Package: seqout (via r-universe)

November 26, 2026

Title Client for Querying the 'Seqout' Genomics Metadata Database

Version 0.2.8

Description Search and download genomics study metadata from 'Seqout' <https://seqout.org>, through its REST (Representational State Transfer) web service <https://seqout.org/api/> or its Parquet data dump <https://seqout.org/data>. 'Seqout' indexes the National Center for Biotechnology Information (NCBI) Gene Expression Omnibus (GEO) <https://www.ncbi.nlm.nih.gov/geo/> and Sequence Read Archive (SRA) <https://www.ncbi.nlm.nih.gov/sra>, the European Bioinformatics Institute (EBI) European Nucleotide Archive (ENA) <https://www.ebi.ac.uk/ena/browser/>, the DNA Data Bank of Japan (DDBJ) Sequence Read Archive (DRA) and Genomic Expression Archive (GEA) <https://www.ddbj.nig.ac.jp>, ArrayExpress <https://www.ebi.ac.uk/biostudies/arrayexpress>, and the Genome Sequence Archive (GSA) <https://ngdc.cncb.ac.cn/gsa/>.

License BSD_3_clause + file LICENSE

URL https://github.com/saketlab/seqout, https://seqout.org, https://seqout.org/cli/R/

BugReports https://github.com/saketlab/seqout/issues

Depends R (>= 4.1.0)

Imports cli, curl, httr2 (>= 1.0.0), methods, rlang, stats, tibble, tools, utils

Suggests DBI, dplyr, jsonlite, dbplyr, DESeq2, DT, duckdb, genevintage (>= 0.1.2), ggplot2, hdf5r, Matrix, Rsamtools, scales, SexSeek, Seurat, SeuratObject, SingleCellExperiment, SummarizedExperiment, testthat (>= 3.0.0), knitr, rhdf5filters, rmarkdown, withr

Config/testthat/edition 3

Encoding UTF-8

VignetteBuilder knitr

Config/roxygen2/version 8.1.1

NeedsCompilation no

Author Aniruddha Mukherjee [aut] (ORCID:
<https://orcid.org/0009-0008-1350-6787>), Mukesh Reddy [aut],
Saket Choudhary [aut, cre] (ORCID:
<https://orcid.org/0000-0001-5202-7633>), Saket Lab [cph]

Maintainer Saket Choudhary <saketc@iitb.ac.in>

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

Date/Publication 2026-11-26 01:52:39 UTC

RemoteUrl https://github.com/cran/seqout

RemoteRef HEAD

RemoteSha c40aade0d7f9c5661a47b66ed4728689f0d7854d

Contents

accession_kind 4
author 5
bind_counts 5
cache_table 7
citations 7
clear_cache 8
country_code_to_name 9
country_facets 10
country_projects 10
country_summary 12
counts_design 13
counts_samples 14
disease_aliases 15
disease_facets 16
disease_projects 16
disease_summary 19
download_bams 20
download_dump 21
download_runs 22
download_supplementary 23
file_role 24
filter_values 24
group_key 26
gsm_series 27
is_filtered 27
linked_geo 28
linked_study 29
longread_facets 29
longread_projects 30
longread_summary 32
map_to_ontology 33
matrices 35

microbe_detections . . . 35
ontology . . . 37
paper . . . 38
perturbation_facets . . . 39
perturbation_projects . . . 39
perturbation_summary . . . 42
project_longread_chemistry . . . 43
project_single_cell . . . 44
project_single_cell_status . . . 45
project_single_cell_summary . . . 46
query . . . 47
quick_annotation . . . 48
register_tables . . . 49
resolve_study . . . 49
run_files . . . 50
sample_microbes . . . 51
sample_search . . . 52
search_facets . . . 55
search_suggest . . . 56
seqout_close . . . 57
seqout_connect . . . 58
seqout_correct_names . . . 59
seqout_counts . . . 61
seqout_counts_files . . . 62
seqout_default . . . 63
seqout_fragments . . . 64
seqout_gene_names . . . 65
seqout_gene_report . . . 66
seqout_get . . . 67
seqout_matrix . . . 68
seqout_online . . . 69
seqout_sce . . . 70
seqout_search . . . 71
seqout_seurat . . . 73
seqout_sex . . . 74
seqout_sex_plot . . . 76
seqout_units . . . 76
single_cell_studies . . . 77
singlecell_facets . . . 78
singlecell_projects . . . 79
singlecell_summary . . . 81
spatial_facets . . . 82
spatial_projects . . . 82
spatial_summary . . . 85
summaries . . . 86
supplementary_files . . . 86
tables . . . 87
tissue_facets . . . 88

tissue_projects 88
tissue_summary 90

Index **92**

`accession_kind` *What an accession refers to*

Description

Uses the accession shape, so no request is made.

Usage

```
accession_kind(accession, archive = FALSE)

AccessionKind(accession, archive = FALSE)
```

Arguments

`accession`	An accession from any archive Seqout holds.
`archive`	Include the archive in the result.

Value

One of `"series"`, `"study"`, `"experiment"`, `"sample"`, `"run"`, `"biosample"` or `"submission"`, or `NA_character_` when the shape is not recognised. With `archive = TRUE`, a named character vector of `kind` and `archive`, the latter one of `"geo"`, `"sra"`, `"ena"`, `"ddbj"`, `"gsa"`, `"arrayexpress"`, `"gea"`, `"bioproject"` or `"biosample"`.

