Package: selexprepR (via r-universe)

July 12, 2026

Title Preprocessing Public HT-SELEX Sequencing Data

Version 0.99.3

Description Provides reproducible preprocessing for high-throughput SELEX sequencing data. Supports primer inference, extraction of random regions, sparse sequence counting across rounds, quality control, and provenance manifests. Public-study discovery and checksum-aware ENA FASTQ download planning are included for reproducible data acquisition.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Depends R (>= 4.5.0)

Imports Biostrings, digest, httr2, jsonlite, Matrix, S4Vectors, SummarizedExperiment, withr

Suggests arrow, BiocStyle, knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

biocViews Software, DataImport, Preprocessing, QualityControl, Sequencing

Config/testthat/edition 3

URL https://github.com/marcorotanegroni/selexprepR

BugReports https://github.com/marcorotanegroni/selexprepR/issues

Config/pak/sysreqs libssl-dev zlib1g-dev

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

Date/Publication 2026-07-12 14:39:54 UTC

RemoteUrl https://github.com/BiocStaging/selexprepR

RemoteRef HEAD

RemoteSha fc89b81f849c07c2db372617ca327964e2f531ba

Contents

`build_selexprep_manifest`

Build an R-native reproducibility manifest

Description

Builds a versioned `selexprep_manifest_v2` object. The reader also accepts the historical Python `selexprep_manifest_v1` schema, so existing analysis records remain inspectable. Hashes are computed for every supplied existing path and are keyed relative to `input_root` or `output_root` when supplied.

Usage

```
build_selexprep_manifest(
  library_report,
  input_paths = character(),
  output_paths = character(),
  accession = NULL,
  bioproject_id = NULL,
```

```
  runs = character(),
  parameters = character(),
  runtime_seconds_per_stage = numeric(),
  flags = character(),
  input_root = NULL,
  output_root = NULL
)
```

Arguments

library_report	A selexprep_library_report.
input_paths	Existing input file paths to hash.
output_paths	Existing output file paths to hash.
accession	Optional public accession.
bioproject_id	Optional BioProject accession.
runs	Run accessions.
parameters	Named character parameters used for the run.
runtime_seconds_per_stage	Named numeric stage durations.
flags	Character QC flag names.
input_root	Optional root used to make input hash keys relative.
output_root	Optional root used to make output hash keys relative.

Value

A validated selexprep_manifest.

Examples

```
p5 <- 'GGTAATACGACTCACTATAGGG'
p3 <- 'CCATGCATGCATGCATGCAT'
report <- selexprep_detect(list(round_00 = rep(paste0(p5, 'ACGTACGTACGTACGT', p3), 500)))
build_selexprep_manifest(report)
```

---

fetch_selexprep_reads *Fetch FASTQ files from an ENA inspection*

Description

Creates a deterministic, checksum-aware download plan from an existing inspection or an accession. By default no bytes are transferred; set dry_run = FALSE to download to outdir. Each completed file is checked against ENA's MD5 value when one is supplied.

Usage

```
fetch_selexprep_reads(
  accession_or_inspection,
  outdir,
  dry_run = TRUE,
  overwrite = FALSE,
  timeout_seconds = 30,
  require_assigned_rounds = TRUE,
  overrides = numeric()
)
```

Arguments

accession_or_inspection	A character accession or a selexprep_inspection returned by inspect_selexprep_accession().
outdir	Destination directory for downloaded FASTQ files.
dry_run	Whether to return the plan without downloading.
overwrite	Whether an existing destination file may be replaced.
timeout_seconds	Positive timeout for metadata and FASTQ requests.
require_assigned_rounds	Whether to refuse ambiguous or missing round assignments before creating a download plan.
overrides	Optional named integer mapping of run accessions to manually curated round numbers.

Value

A selexprep_fetch_result list containing the run-level $plan, $downloaded_files, and $dry_run.

Examples

```
inspection <- inspect_selexprep_accession('PRJDB19098')
fetch_selexprep_reads(inspection, 'raw/PRJDB19098')
```

---

infer_selexprep_rounds

Infer SELEX round assignments from public metadata

Description

Applies a conservative, ordered metadata cascade: structured sample attributes, sample title, library name, experiment title, and design description. A run remains unsafe when no round is found or when one field contains conflicting candidates. The function records evidence rather than using enrichment signals to repair metadata.

