequence`, `quality`, and `seq_length`.

Examples

```
fq_file <- system.file("extdata", "example.fastq", package = "molpathR")
if (nzchar(fq_file)) {
  result <- mp_read_fastq(fq_file, n = 100)
  print(result)
}
```

`mp_read_nexus_clinical`
Read Nexus clinical data

Description

Parse a clinical data export (CSV format) from a clinical information system such as Nexus.

Usage

```
mp_read_nexus_clinical(path_or_connection)
```

Arguments

`path_or_connection`
: Character scalar. Path to the CSV file, or an open connection.

Value

A `molpath_parsed` object whose `data` slot is a tibble with columns `patient_id`, `parameter`, `value`, `unit`, `date`, and `source`.

Examples

```
clin_file <- system.file("extdata", "nexus_clinical.csv",
                         package = "molpathR")
if (nzchar(clin_file)) {
  result <- mp_read_nexus_clinical(clin_file)
  print(result)
}
```

`mp_read_nexus_pathology` *Read Nexus pathology data*

Description

Parse an export from a pathology information system (e.g. Nexus) in either CSV or XML format.

Usage

```
mp_read_nexus_pathology(path_or_connection)
```

Arguments

`path_or_connection`
: Character scalar. Path to the CSV or XML export file, or an open connection.

Value

A `molpath_parsed` object whose `data` slot is a tibble with columns `patient_id`, `name`, `dob`, `sex`, `sample_id`, `sample_type`, `diagnosis`, and `report_date`.

Examples

```
nexus_file <- system.file("extdata", "nexus_pathology.csv",
                          package = "molpathR")
if (nzchar(nexus_file)) {
  result <- mp_read_nexus_pathology(nexus_file)
  print(result)
}
```

`mp_read_pdf_report` *Read a PDF pathology report*

Description

Extract structured information from a pathology PDF report using `pdftools::pdf_text()` and regex-based section parsing. Complete extracted pages and text are retained in metadata even when section heuristics match only part of the report. Database report summaries use that complete text. This does not perform OCR or validate extraction accuracy against the PDF.

Usage

```
mp_read_pdf_report(path, template = "generic")
```

Arguments

`path`	Character scalar. Path to the `.pdf` file.
`template`	Character scalar. Template name controlling which regex patterns are applied. Currently `"generic"` (default) is supported.

Value

A `molpath_parsed` object whose `data` slot is a tibble with columns `section` and `content` extracted from the report (Patient, Sample, Findings, Interpretation, Recommendations).

Examples

```
pdf_file <- system.file("extdata", "report.pdf", package = "molpathR")
if (nzchar(pdf_file)) {
  result <- mp_read_pdf_report(pdf_file)
  print(result)
}
```

mp_read_survival *Read survival / outcome data*

Description

Read a survival data file in Excel or CSV format and standardise the column names to a canonical schema.

Usage

```
mp_read_survival(path)
```

Arguments

path	Character scalar. Path to a .xlsx, .xls, or .csv file.

Value

A molpath_parsed object whose data slot is a tibble with columns patient_id, os_months, os_status, pfs_months, and pfs_status.

Examples

```
surv_file <- system.file("extdata", "survival.csv", package = "molpathR")
if (nzchar(surv_file)) {
  result <- mp_read_survival(surv_file)
  print(result)
}
```

mp_read_vcf *Read a VCF file*

Description

Parse a Variant Call Format (VCF 4.x) file and return a tidy tibble of variant calls.

Usage

```
mp_read_vcf(path)
```

Arguments

path	Character scalar. Path to a .vcf or .vcf.gz file.

Details

A validated text parser preserves original ID, INFO, FORMAT and sample fields for single- and multi-sample VCFs, including gzip input. Installing VariantAnnotation does not change the returned representation. Header lines and declared sample names are retained in the result metadata.

Value

A `molpath_parsed` object whose `data` slot is a tibble with columns `chrom`, `pos`, `id`, `ref`, `alt`, `qual`, `filter`, `info`, and any sample-level columns present in the file, plus `format` when declared. Sample names conflicting with these fixed output names are rejected.

Examples

```
vcf_file <- system.file("extdata", "example.vcf", package = "molpathR")
if (nzchar(vcf_file)) {
  result <- mp_read_vcf(vcf_file)
  print(result)
}
```

`mp_read_xml_report` *Read an XML variant report*

Description

Parse a structured XML variant interpretation report (e.g. from Molecular Health or similar providers) into a tidy tibble.

Usage

```
mp_read_xml_report(path, provider = "molecular_health")
```

Arguments

`path`	Character scalar. Path to the `.xml` file.
`provider`	Character scalar. Report provider template to apply. Currently supported: `"molecular_health"` (default).

Value

A `molpath_parsed` object whose `data` slot is a tibble with one row per variant and columns for patient info, sample info, gene, variant description, classification, evidence level, therapeutic implications and an explicit VAF proportion when supplied. VAF must be between 0 and 1, inclusive; percentages or other units are not inferred.