Examples

```
AccessionKind("GSE168652")
AccessionKind("SRR13927092")
AccessionKind("not-an-accession")

# PRJCA resolves to GSA under the PRJ shape
AccessionKind("PRJCA042384", archive = TRUE)
AccessionKind("PRJCA042384", archive = TRUE)[["archive"]]
```

author *Datasets linked to an author*

Description

Datasets named by an author's publications.

Usage

```
author(name, limit = 200, con = .con())

Author(name, limit = 200, con = .con())
```

Arguments

name	The author name as it appears in the publication record.
limit	Maximum datasets to return.
con	A seqout_connection. Defaults to the shared REST connection.

Value

A tibble of projects.

Examples

```
Author("Aviv Regev")
```

bind_counts *Bind counts matrices across samples*

Description

join = "inner", the default, keeps the features common to every matrix and warns when that drops any. Peak-by-cell matrices called per sample rarely share a feature space, so the difference can be large.

Usage

```
bind_counts(
  x,
  labels = NULL,
  max_cells = NULL,
  strict = FALSE,
  join = c("inner", "outer")
)

BindCounts(
  x,
  labels = NULL,
  max_cells = NULL,
  strict = FALSE,
  join = c("inner", "outer")
)
```

Arguments

x	A list of seqout_matrix objects, as matrices() returns.
labels	Column-name prefix per matrix. Defaults to the list names.
max_cells	Cap on columns kept per matrix, sampled at random. NULL keeps all; set a seed for reproducibility.
strict	Stop rather than warn when the feature sets differ. Ignored for join = "outer", which drops nothing.
join	"inner" for the shared features, "outer" for the union with absent features zero-filled.

Details

join = "outer" keeps the union and fills absent features with zero. A zero there means "not in this matrix", not "measured as zero".

Dense and sparse inputs may be mixed. Dense outer joins exceeding 5e7 elements use sparse storage to bound memory when features barely overlap.

Value

A matrix, dgCMatrix if the inputs were sparse.

Examples

```
## Not run:
# downloads two count files from NCBI GEO
counts <- SeqoutListCounts("GSE135251", cache_dir = tempdir())
picked <- c("GSM3998167", "GSM3998168")
merged <- BindCounts(Matrices(counts, sample = picked), labels = picked)

## End(Not run)
```

cache_table *Cache a remote view as a local DuckDB table*

Description

Queries against the cached table use local storage.

Usage

```
cache_table(table, con = .con())

CacheTable(table, con = .con())
```

Arguments

table	Table or view name, such as "geo_series".
con	A Parquet seqout_connection from seqout_connect().

Value

The local table name (invisibly).

Examples

```
## Not run:
# copies a remote Parquet table into the local DuckDB session
con <- SeqoutConnect("parquet", quiet = TRUE)
CacheTable("geo_series", con = con)
SeqoutClose(con)

## End(Not run)
```

citations *BibTeX for the papers behind a dataset*

Description

Assembles entries from the dataset publication record. type = "all" includes reanalysis papers.

Usage

```
citations(accession, type = "original", con = .con())

Citations(accession, type = "original", con = .con())
```

Arguments

accession	A project accession, or several. Several return one bibliography, each entry named for the dataset it came from.
type	"original", the default, or "all".
con	A seqout_connection. Defaults to the shared REST connection.

Details

A dataset with no linked paper returns character(0).

Reads REST. The dump lacks reanalysis papers and often publication dates.

Value

A character string of BibTeX entries, or character(0).

See Also

seqout_get(x)$pubs for the same papers as a tibble, and paper() to go the other way, from a paper to the datasets.

Examples

```
cat(Citations("GSE151530"))

# include reanalysis papers
cat(Citations("GSE168652", type = "all"))

writeLines(Citations("GSE151530"), file.path(tempdir(), "GSE151530.bib"))
```

---

clear_cache *Clear locally cached tables*

Description

Removes *_local tables created by cache_table().

Usage

```
clear_cache(con = .con())

ClearCache(con = .con())
```

Arguments

con	A Parquet seqout_connection from seqout_connect().

Value

Number of tables removed (invisibly).

Examples

```
## Not run:
con <- SeqoutConnect("parquet", quiet = TRUE)
CacheTable("geo_series", con = con)
ClearCache(con)
SeqoutClose(con)

## End(Not run)
```

`country_code_to_name` *Convert between country names and ISO 3166-1 alpha-3 codes*

Description

Exact lookups against a fixed table; spellings the table lacks give `NA`.

Usage

```
country_code_to_name(code)

country_name_to_code(name)

CountryCodeToName(code)

CountryNameToCode(name)
```

Arguments

`code`, `name`	Character vectors of alpha-3 codes (e.g. `"IND"`) or country names (e.g. `"India"`).

Value

A character vector the length of the input, `NA` where there is no match.

Examples

```
CountryCodeToName(c("IND", "USA", "XYZ"))
CountryNameToCode("India")
```

country_facets *Per-country facet counts*

Description

Study counts per organism, assay category, archive, year and single-cell status for one country. Each facet holds its top 40 values.

Usage

```
country_facets(code, con = .con())

CountryFacets(code, con = .con())
```

Arguments

code	Character. A two-letter country code.
con	A seqout_connection. Defaults to the shared REST connection; a Parquet connection reads the dump.

Value

A tibble with facet, value and studies columns.

See Also

country_projects().

Examples

```
f <- CountryFacets("IN")
f[f$facet == "organism", ]
```

country_projects *Studies submitted from a country*

Description

One row per study submitted from code. Filter values come from country_facets().