Usage

```
infer_selexprep_rounds(runs, overrides = numeric())
```

Arguments

runs	A data frame or S4Vectors::DataFrame containing run_accession and optional metadata columns sample_title, library_name, experiment_title, design_description, and sample_attributes.
overrides	Optional named integer vector mapping run accessions to manually curated round numbers.

Value

A S4Vectors::DataFrame with one conservative round assignment per run. round_candidates is a list-column and preserves ambiguity.

Examples

```
runs <- data.frame(
    run_accession = c('SRR1', 'SRR2'),
    sample_title = c('Thrombin Round 1', 'Thrombin Round 2')
)
infer_selexprep_rounds(runs)
```

---

inspect_selexprep_accession

Inspect public sequencing metadata at ENA

Description

Queries ENA's read-run filereport endpoint without downloading sequence data. The returned selexprep_inspection retains verbatim library strategy/source metadata and one row per run with its FASTQ URLs, sizes, and MD5 checksums.

Usage

```
inspect_selexprep_accession(accession, timeout_seconds = 30)
```

Arguments

accession	Study- or run-level ENA/INSDC accession.
timeout_seconds	Positive network timeout in seconds.

Value

A selexprep_inspection list with a run-level S4Vectors::DataFrame in $runs.

Examples

```
inspection <- inspect_selexprep_accession('PRJDB19098')
inspection$runs
```

`plot.selexprep_qc` *Plot SELEX quality-control diagnostics*

Description

Renders an R-native diagnostic surface corresponding to the four QC artifacts produced by the original Python workflow. Read retention and its primer-match proxy use extraction provenance from `y`. Sequence lengths are shown as read-weighted, per-round fractions so rounds with different depths remain comparable. The diversity view contains unique counts, Shannon entropy, and top-100 coverage.

Usage

```
## S3 method for class 'selexprep_qc'
plot(
  x,
  y = NULL,
  which = c("retention", "primer_match", "length", "diversity"),
  ...
)
```

Arguments

`x`	A `selexprep_qc` object returned by `selexprep_qc()`.
`y`	Optional `SummarizedExperiment` used to compute `x`. It supplies count distributions and extraction provenance.
`which`	One or more diagnostic groups: `'retention'`, `'primer_match'`, `'length'`, or `'diversity'`.
`...`	Named graphical parameters passed to `graphics::par()`.

Details

When `y` is omitted, summaries already retained in `x` are still plotted. Panels that require raw counts or extraction provenance state that the data are unavailable instead of reconstructing them from incomplete summaries.

Value

`x`, invisibly.

Examples

```
p5 <- 'GGTAATACGACTCACTATAGGG'
p3 <- 'CCATGCATGCATGCATGCAT'
pools <- list(round_00 = rep(paste0(p5, 'ACGTACGT', p3), 500))
result <- run_selexprep(pools, low_total_reads = 0)
plot(S4Vectors::metadata(result)$qc, result, which = 'diversity')
```

---

read_library_report *Read a LibraryReport JSON file*

Description

Read a LibraryReport JSON file

Usage

```
read_library_report(path)
```

Arguments

path	Path to a JSON file written by `write_library_report()` or by the compatible Python implementation.

Value

A validated `selexprep_library_report`.

Examples

```
p5 <- 'GGTAATACGACTCACTATAGGG'
p3 <- 'CCATGCATGCATGCATGCAT'
report <- selexprep_detect(list(round_00 = rep(paste0(p5, 'ACGTACGTACGTACGT', p3), 500)))
path <- tempfile(fileext = '.json')
write_library_report(report, path)
read_library_report(path)
```

---

read_selexprep_counts *Read a selexprep count table*

Description

Reads the stable four-column count contract (`sequence`, `reads`, `rank`, `rpm`) from TSV, CSV, RDS, or Parquet. Parquet support is optional and requires `arrow`; TSV is the portable default for R-native workflows.

Usage

```
read_selexprep_counts(path)
```

Arguments

path	Path to a .tsv, .csv, .rds, or .parquet count file.

Value

A validated S4Vectors::DataFrame with a Biostrings::BStringSet sequence column.

Examples

```
path <- tempfile(fileext = '.tsv')
write_selexprep_counts(selexprep_count(c('ACGT', 'ACGT', 'GGGG')), path)
read_selexprep_counts(path)
```

---

read_selexprep_fastq *Read sequences from a FASTQ file*

Description

Reads a plain or gzip-compressed FASTQ file into a Biostrings::BStringSet. BStringSet deliberately preserves both DNA and RNA alphabets, avoiding the loss of uracil that would occur when every input is coerced to DNA.