Examples

```
xml_file <- system.file("extdata", "report.xml", package = "molpathR")
if (nzchar(xml_file)) {
  result <- mp_read_xml_report(xml_file, provider = "molecular_health")
  print(result)
}
```

`mp_run_app` *Launch the molpathR Shiny application*

Description

Starts an interactive Shiny application for exploring molecular pathology data. If a `molpath_db` object is provided, the app launches pre-loaded with that database. Otherwise, the user can upload data through the app interface.

Usage

```
mp_run_app(db = NULL, ...)
```

Arguments

db	Optional `molpath_db` object to pre-load into the application.
...	Additional arguments passed to `shiny::runApp()`.

Details

CSV/TSV downloads require a working `zip` executable (or `R_ZIPCMD`). Excel downloads additionally require the optional `writexl` package. Uploaded files are staged and validated as one batch before changing session data.

Value

This function does not return a value; it launches a Shiny app.

Examples

```
## Not run:
# Launch with example data
db <- mp_example_db()
mp_run_app(db)

# Launch empty — upload data in browser
mp_run_app()

## End(Not run)
```

mp_save_db *Save a molpath_db to disk*

Description

Validates the database and saves it as a compressed RDS file. Serialization and read-back verification occur in a sibling temporary file before replacement. If replacement fails, the previous destination is restored or an explicit recovery path is reported. Use one writer per destination; this is not a multi-process database transaction. The file path is returned invisibly for use in pipelines.

Usage

```
mp_save_db(db, path)
```

Arguments

db	A molpath_db object to save.
path	Character string giving the file path to save to. Should typically end in .rds.

Value

The file path, returned invisibly.

Examples

```
db <- mp_example_db(n_patients = 5, seed = 1)
tmp <- tempfile(fileext = ".rds")
mp_save_db(db, tmp)
unlink(tmp)
```

mp_search *Free-text search across a molpath_db*

Description

Searches all text fields across all tables for the given term (case-insensitive).

Usage

```
mp_search(db, term)
```

Arguments

db	A molpath_db object.
term	A single search string.

Value

A named list of tibbles, one per table, containing rows that match the search term. Empty tibbles are omitted.

Examples

```
db <- mp_example_db(n_patients = 20, seed = 1)
mp_search(db, "TP53")
```

`mp_summary` *Cohort-level summary of a molpath_db*

Description

Computes summary statistics across all layers of the database.

Usage

```
mp_summary(db)
```

Arguments

db	A `molpath_db` object.

Value

An S3 object of class `molpath_summary` (a named list) containing `n_patients`, `n_samples`, `n_variants`, `n_reports`, `n_clinical`, `diagnosis_distribution`, `variant_gene_counts`, `classification_distribution`, `sample_type_distribution`, `date_range`, and `completeness`.

Examples

```
db <- mp_example_db(n_patients = 20, seed = 1)
mp_summary(db)
```

`mp_validate_db` *Validate a molpath_db for referential integrity and completeness*

Description

Checks that all foreign key relationships in the database are satisfied (e.g., every sample references an existing patient) and reports on data completeness across tables.

Usage

```
mp_validate_db(db)
```

Arguments

`db`	A `molpath_db` object to validate.

Details

The following referential integrity checks are performed:

- All `patient_id` values in the samples table exist in patients.
- All `sample_id` values in the variants table exist in samples.
- All `sample_id` values in the reports table exist in samples.
- All `patient_id` values in the clinical table exist in patients.
- All `patient_id` values in the survival table exist in patients.

When called interactively, results are printed using `cli` formatting.

Value

Invisibly, a list with three elements:

valid Logical. `TRUE` if no referential integrity issues were found.

issues Character vector of issue descriptions (empty if valid).

completeness Named numeric vector with percentages of patients having data in each table (e.g., `pct_with_variants`, `pct_with_survival`).

Examples

```
db <- mp_example_db(n_patients = 10, seed = 1)
validation <- mp_validate_db(db)
validation$valid
```

new_molpath_db *Create a new molpath_db object*

Description

Constructs a molpath_db object, which is a list of linked tibbles representing the core data model for molecular pathology workflows.

Usage

```
new_molpath_db(
 patients = tibble::tibble(patient_id = character(), age = integer(), sex = character(),
    diagnosis = character()),
 samples = tibble::tibble(sample_id = character(), patient_id = character(), sample_type
    = character(), date = as.Date(character()), source_file = character()),
 variants = tibble::tibble(sample_id = character(), gene = character(), variant =
    character(), classification = character(), vaf = double()),
  reports = tibble::tibble(sample_id = character(), report_type = character(),
   report_date = as.Date(character()), summary_text = character(), source_file =
    character()),
 clinical = tibble::tibble(patient_id = character(), parameter = character(), value =
    character(), date = as.Date(character()), source = character()),
 survival = tibble::tibble(patient_id = character(), os_months = double(), os_status =
    integer(), pfs_months = double(), pfs_status = integer()),
 metadata = list(creation_date = Sys.time(), source_files = character(), parser_versions
    = list())
)
```

Arguments

patients	A tibble with columns patient_id, age, sex, diagnosis, and optionally others.
samples	A tibble with columns sample_id, patient_id, sample_type, date, source_file, and optionally others.
variants	A tibble with columns sample_id, gene, variant, classification, vaf, and optionally others.
reports	A tibble with columns sample_id, report_type, report_date, summary_text, source_file, and optionally others.
clinical	A tibble with columns patient_id, parameter, value, date, source, and optionally others.
survival	A tibble with columns patient_id, os_months, os_status, pfs_months, pfs_status, and optionally others.
metadata	A list containing creation_date (POSIXct), source_files (character vector), and parser_versions (named list).