Usage

```
country_projects(
  code,
  organism = NULL,
  assay_l1 = NULL,
  source = NULL,
  has_fastq = NULL,
  has_sra = NULL,
  is_single_cell = NULL,
  q = NULL,
  sort = "year",
  order = "desc",
  limit = NULL,
  offset = 0,
  con = .con()
)

CountryProjects(
  code,
  organism = NULL,
  assay_l1 = NULL,
  source = NULL,
  has_fastq = NULL,
  has_sra = NULL,
  is_single_cell = NULL,
  q = NULL,
  sort = "year",
  order = "desc",
  limit = NULL,
  offset = 0,
  con = .con()
)
```

Arguments

code	Character. A two-letter country code.
organism, assay_l1, source	Character. Filter to one value from country_facets().
has_fastq, has_sra, is_single_cell	Logical filters.
q	Character. Case-insensitive substring match on title or study_accession.
sort	One of "n_samples", "n_experiments", "year", "title", "study_accession", "organism".
order	"desc", the default, or "asc".
limit	Maximum studies; NULL reads all, in pages of up to 200.
offset	Number of studies to skip.

con	A seqout_connection. Defaults to the shared REST connection; a Parquet connection reads the dump.

Value

A study tibble, with a total attribute for the filtered count before limit cut it.

See Also

country_summary(), country_facets().

Examples

```
# single-cell studies submitted from India, newest first
CountryProjects("IN", is_single_cell = TRUE, limit = 100)
```

---

country_summary *Corpus totals for one country's submitted studies*

Description

code is an ISO-3166-1 alpha-2 code (e.g. "US", "IN"); matching is case-insensitive. A code that isn't two letters, or matches nothing, 404s.

Usage

```
country_summary(code, con = .con())

CountrySummary(code, con = .con())
```

Arguments

code	Character. A two-letter country code.
con	A seqout_connection. Defaults to the shared REST connection; a Parquet connection reads the dump.

Value

A one-row tibble of totals.

See Also

country_facets() for the filter values, country_projects() for the studies themselves.

Examples

```
CountrySummary("US")
```

`counts_design` *The study's sample table, one row per sample*

Description

The submitter's characteristics spread into one column each. Reuses the sample record `seqout_counts()` fetched.

Usage

```
counts_design(counts)

CountsDesign(counts)
```

Arguments

`counts`	A `seqout_counts` tibble from `seqout_counts()`.

Value

A tibble, one row per sample of the study.

See Also

`counts_samples()` for the harmonised cohort fields.

Examples

```
counts <- SeqoutListCounts("GSE135251", cache_dir = tempdir())
col_data <- CountsDesign(counts)
```

`counts_samples` *Choose which samples of a study to read*

Description

Keeps samples matching `sample_search()` with a readable unit.

Usage

```
counts_samples(counts, ..., min_cell_count = 1L)

CountsSamples(counts, ..., min_cell_count = 1L)
```

Arguments

`counts`	A `seqout_counts` tibble from `seqout_counts()`, built on a GSE.
`...`	Filters for `sample_search()`, by name, such as `tissue = "liver"`. `study_accession` comes from `counts`.
`min_cell_count`	Smallest cell count to keep. Samples with no recorded count go too; `NULL` keeps everything.

Value

Matching samples, sorted by cells, with `unit` and `format`.

See Also

`seqout_counts()` for every unit, unfiltered.

Examples

```
## Not run:
counts <- seqout_counts("GSE182159")
liver <- counts_samples(counts, tissue = "liver", min_cell_count = 1000)
m <- seqout_matrix(counts, sample = liver$unit[1])

## End(Not run)
```

`disease_aliases` *Resolve a rare-disease name to its MONDO ids*

Description

Matches `q` against the GARD and NORD catalogue names and their aliases, so a colloquial or older name finds the MONDO ids `disease_projects()` can then filter on.

Usage

```
disease_aliases(q, limit = 20, con = .con())

DiseaseAliases(q, limit = 20, con = .con())
```

Arguments

`q`	Character. A disease name or part of one.
`limit`	Maximum matches to return.
`con`	A `seqout_connection`. Defaults to the shared REST connection; a Parquet connection reads the dump.

Value

A tibble with `display_name`, `alias`, `mondo_ids` and `sources` (list columns), with a `total` attribute for all matches and `truncated` for whether `total` exceeds `limit`.

See Also

`disease_projects()`.

Examples

```
DiseaseAliases("marfan")
```

disease_facets *Disease facet counts*

Description

Study counts per disease category/specialty/group (curated collections) or organism/assay/journal/country/year/status (a free-text term); see `disease_summary()` for the two modes.

Usage

```
disease_facets(collection, con = .con())

DiseaseFacets(collection, con = .con())
```

Arguments

`collection`	Character. `"rare"`, `"nord"`, or a free-text disease name/MONDO term (e.g. `"fatty liver disease"`).
`con`	A `seqout_connection`. Defaults to the shared REST connection; a Parquet connection reads the dump.

Value

A tibble with `facet`, `value` and `studies` columns.

See Also

`disease_projects()`.

Examples

```
DiseaseFacets("rare")
```

disease_projects *Studies matching a disease collection or a free-text disease term*

Description

One row per study, filtered and sorted; see `disease_summary()` for what distinguishes a curated collection from a free-text term. A curated row carries the catalogue match (`catalogue_diseases`, `mondo_ids`), per-scope sample counts, and stated/read-derived sex tallies; `category`/ `specialty`/`group` only apply to `"rare"`, `nord_type` only to `"nord"`. A term row carries the study's usual metadata plus `n_samples_with_disease`, the count of samples resolved to `collection` specifically (as opposed to `n_samples`, the study's total).