Usage

```
read_selexprep_fastq(path, max_reads = NULL)
```

Arguments

path	Path to a FASTQ or FASTQ.gz file.
max_reads	Optional positive maximum number of records to retain.

Value

A Biostrings::BStringSet of read sequences.

Examples

```
path <- tempfile(fileext = '.fastq')
writeLines(c('@read_1', 'ACGU', '+', 'IIII'), path)
read_selexprep_fastq(path)
```

`read_selexprep_inputs` *Read FASTQ files into round-aware pipeline inputs*

Description

Converts local FASTQ paths or a completed ENA fetch result into the R-native inputs used by `run_selexprep()`. Files belonging to the same round and mate are concatenated in input-table order. Plain and gzip-compressed FASTQ files are supported through `read_selexprep_fastq()`.

Usage

```
read_selexprep_inputs(x, round = NULL, mate = NULL, max_reads_per_file = NULL)
```

Arguments

`x`	A character vector of FASTQ paths, a data frame-like object with `path` and `round_number` columns, or a non-dry-run `selexprep_fetch_result` returned by `fetch_selexprep_reads()`.
`round`	Optional vector of non-negative round numbers. It is required for character paths unless their vector names end in round numbers.
`mate`	Optional vector containing `R1`/`R2` or `1`/`2`. When omitted, mate labels are inferred from filenames.
`max_reads_per_file`	Optional positive maximum number of records read from each FASTQ file.

Details

Mate labels are inferred conservatively from conventional `_R1`, `_R2`, `_1`, and `_2` filename suffixes. Files without a mate suffix are treated as single-end only when no R2 file is present. Paired inputs must have both mates, with equal loaded read counts, in every round.

Value

A `selexprep_read_inputs` list with `sequences_by_round`, optional `paired_mate_streams`, `read_source`, and a file-level `DataFrame` named `files`. The first three elements can be passed directly to `run_selexprep()`, or use `run_selexprep_files()`.

Examples

```
path <- tempfile(fileext = '.fastq')
writeLines(c('@read_1', 'ACGT', '+', 'IIII'), path)
inputs <- read_selexprep_inputs(path, round = 0)
inputs$sequences_by_round
```

read_selexprep_manifest *Read a selexprep manifest JSON file*

Description

Reads and validates both the portable Python `selexprep_manifest_v1` format and the R-native `selexprep_manifest_v2` format.

Usage

```
read_selexprep_manifest(path)
```

Arguments

`path`	Path to a selexprep manifest JSON file.

Value

A validated `selexprep_manifest`.

Examples

```
p5 <- 'GGTAATACGACTCACTATAGGG'
p3 <- 'CCATGCATGCATGCATGCAT'
report <- selexprep_detect(list(round_00 = rep(paste0(p5, 'ACGTACGTACGTACGT', p3), 500)))
path <- tempfile(fileext = '.json')
write_selexprep_manifest(build_selexprep_manifest(report), path)
read_selexprep_manifest(path)
```

reverse_complement *Reverse-complement nucleotide sequences*

Description

This function reproduces the sequence-counting behavior of the Python implementation. It accepts DNA or RNA bases, keeps input case, maps U/u to A/a, and preserves ambiguous bases such as N/n. It is vectorized over `sequence`.

Usage

```
reverse_complement(sequence)
```

Arguments

`sequence`	A character vector of nucleotide sequences without missing values.

Value

A character vector of reverse-complemented sequences.

Examples

```
reverse_complement(c('ACGT', 'ACGU', 'ANNT'))
```

`run_selexprep` *Run the in-memory SELEX preprocessing workflow*

Description

Composes primer detection, extraction, sequence counting and quality control into a single Bioconductor result. The primary output is a `SummarizedExperiment` with one column per round and a sparse `counts` assay. The LibraryReport, extraction result and QC report are retained in `metadata()` for reproducibility and inspection.

Usage

```
run_selexprep(
  sequences_by_round,
  read_source = c("R1", "R2", "R1_AND_R2", "INTERLEAVED", "UNKNOWN"),
  paired_mate_streams = NULL,
  accession = NULL,
  sampling_seed = 42L,
  max_reads_per_round = NULL,
  low_total_reads = 10000L,
  primer_5p = NULL,
  primer_3p = NULL,
  extraction_mode = NULL
)
```

Arguments

`sequences_by_round`
: A named list of R1 pools, as accepted by `selexprep_detect()`, or a `selexprep_read_inputs` object.