Value

An S3 object of class `molpath_db`.

Examples

```
db <- new_molpath_db()
db
```

`new_molpath_parsed` *Create a new molpath_parsed object*

Description

Constructs a `molpath_parsed` object representing the result of parsing a single data source. This serves as an intermediate representation before data is integrated into a `molpath_db`.

Usage

```
new_molpath_parsed(
  data = tibble::tibble(),
  source_type = character(1L),
  source_file = character(1L),
  parse_date = Sys.time(),
  metadata = list()
)
```

Arguments

`data`	A tibble (or data frame) containing the parsed data.
`source_type`	A character string identifying the source type (e.g., `"vcf"`, `"xml_report"`, `"pdf_report"`, `"excel"`, `"csv"`).
`source_file`	A character string with the path to the source file.
`parse_date`	A `POSIXct` timestamp of when the data was parsed. Defaults to the current time.
`metadata`	A named list of parser-specific provenance, such as original VCF headers.

Value

An S3 object of class `molpath_parsed`.

Examples

```
parsed <- new_molpath_parsed(
  data = tibble::tibble(gene = "BRAF", variant = "V600E"),
  source_type = "vcf",
  source_file = "sample1.vcf"
)
parsed
```

print.molpath_db *Print a molpath_db object*

Description

Displays a human-readable summary of a molpath_db, including table dimensions, date ranges, and data completeness.

Usage

```
## S3 method for class 'molpath_db'
print(x, ...)
```

Arguments

x	A molpath_db object.
...	Additional arguments (ignored).

Value

The molpath_db object (invisibly).

print.molpath_parsed *Print a molpath_parsed object*

Description

Displays a compact summary of a parsed data source, including source metadata and a preview of the data.

Usage

```
## S3 method for class 'molpath_parsed'
print(x, ...)
```

Arguments

x	A molpath_parsed object.
...	Additional arguments (ignored).

Value

The molpath_parsed object (invisibly).

`summary.molpath_db` *Summarise a molpath_db object*

Description

Prints a detailed summary of a `molpath_db`, including per-table completeness, unique counts for key columns, and metadata.

Usage

```
## S3 method for class 'molpath_db'
summary(object, ...)
```

Arguments

`object`	A `molpath_db` object.
`...`	Additional arguments (ignored).

Value

The `molpath_db` object (invisibly).

`summary.molpath_parsed` *Summarise a molpath_parsed object*

Description

Prints a detailed summary of a parsed data source, including column types, missing value counts, and a preview of distinct values.

Usage

```
## S3 method for class 'molpath_parsed'
summary(object, ...)
```

Arguments

`object`	A `molpath_parsed` object.
`...`	Additional arguments (ignored).

Value

The `molpath_parsed` object (invisibly).

validate_molpath_db *Validate a molpath_db object*

Description

Checks that all required tables are present, that each is a tibble, and that mandatory columns, types, value ranges and primary identifiers are valid. Incomplete cross-table links are reported as warnings; use mp_validate_db() for a structured link report. Missing parent evidence is not fabricated.

Usage

```
validate_molpath_db(x)
```

Arguments

x	An object of class molpath_db.

Value

The original molpath_db object (invisibly). Structural or value corruption raises an error. Incomplete links warn and remain unchanged.

Examples

```
db <- new_molpath_db()
validate_molpath_db(db)
```

Index

dplyr::filter(), *13*

ggplot2::ggplot, *9–12*

mp_add_data, 3
mp_build_db, 3
mp_example_db, 5
mp_example_files, 6
mp_get_patient, 7
mp_load_db, 7
mp_merge_db, 8
mp_plot_cohort_overview, 9
mp_plot_mutation_spectrum, 9
mp_plot_survival, 10
mp_plot_timeline, 11
mp_plot_vaf_distribution, 11
mp_plot_variant_landscape, 12
mp_query_patients, 13
mp_query_samples, 13
mp_query_variants, 14
mp_read_auto, 15
mp_read_bam, *15*, 16
mp_read_fastq, *15*, 17
mp_read_nexus_clinical, 17
mp_read_nexus_pathology, 18
mp_read_pdf_report, *15*, 19
mp_read_survival, 20
mp_read_vcf, *15*, 20
mp_read_xml_report, *15*, 21
mp_run_app, 22
mp_save_db, 23
mp_search, 23
mp_summary, 24
mp_validate_db, 25
mp_validate_db(), *3*, *7*, *30*

new_molpath_db, 26
new_molpath_parsed, 27

print.molpath_db, 28
print.molpath_parsed, 28

shiny::runApp(), *22*
summary.molpath_db, 29
summary.molpath_parsed, 29

tibble, *16–21*

validate_molpath_db, 30