Usage

```
disease_projects(
  collection,
  category = NULL,
  specialty = NULL,
  group = NULL,
  nord_type = NULL,
  ancestry = NULL,
  inheritance = NULL,
  disease = NULL,
  assay_category = NULL,
  organism = NULL,
  assay_l1 = NULL,
  source = NULL,
  has_fastq = NULL,
  has_sra = NULL,
  is_single_cell = NULL,
  is_long_read = NULL,
  q = NULL,
  scope = "human_primary",
  sort = NULL,
  order = "desc",
  limit = NULL,
  offset = 0,
  con = .con()
)

DiseaseProjects(
  collection,
  category = NULL,
  specialty = NULL,
  group = NULL,
  nord_type = NULL,
  ancestry = NULL,
  inheritance = NULL,
  disease = NULL,
  assay_category = NULL,
  organism = NULL,
  assay_l1 = NULL,
  source = NULL,
  has_fastq = NULL,
  has_sra = NULL,
  is_single_cell = NULL,
  is_long_read = NULL,
  q = NULL,
  scope = "human_primary",
  sort = NULL,
  order = "desc",
```

```
  limit = NULL,
  offset = 0,
  con = .con()
)
```

Arguments

collection	Character. "rare", "nord", or a free-text disease name/MONDO term (e.g. "fatty liver disease").
category, specialty, group	Character. "rare"-only filters, from disease_facets().
nord_type	Character. "nord"-only filter, from disease_facets().
ancestry, inheritance, disease, assay_category	Character. Curated filters shared by both collections.
organism, assay_l1, source	Character. Filter to one value from disease_facets().
has_fastq, has_sra, is_single_cell, is_long_read	Logical filters.
q	Character. Case-insensitive substring match on title or study_accession.
scope	One of "human_primary" (default), "patient_derived_model", "cell_line", "all". Curated collections only.
sort	A curated collection sorts by "cells" (default), "n_samples", "n_diseases", "title", "study_accession", "assay_category", or "organism"; a free-text term sorts by "pub_date" (default), "n_samples", "n_experiments", "title", "study_accession", or "organism".
order	"desc", the default, or "asc".
limit	Maximum studies; NULL reads all, in pages of up to 200.
offset	Number of studies to skip. Curated collections only; a free-text term pages by cursor internally and errors if offset is set.
con	A seqout_connection. Defaults to the shared REST connection; a Parquet connection reads the dump.

Details

has_fastq/has_sra are NA when the study is absent from the download-links table (unknown, not unavailable); filtering on them accepts only TRUE for that reason. scope (curated only) defaults to "human_primary" (human material that is neither an immortalized line nor a derived model); use "patient_derived_model", "cell_line", or "all" to see what that excludes.

Value

A study tibble, with a total attribute for the filtered count before limit cut it.

See Also

disease_summary(), disease_facets().

Examples

```
DiseaseProjects("rare", category = "Cancers", limit = 100)
DiseaseProjects("fatty liver disease", organism = "Homo sapiens", limit = 10)
```

`disease_summary` *Corpus totals for a disease collection or a free-text disease term*

Description

`collection` is either a curated catalogue name (`"rare"` for NIH GARD, `"nord"` for NORD) or any other string, resolved as free text against the full MONDO ontology and expanded to its descendants. The two modes return different columns: a curated summary carries per-scope study counts and stated/read-derived sex tallies; a term summary carries `term`/`resolution`/`matched_labels` describing how the text resolved, plus `matched_samples` scoped to exactly the resolved MONDO ids (`samples` is every sample in a matching study, not just the ones that resolved).

Usage

```
disease_summary(collection, con = .con())

DiseaseSummary(collection, con = .con())
```

Arguments

`collection`	Character. `"rare"`, `"nord"`, or a free-text disease name/MONDO term (e.g. `"fatty liver disease"`).
`con`	A `seqout_connection`. Defaults to the shared REST connection; a Parquet connection reads the dump.

Details

A free-text term with no MONDO match 404s.

Value

A one-row tibble. Columns depend on `collection`; see Details.

See Also

`disease_facets()` for the filter values, `disease_projects()` for the studies themselves.

Examples

```
DiseaseSummary("rare")
DiseaseSummary("fatty liver disease")
```

`download_bams` *Download the alignment files a submitter sent*

Description

Fetches submitter BAMs aligned to their chosen reference.

Usage

```
download_bams(accession, dest_dir, quiet = FALSE, con = .con())

DownloadBAMs(accession, dest_dir, quiet = FALSE, con = .con())
```

Arguments

`accession`	Any accession Seqout holds. Resolved to its study, since the archive files alignments against one.
`dest_dir`	Directory to write into, created when missing. There is no default, so nothing lands in the working directory unasked.
`quiet`	Suppress progress messages.
`con`	A `seqout_connection`. Defaults to the shared REST connection.

Details

Requester-pays files are named with the command to fetch them.

Value

The paths of the downloaded files, invisibly.

See Also

`download_runs()` for the reads, and `seqout_get(x)$bams` to see what exists before fetching any of it.

Examples

```
## Not run:
SeqoutGet("SRP071083")$bams # requester-pays inventory
DownloadBAMs("SRP071083", "SRP071083")

## End(Not run)
```

`download_dump` *Download the Parquet dump*

Description

Fetches `.parquet` files for the Parquet backend. Existing files are skipped. HEAD sizes are reported before transfer when available.