`read_source`
: Input read layout.

`paired_mate_streams`
: Optional named R2 pools.

`accession`
: Optional public-study accession stored in metadata.

`sampling_seed`
: Seed forwarded to `selexprep_detect()`.

`max_reads_per_round`
: Optional detection subsampling cap.

`low_total_reads`
: QC threshold for per-round read depth.

primer_5p	Optional manually reviewed 5-prime primer override.
primer_3p	Optional manually reviewed 3-prime primer override.
extraction_mode	Optional manually reviewed extraction-mode override.

Details

Split-primer paired-end inputs return a valid zero-row experiment because full inserts cannot be counted without read merging. Their separate trimmed sides remain available in `metadata(result)$extraction`.

Value

A `SummarizedExperiment` with the LibraryReport, extraction and QC result in its metadata.

Examples

```
p5 <- 'GGTAATACGACTCACTATAGGG'
p3 <- 'CCATGCATGCATGCATGCAT'
pools <- list(round_00 = rep(paste0(p5, 'ACGTACGTACGTACGT', p3), 500))
run_selexprep(pools, low_total_reads = 0)
```

---

run_selexprep_files *Run selexprepR directly from FASTQ files*

Description

Convenience bridge combining `read_selexprep_inputs()` and `run_selexprep()`. File-level provenance is attached to `metadata(result)$input_files`.

Usage

```
run_selexprep_files(
  x,
  round = NULL,
  mate = NULL,
  max_reads_per_file = NULL,
  ...
)
```

Arguments

x	A character vector of FASTQ paths, a data frame-like object with `path` and `round_number` columns, or a non-dry-run `selexprep_fetch_result` returned by `fetch_selexprep_reads()`.
round	Optional vector of non-negative round numbers. It is required for character paths unless their vector names end in round numbers.

`mate` Optional vector containing `R1`/`R2` or `1`/`2`. When omitted, mate labels are inferred from filenames.

`max_reads_per_file` Optional positive maximum number of records read from each FASTQ file.

`...` Named arguments forwarded to `run_selexprep()`. The input sequence, paired-mate, and read-source arguments are supplied by this function and cannot be overridden.

Value

A `SummarizedExperiment` as returned by `run_selexprep()`.

Examples

```
experiment <- run_selexprep_files(
    c(round_00 = 'round_00.fastq.gz',
      round_01 = 'round_01.fastq.gz')
)
```

---

selexprep-classes *selexprepR result objects*

Description

The package uses lightweight S3 lists for decision records and reports, while sequence abundance data use the standard Bioconductor `SummarizedExperiment` container. These lists are deliberately explicit and serializable: they can be written to the versioned JSON contracts without loss of meaning.

Details

The list elements are stable public contracts:

- `selexprep_library_report` stores `primer_5p`, `primer_3p`, their variant evidence (`variants_5p`, `variants_3p`), `known_adapter_hits`, `extraction_mode`, `full_insert_recovered`, `read_source`, `required_action`, `orientation`, random-region length summaries (`n_length_mode`, `n_length_distribution`, `n_length_confidence`), primer match and position-consistency rates, the inference fraction and seed, overall `confidence`, `status`, and an optional `failure_reason`.
- `selexprep_extraction` stores `extraction_mode`, `full_insert_recovered`, primer-stripped `sequences_by_round`, named `input_reads` and `output_reads` counts, and the pre-trim forward/reverse/ambiguous `strand_distribution` for each round.
- `selexprep_qc` stores a `per_round` metrics `DataFrame`, a pairwise `round_similarity` `DataFrame`, a `flags` `DataFrame` whose `evidence` column contains structured lists, and the applied `settings`.
- `selexprep_manifest` stores the schema and package/runtime versions, accession and run identifiers, input/output SHA-256 maps, the embedded library report and its classification, parameters, stage durations, QC flags, and sampling seed. Readers accept portable Python v1 and R-native v2 schemas.