Usage

```
download_dump(
  con = .con(),
  dest_dir,
  tables = NULL,
  overwrite = FALSE,
  quiet = FALSE
)

DownloadDump(
  con = .con(),
  dest_dir,
  tables = NULL,
  overwrite = FALSE,
  quiet = FALSE
)
```

Arguments

`con`	A `seqout_connection`. Either backend works: the files are read from the same place both point at.
`dest_dir`	Directory to write into, created when missing. There is no default, so nothing lands in the working directory unasked.
`tables`	Which tables to fetch. The default, `NULL`, fetches all of them. See `tables()`.
`overwrite`	Fetch a file even when it is already on disk.
`quiet`	Suppress progress messages.

Value

The paths of the downloaded files, invisibly.

See Also

`seqout_connect()` to read the result, and `cache_table()` to keep one table in local DuckDB storage.

Examples

```
## Not run:
# two common tables
DownloadDump(
  dest_dir = file.path(tempdir(), "seqout-dump"),
  tables = c("unified_metadata", "geo_series")
)

con <- SeqoutConnect("parquet", data_dir = file.path(tempdir(), "seqout-dump"))
Query("SELECT count(*) FROM unified_metadata", con = con)

## End(Not run)
```

`download_runs` *Download the read files of an accession*

Description

Runs download their own files; larger accessions download all resolved runs. Default mode order is fastq, full SRA, then SRA lite. Published checksums are verified.

Usage

```
download_runs(accession, dest_dir, mode = NULL, quiet = FALSE, con = .con())

DownloadRuns(accession, dest_dir, mode = NULL, quiet = FALSE, con = .con())
```

Arguments

`accession`	Any accession Seqout holds.
`dest_dir`	Directory to write into, created when missing. There is no default, so nothing lands in the working directory unasked.
`mode`	Which copy to take: `"fastq"`, `"sra"` (NCBI's full-quality copy) or `"sra_lite"` (the same reads with binned quality scores). `NULL` takes the first each run offers, in that order.
`quiet`	Suppress progress messages.
`con`	A `seqout_connection`. Defaults to the shared REST connection.

Value

The paths of the downloaded files, invisibly.

Examples

```
## Not run:
DownloadRuns("SRR12012336", "SRR12012336") # one run
DownloadRuns("SRP267291", "SRP267291") # every run of a study
DownloadRuns("SRP150719", "SRP150719", mode = "fastq") # insist on ENA fastq

## End(Not run)
```

`download_supplementary` *Download the supplementary files of an accession*

Description

Downloads processed files: matrices, annotations and archives. Samples stay scoped to their own files.

Usage

```
download_supplementary(accession, dest_dir, quiet = FALSE, con = .con())

DownloadSupplementary(accession, dest_dir, quiet = FALSE, con = .con())
```

Arguments

`accession`	Any accession Seqout holds.
`dest_dir`	Directory to write into, created when missing. There is no default, so nothing lands in the working directory unasked.
`quiet`	Suppress progress messages.
`con`	A `seqout_connection`. Defaults to the shared REST connection.

Value

The paths of the downloaded files, invisibly.

Examples

```
## Not run:
DownloadSupplementary("GSE168652", "GSE168652") # series files and every sample's
DownloadSupplementary("GSM8433846", "GSM8433846") # that one sample's
DownloadSupplementary("E-MTAB-11467", "E-MTAB-11467") # processed archives and raw

## End(Not run)
```

file_role *What a supplementary file is*

Description

Names the role from the filename, before download.

Usage

```
file_role(name)

FileRole(name)
```

Arguments

name	A file name or URL.

Value

One of "mtx", "barcodes", "features", "h5", "h5ad", "rds", "rcc", "table", "tar", "metadata" or "skip".

Examples

```
FileRole("GSM123_matrix.mtx.gz")
FileRole("GSM123_barcodes.tsv.gz")
FileRole("GSE1_cell_metadata.csv.gz")
FileRole("GSM1_fragments.tsv.gz")
FileRole("GSM4567420_01_NAS5.RCC.gz")
```

filter_values *The values a search filter can match*

Description

Enumerates values accepted by seqout_search(), with record counts.

Usage

```
list_library_strategies(con = .con())

list_instrument_models(con = .con())

list_journals(con = .con(), limit = 500)

list_centers(con = .con(), limit = 500)
```

```
list_organisms(con = .con(), common_names = FALSE)

list_assays(con = .con(), country = NULL)

list_platforms(con = .con())

ListAssays(con = .con(), country = NULL)

ListCenters(con = .con(), limit = 500)

ListInstrumentModels(con = .con())

ListJournals(con = .con(), limit = 500)

ListLibraryStrategies(con = .con())

ListOrganisms(con = .con(), common_names = FALSE)

ListPlatforms(con = .con())
```

Arguments

con	A seqout_connection. Defaults to the shared REST connection.
limit	The maximum number of values. Only list_journals() and list_centers() take it.
common_names	Add a common_name column, NA where none is on record. list_organisms() returns every organism recorded across archives.
country	Scope the assay counts to one country, by name. The default, NULL, counts over every archive.

Details

list_assays() returns both assay levels in a level column. These functions read REST; query() can count from the dump.

Value

A tibble of values and counts.

See Also

seqout_search().

Examples

```
ListLibraryStrategies()
```

```
# instrument_model values
ListInstrumentModels()

# connection-first call
SeqoutConnect("api") |> ListJournals(limit = 20)

# pass a value into search
top <- ListLibraryStrategies()$value[1]
SeqoutSearch("liver", library_strategy = top, limit = 10)
```

---

group_key *Shared key for the files of one 10x unit*

Description

CellRanger directories win; otherwise strip compression and role tokens. Matching stems place `matrix.mtx`, `barcodes.tsv`, and `features.tsv` together.

Usage

```
group_key(name)

GroupKey(name)
```

Arguments

`name`	A file name or path.

Value

The grouping key.

Examples

```
GroupKey("GSM123_pbmc_matrix.mtx.gz")
GroupKey("GSM123/filtered_feature_bc_matrix/matrix.mtx.gz")
```

gsm_series *Find the GEO series a sample belongs to*

Description

Find the GEO series a sample belongs to

Usage

```
gsm_series(gsm, con = .con())

GSMSeries(gsm, con = .con())
```

Arguments

gsm	A GEO sample accession (GSM).
con	A seqout_connection. Defaults to the shared REST connection; a Parquet connection reads the dump.

Value

The GSE accession, or NA_character_.

Examples

```
GSMSeries("GSM8994520")
```

is_filtered *CellRanger filtered output*

Description

Filtered output ranks before raw.

Usage

```
is_filtered(name)

IsFiltered(name)
```

Arguments

name	A file name.

Value

TRUE when the name marks filtered output.

Examples

```
IsFiltered("GSM1_filtered_feature_bc_matrix.h5")
IsFiltered("GSM1_raw_feature_bc_matrix.h5")
```

---

linked_geo *Find the series holding the processed files*

Description

Resolves a sequencing study to its GEO or ArrayExpress series.

Usage

```
linked_geo(accession, con = .con())

LinkedGEO(accession, con = .con())
```

Arguments

accession	A study accession (SRP, ERP, DRP, PRJ).
con	A seqout_connection. Defaults to the shared REST connection; a Parquet connection reads the dump.

Value

The linked series accession, or NA_character_.

Examples

```
LinkedGEO("SRP151530")
```

linked_study *Find the sequencing study linked to a series*

Description

GEO and ArrayExpress runs live under linked sequencing studies. GEA resolves through BioProject.

Usage

```
linked_study(accession, con = .con())

LinkedStudy(accession, con = .con())
```

Arguments

accession	A series accession (GSE, E-MTAB-N, E-GEAD-N).
con	A seqout_connection. Defaults to the shared REST connection; a Parquet connection reads the dump.

Value

The linked study accession, or NA_character_.

Examples

```
LinkedStudy("GSE168652")
```

longread_facets *Long-read facet counts*

Description

Study counts per technology, platform, instrument model, library strategy, organism, archive, chemistry and year.

Usage

```
longread_facets(con = .con())

LongreadFacets(con = .con())
```

Arguments

con	A seqout_connection. Defaults to the shared REST connection; a Parquet connection reads the dump.

Value

A tibble with facet, value and studies columns.

See Also

longread_projects().

Examples

```
f <- LongreadFacets()
f[f$facet == "technology", ]
```

---

longread_projects *Studies with a PacBio or Oxford Nanopore experiment, any archive*

Description

One row per study; a study mirrored in more than one archive counts once. long_read_only = TRUE had no other platform; FALSE is hybrid; NA has no experiment rows to judge by (GEO-only entries). Filter values come from longread_facets().

Usage

```
longread_projects(
  technology = NULL,
  platform = NULL,
  instrument_model = NULL,
  library_strategy = NULL,
  organism = NULL,
  archive = NULL,
  chemistry = NULL,
  assay_l1 = NULL,
  year = NULL,
  has_fastq = NULL,
  has_sra = NULL,
  long_read_only = NULL,
  has_exact_chemistry = NULL,
  single_cell = NULL,
  sort = "n_experiments",
```

```
  order = "desc",
  limit = NULL,
  offset = 0,
  con = .con()
)

LongreadProjects(
  technology = NULL,
  platform = NULL,
  instrument_model = NULL,
  library_strategy = NULL,
  organism = NULL,
  archive = NULL,
  chemistry = NULL,
  assay_l1 = NULL,
  year = NULL,
  has_fastq = NULL,
  has_sra = NULL,
  long_read_only = NULL,
  has_exact_chemistry = NULL,
  single_cell = NULL,
  sort = "n_experiments",
  order = "desc",
  limit = NULL,
  offset = 0,
  con = .con()
)
```

Arguments

technology, platform, instrument_model, library_strategy, organism, archive, chemistry, assay_l1	Character. Filter to one value from longread_facets().
year	Integer. Publication year.
has_fastq, has_sra, long_read_only, has_exact_chemistry	Logical filters.
single_cell	TRUE/FALSE to keep only studies with (or without) a single-cell assay; NULL (default) leaves both in.
sort	One of "n_experiments", "n_samples", "n_runs", "first_published", "title", "study_accession", "organism".
order	"desc", the default, or "asc".
limit	Maximum studies; NULL reads all, in pages of up to 200.
offset	Number of studies to skip.
con	A seqout_connection. Defaults to the shared REST connection; a Parquet connection reads the dump.

Details

`single_cell` keeps only studies also flagged single-cell (`is_single_cell`). The server has no such parameter, so this filters locally.

Value

A study tibble, with a `total` attribute for the filtered count before `limit` cut it.