- `selexprep_inspection` stores the requested `accession`, BioProject ID, study title, library strategy/source, and a run-level `runs` `DataFrame` containing accessions, titles, sizes, FASTQ URLs, and checksums.
- `selexprep_fetch_result` stores the resolved run-level download `plan`, `downloaded_files`, and the `dry_run` status.
- `selexprep_read_inputs` stores round-aware R1 pools, optional R2 pools, the inferred `read_source`, and a file-level provenance `DataFrame`.
- `selexprep_demultiplex` extends `selexprep_read_inputs` with unassigned reads, per-round assignment summaries, and the validated barcode map.

S3 lists are used for these small decision and provenance records because their named elements map losslessly to the versioned JSON contracts. The high-dimensional assay remains a standard `SummarizedExperiment` with a sparse `Matrix` assay.

Value

A description of the result classes and their stable list elements.

Examples

```
p5 <- 'GGTAATACGACTCACTATAGGG'
p3 <- 'CCATGCATGCATGCATGCAT'
report <- selexprep_detect(list(round_00 = rep(paste0(p5, 'ACGTACGTACGTACGT', p3), 500)))
class(report)
```

---

selexprep_catalog *Query the bundled public HT-SELEX study catalog*

Description

Returns the curated catalog snapshot distributed with the package. The optional text query is matched case-insensitively against the BioProject ID, study title, and target, making it useful for discovery while leaving raw metadata values intact.

Usage

```
selexprep_catalog(query = NULL)
```

Arguments

`query`	Optional non-empty search string.

Value

An `S4Vectors::DataFrame` of curated study metadata.

Examples

```
selexprep_catalog('FGF-9')
```

selexprep_count *Count unique SELEX sequences*

Description

Counts the observed sequences in one primer-stripped SELEX round. The result is a `S4Vectors::DataFrame` compatible with the stable Python output schema: `sequence`, `reads`, `rank`, and `rpm`. The sequence column is a `Biostrings::BStringSet` so that both DNA and RNA alphabets can be carried without lossy coercion.

Usage

```
selexprep_count(sequences)
```

Arguments

`sequences`	A character vector or a `Biostrings::XStringSet` containing the primer-stripped sequences from one round.

Details

Ties in abundance are ordered lexicographically by sequence. The original Python implementation does not define a tie-breaker; making it explicit in R provides reproducible ranks across platforms.

Value

A `S4Vectors::DataFrame` with `sequence`, `reads`, `rank`, and `rpm` columns, ordered by decreasing read count.

Examples

```
selexprep_count(c('ACGT', 'ACGT', 'GGGG'))
```

selexprep_demultiplex *Demultiplex SELEX reads with user-supplied barcodes*

Description

Assigns reads to selection rounds from an equal-length barcode at the R1 5-prime end. Barcode sequences are supplied explicitly; they are never inferred. R2 mates follow their R1 assignment and remain untrimmed.

Usage

```
selexprep_demultiplex(
  sequences,
  barcodes,
  paired_mates = NULL,
  max_mismatches = 1L,
  trim_barcode = TRUE
)
```

Arguments

sequences	R1 sequences as a character vector or Biostrings::XStringSet.
barcodes	Named integer vector mapping barcode sequences to non-negative round numbers, for example c(AAAAA = 0L, TTTTT = 1L).
paired_mates	Optional R2 sequences in the same order as sequences.
max_mismatches	Maximum Hamming distance allowed from a barcode.
trim_barcode	Whether to remove the matched barcode from assigned R1 sequences.

Details

Barcodes must have pairwise Hamming distance of at least 2 * max_mismatches + 1, preventing ambiguous assignments. RNA U and DNA T are treated as equivalent during matching.

Value

A selexprep_demultiplex object. Its first four fields follow the selexprep_read_inputs contract and can be supplied directly to run_selexprep(). Additional fields contain unassigned reads, a summary, and the validated barcode map.

Examples

```
reads <- c("AAAAACCCCC", "AAAATGGGGG", "TTTTTACGTA", "GGGGGAAAAA")
demultiplexed <- selexprep_demultiplex(
    reads,
    c(AAAAA = 0L, TTTTT = 1L)
)
demultiplexed$summary
```

---

selexprep_detect *Detect SELEX library structure and constant regions*

Description

Detects 5' and 3' constant regions in the earliest supplied SELEX round, then verifies the call across rounds. The returned LibraryReport separates inferred biology, input layout, and required downstream action. When primer evidence is insufficient, it returns an explicit safe-failure report rather than guessing.