See Also

`longread_summary()`, `longread_facets()`, `project_longread_chemistry()` for one study's per-run detail.

Examples

```
# long-read-only datasets
LongreadProjects(long_read_only = TRUE, limit = 10)

# long-read AND single-cell
LongreadProjects(single_cell = TRUE, limit = 10)

# Oxford Nanopore human studies, newest first
LongreadProjects(
  technology = "Oxford Nanopore", organism = "Homo sapiens",
  sort = "first_published", limit = 10
)
```

`longread_summary` *Corpus-wide long-read totals*

Description

Totals for studies with at least one PacBio or Oxford Nanopore experiment, across GEO, SRA, ENA, DDBJ DRA and GSA. A study mirrored in more than one archive counts once. `studies_hybrid` also sequenced on a short-read platform; `studies_long_read_only` did not; the remainder have no experiment rows to judge by (GEO-only entries).

Usage

```
longread_summary(con = .con())

LongreadSummary(con = .con())
```

Arguments

con	A seqout_connection. Defaults to the shared REST connection; a Parquet connection reads the dump.

Value

A one-row tibble of totals.

See Also

longread_facets() for the filter values, longread_projects() for the studies themselves.

Examples

```
LongreadSummary()
```

---

map_to_ontology *Map a column of labels to ontology identifiers*

Description

Adds <column>_ontology_id columns with comma-separated CURIEs. Missing or empty labels become NA.

Usage

```
map_to_ontology(
  x,
  columns,
  ontology = NULL,
  use_synonyms = FALSE,
  max_hops = 1,
  con = .con()
)

MapToOntology(
  x,
  columns,
  ontology = NULL,
  use_synonyms = FALSE,
  max_hops = 1,
  con = .con()
)
```

Arguments

x	A data frame to copy and annotate.
columns	Character. The name of one column, or of several.
ontology	Keep the identifiers of one source only, for example "CL" for cell types or "UBERON" for anatomy. The default keeps all of them.
use_synonyms	Let a label with no identifier of its own take the identifiers of its synonyms. FALSE by default.
max_hops	How far to walk the synonym links, 1 to 4. Read only when use_synonyms is TRUE.
con	A seqout_connection. Defaults to the shared REST connection.

Details

Only the identifiers of the label itself are read. Under use_synonyms, synonym IDs can be narrower; own IDs win.

Reads REST. The ontology graph is absent from the dump.

Value

x with one new character column for each column that you named.

See Also

ontology() for what the graph holds about one term.

Examples

```
meta <- data.frame(celltype = c("T cell", "hepatocyte", "HeLa"))
MapToOntology(meta, "celltype")

# Cell Ontology IDs
MapToOntology(meta, "celltype", ontology = "CL")

# allow synonym IDs
MapToOntology(meta, "celltype", use_synonyms = TRUE)
```

matrices *Read every preferred unit*

Description

Broken units are skipped with a warning.

Usage

```
matrices(counts, sample = NULL)

Matrices(counts, sample = NULL)
```

Arguments

counts	A seqout_counts tibble from seqout_counts().
sample	Unit labels or sample accessions to read. NULL reads every preferred unit.

Value

A named list of seqout_matrix objects, keyed by unit label.

Examples

```
## Not run:
# downloads two count files from NCBI GEO
counts <- SeqoutListCounts("GSE135251", cache_dir = tempdir())
mats <- Matrices(counts, sample = c("GSM3998167", "GSM3998168"))
names(mats)

## End(Not run)
```

microbe_detections *The microbial detections behind a cohort, one row each*

Description

Flattens the microbes list column from sample_search() into one row per (sample, organism, run).

Usage

```
microbe_detections(
  x,
  validated_only = FALSE,
  columns = c("organism", "class", "kingdom", "breadth_frac", "kmer_mass")
)

MicrobeDetections(
  x,
  validated_only = FALSE,
  columns = c("organism", "class", "kingdom", "breadth_frac", "kmer_mass")
)
```

Arguments

x	A sample tibble from sample_search(), carrying microbes.
validated_only	Keep only detections that pass the stored gates.
columns	Detection fields to keep. The default set covers the organism, its class and the two evidence measures.

Value

A tibble of sample plus the requested detection fields, empty when nothing was detected.

See Also

sample_search() for the samples, sample_microbes() for every detection in one sample.

Examples

```
hpv <- SampleSearch(
  disease_ontology_id = "MONDO:0002974", microbe = "HPV", limit = 10
)
d <- MicrobeDetections(hpv)
sort(table(d$organism), decreasing = TRUE)

# only the detections that pass the gates
MicrobeDetections(hpv, validated_only = TRUE)
```

ontology *What the ontology graph knows about a term*

Description

Looks up one term in the graph used for search expansion. Returns source IDs, synonyms and direct children.

Usage

```
ontology(term, max_hops = 2, children = TRUE, con = .con())

Ontology(term, max_hops = 2, children = TRUE, con = .con())
```

Arguments

term	The word or phrase to look up. Case does not matter.
max_hops	How far to walk the synonym links, 1 to 4.
children	Set FALSE to skip the children, which is much cheaper.
con	A seqout_connection. Defaults to the shared REST connection.

Details

max_hops bounds the walk over the synonym links only. Children are always the direct children of the resulting synonym cluster, at any max_hops.

A term absent from the graph returns an empty tibble.

Reads REST. The ontology graph is absent from the dump.

Value

A tibble with one row for the term, one for each synonym and one for each child. Columns: relation, name, xrefs (a list column of source CURIEs) and has_children. The synonym_total, children_truncated and max_hops attributes report what the server capped.