Usage

```
selexprep_detect(
  sequences_by_round,
  read_source = c("R1", "R2", "R1_AND_R2", "INTERLEAVED", "UNKNOWN"),
  paired_mate_streams = NULL,
  sampling_seed = 42L,
  max_reads_per_round = NULL
)
```

Arguments

sequences_by_round	A named list of character vectors or Biostrings::XStringSet objects. Each element is one sequencing round.
read_source	One of 'R1', 'R2', 'R1_AND_R2', 'INTERLEAVED', or 'UNKNOWN'.
paired_mate_streams	Optional named list containing the corresponding R2 sequences. Required only to detect split-primer paired-end layouts.
sampling_seed	Integer seed used when max_reads_per_round samples input reads.
max_reads_per_round	Optional positive cap on reads used per round.

Value

A selexprep_library_report.

Examples

```
p5 <- 'GGTAATACGACTCACTATAGGG'
p3 <- 'CCATGCATGCATGCATGCAT'
selexprep_detect(list(round_00 = rep(paste0(p5, 'ACGTACGTACGTACGT', p3), 500)))
```

---

selexprep_extract *Extract random regions from a SELEX library*

Description

Uses a selexprep_library_report to remove the inferred constant regions. The implementation operates directly on R sequence objects and preserves the explicit safe-failure behavior of the original pipeline: reports marked as unable to extract stop with an error unless the caller supplies a corrected report. In split-primer paired-end mode, the two trimmed sides remain separate; they are never silently merged.

Usage

```
selexprep_extract(
  sequences_by_round,
  library_report,
  paired_mate_streams = NULL,
  primer_5p = NULL,
  primer_3p = NULL,
  extraction_mode = NULL
)
```

Arguments

sequences_by_round	A named list of character vectors or Biostrings::XStringSet objects containing R1 sequences.
library_report	A selexprep_library_report from selexprep_detect() or read_library_report().
paired_mate_streams	Optional named R2 sequence pools, required for PAIRED_END_SPLIT_PRIMERS reports.
primer_5p	Optional manually reviewed 5-prime primer override.
primer_3p	Optional manually reviewed 3-prime primer override.
extraction_mode	Optional manually reviewed extraction-mode override.

Details

Full-primer matching follows the cutadapt defaults used by the Python pipeline: adapters may occur away from the read boundary, substitutions and indels are allowed up to a 10 percent error rate, and RNA U is matched as DNA T. The returned insert retains the alphabet and case of the input read.

Value

A selexprep_extraction. Full and single-primer modes store one Biostrings::BStringSet per round; paired split mode stores an r1 and r2 BStringSet for every round.

Examples

```
primer_5p <- 'GGTAATACGACTCACTATAGGG'
primer_3p <- 'CCATGCATGCATGCATGCAT'
bases <- c('A', 'C', 'G', 'T')
random_regions <- vapply(0:499, function(i) {
    paste0(bases[(i * 7 + 0:15 * 13) %% 4 + 1], collapse = '')
}, character(1))
reads <- paste0(primer_5p, random_regions, primer_3p)
report <- selexprep_detect(list(round_00 = reads))
selexprep_extract(list(round_00 = reads[seq_len(3L)]), report)
```

`selexprep_public_catalog` *Curated public HT-SELEX catalog*

Description

A frozen, offline snapshot of 240 public HT-SELEX deposits. Eight experimental fields were extracted independently by two language models, reconciled, and retained with a per-field curation status. The flat dataset intentionally omits evidence quotations; immutable links to the canonical provenance-rich JSON and its generation materials are stored in `S4Vectors::metadata(selexprep_public_catalog)`.

Usage

```
data(selexprep_public_catalog)
```

Format

An `S4Vectors::DataFrame` with 240 rows and 19 columns:

`bioproject_id` ENA BioProject accession or a platform-prefixed Figshare or Zenodo deposit identifier.

`source` `ena`, or the platform-prefixed Figshare or Zenodo deposit identifier used as the source key.

`study_title` Title recorded for the public study or deposit.

`study_type`, `study_type_curation` Curated experiment type and its reconciliation status.

`target`, `target_curation` Selection target and its reconciliation status.

`target_class`, `target_class_curation` Target class and its reconciliation status.

`chemistry`, `chemistry_curation` DNA or RNA library chemistry and its reconciliation status.

`n_random`, `n_random_curation` Reported random-region length and its reconciliation status.

`n_rounds`, `n_rounds_curation` Reported selection-round count and its reconciliation status.

`selection_format`, `selection_format_curation` Reported selection format and its reconciliation status.

`counter_selection`, `counter_selection_curation` Reported counter-selection conditions and their reconciliation status.

Curation statuses are `concordant`, `discordant`, `not_stated`, `verified`, `single_source:claude`, or `single_source:codex`.

Details

Missing values mean that a field was not stated in the curated sources. A `discordant` status means that both extracted values are retained in the corresponding value column, separated by `' || '`.

Source

The snapshot was migrated without modification from the original selexprep catalog at commit `b6792637ec9bed78f131e8f63e12e155f733e9ae`. See the object metadata and `inst/CATALOG_PROVENANCE.md` for immutable source links and terms.

`selexprep_qc` *Quality-control summary for a SELEX count experiment*

Description

Calculates depth, diversity, enrichment and sequence-length summaries for each round of a `SummarizedExperiment` returned by `run_selexprep()`. It also returns actionable flags for low depth, unstable random-region length, weak primer evidence and paired-end libraries that need read merging.

Usage

```
selexprep_qc(
  experiment,
  library_report = NULL,
  low_total_reads = 10000L,
  rarefaction_depth = 10000L,
  rarefaction_seed = 42L,
  kmer_size = 6L,
  kmer_top_sequences = 10000L
)
```

Arguments

`experiment`	A `SummarizedExperiment` with an assay named `'counts'` and a `sequence` column in `rowData`.
`library_report`	Optional `selexprep_library_report`. When omitted, a report stored in the experiment metadata is used when available.
`low_total_reads`	Minimum number of reads expected in every round.
`rarefaction_depth`	Maximum common depth used when comparing observed diversity across rounds.
`rarefaction_seed`	Integer seed for deterministic rarefaction.
`kmer_size`	Length of canonical k-mers used for between-round consistency diagnostics.
`kmer_top_sequences`	Maximum number of abundant sequences considered from each round for k-mer diagnostics.

Value

A `selexprep_qc` list with `per_round`, `round_similarity`, and `flags` DataFrames plus the applied `settings`.

Examples

```
p5 <- 'GGTAATACGACTCACTATAGGG'
p3 <- 'CCATGCATGCATGCATGCAT'
pools <- list(round_00 = rep(paste0(p5, 'ACGTACGTACGTACGT', p3), 500))
result <- run_selexprep(pools, low_total_reads = 0)
selexprep_qc(result)
```

---

write_library_report *Write a LibraryReport as deterministic JSON*

Description

Write a LibraryReport as deterministic JSON

Usage

```
write_library_report(report, path)
```

Arguments

report	A selexprep_library_report.
path	Output JSON path.

Value

The SHA-256 digest of the emitted UTF-8 JSON, invisibly.

Examples

```
p5 <- 'GGTAATACGACTCACTATAGGG'
p3 <- 'CCATGCATGCATGCATGCAT'
report <- selexprep_detect(list(round_00 = rep(paste0(p5, 'ACGTACGTACGTACGT', p3), 500)))
path <- tempfile(fileext = '.json')
write_library_report(report, path)
```

---

write_selexprep_counts *Write a selexprep count table*

Description

Writes the stable four-column count contract. Use TSV for a dependency-free, portable exchange format; Parquet is available when the optional arrow package is installed.

Usage

```
write_selexprep_counts(counts, path)
```

Arguments

counts	A count table returned by selexprep_count().
path	Output path ending in .tsv, .csv, .rds, or .parquet.

Value

path, invisibly.

Examples

```
counts <- selexprep_count(c('ACGT', 'ACGT', 'GGGG'))
path <- tempfile(fileext = '.tsv')
write_selexprep_counts(counts, path)
```

---

write_selexprep_manifest

Write a selexprep manifest as deterministic JSON

Description

Write a selexprep manifest as deterministic JSON

Usage

```
write_selexprep_manifest(manifest, path)
```

Arguments

manifest	A selexprep_manifest.
path	Output JSON path.

Value

The SHA-256 digest of the emitted UTF-8 JSON, invisibly.

Examples

```
p5 <- 'GGTAATACGACTCACTATAGGG'
p3 <- 'CCATGCATGCATGCATGCAT'
report <- selexprep_detect(list(round_00 = rep(paste0(p5, 'ACGTACGTACGTACGT', p3), 500)))
path <- tempfile(fileext = '.json')
write_selexprep_manifest(build_selexprep_manifest(report), path)
```

Index