See Also

seqout_search(), which expands a plain query through this graph.

Examples

```
Ontology("liver")

# The identifiers alone, without the hierarchy query
Ontology("breast cancer", children = FALSE)$xrefs

# Terms one level down that expand further
```

```
onto <- Ontology("liver")
onto[onto$relation == "child" & onto$has_children, "name"]
```

---

paper *Find projects linked to a publication*

Description

Find projects linked to a publication

Usage

```
paper(id = NULL, pmid = NULL, doi = NULL, con = .con())

Paper(id = NULL, pmid = NULL, doi = NULL, con = .con())
```

Arguments

id	Bare PubMed ID (digits) or DOI starting 10.. URLs error.
pmid	PubMed ID.
doi	DOI.
con	A seqout_connection. Defaults to the shared REST connection.

Value

A tibble of linked projects, empty when the publication is unknown.

Examples

```
Paper("34764296")
Paper(doi = "10.1038/s41467-021-26864-x")
```

perturbation_facets *Perturbation-study facet counts*

Description

Study counts per perturbation type, confidence, data availability, genetic tool, perturbation method, compound, organism, tissue, readout assay and year. Each facet holds its top 100 values.

Usage

```
perturbation_facets(con = .con())

PerturbationFacets(con = .con())
```

Arguments

con	A seqout_connection. Defaults to the shared REST connection; a Parquet connection reads the dump.

Value

A tibble with facet, value and studies columns.

See Also

perturbation_projects().

Examples

```
f <- PerturbationFacets()
f[f$facet == "perturbation_method", ]
```

perturbation_projects *Single-cell perturbation studies*

Description

One row per single-cell study with genetic (CRISPR knockout, CRISPRi, CRISPRa, RNAi, ORF, base or prime editing), chemical (drug or biologic) or other-stimulus perturbation evidence, with the readout assay and whether a counted matrix, raw FASTQ, both or neither is available.

Usage

```
perturbation_projects(
  perturbation_type = NULL,
  confidence = NULL,
  min_confidence = NULL,
  data_availability = NULL,
  genetic_subtype = NULL,
  perturbation_method = NULL,
  compound = NULL,
  readout_assay = NULL,
  cell_line = NULL,
  organism = NULL,
  tissue = NULL,
  year = NULL,
  has_matrix = NULL,
  has_fastq = NULL,
  has_control_arm = NULL,
  is_pooled = NULL,
  is_long_read = NULL,
  q = NULL,
  sort = "confidence",
  order = "desc",
  limit = NULL,
  offset = 0,
  con = .con()
)

PerturbationProjects(
  perturbation_type = NULL,
  confidence = NULL,
  min_confidence = NULL,
  data_availability = NULL,
  genetic_subtype = NULL,
  perturbation_method = NULL,
  compound = NULL,
  readout_assay = NULL,
  cell_line = NULL,
  organism = NULL,
  tissue = NULL,
  year = NULL,
  has_matrix = NULL,
  has_fastq = NULL,
  has_control_arm = NULL,
  is_pooled = NULL,
  is_long_read = NULL,
  q = NULL,
  sort = "confidence",
  order = "desc",
```

```
  limit = NULL,
  offset = 0,
  con = .con()
)
```

Arguments

perturbation_type	One of "genetic", "chemical", "both", "other".
confidence	One of "high", "medium", "low"; matches that tier exactly.
min_confidence	One of "high", "medium", "low"; matches that tier and stronger, so "medium" keeps high and medium.
data_availability	One of "both", "matrix_only", "fastq_only", "neither".
genetic_subtype, perturbation_method, compound, readout_assay, cell_line, organism, tissue	Character. Filter to one value from perturbation_facets().
year	Integer. Publication year.
has_matrix, has_fastq, has_control_arm, is_pooled, is_long_read	Logical filters.
q	Character. Case-insensitive substring match on title or study_accession.
sort	One of "confidence", the default, "n_samples", "n_cells", "year", "title", "study_accession", "organism".
order	"desc", the default, or "asc".
limit	Maximum studies; NULL reads all, in pages of up to 200.
offset	Number of studies to skip.
con	A seqout_connection. Defaults to the shared REST connection; a Parquet connection reads the dump.

Details

Detection is rule-based, from named methods in the study's own text (Perturb-seq, CROP-seq, ECCITE-seq, Mosaic-seq, TAP-seq, Spear-ATAC, sci-Plex, MIX-Seq and others), guide-library and pooled-screen wording, a control-arm design test on sample treatment values, and a ChEBI compound lexicon.

confidence is "high", "medium" or "low". High needs a named method, a guide library with screen text, or several compound treatments beside a control arm. Medium is corroborated by a second signal. Low is a single weak signal and includes false positives, so start from min_confidence = "medium" when you want studies to rely on. evidence lists the signals behind each row. perturbation_type is "genetic", "chemical", "both" or "other" (a cytokine or stimulation design with a control). Germline knockout and transgenic models are not evidence, and doxycycline or tamoxifen used as Cre or Tet inducers never count as the perturbation.

compounds are ChEBI-matched names from sample treatment values. They are empty when the drug arms live only in supplementary files or cell barcodes, as in sci-Plex, so a named method with no compounds is expected. data_availability is "both", "matrix_only", "fastq_only" or "neither", from has_matrix and has_fastq. Filter values come from perturbation_facets().

Value

A study tibble, with a `total` attribute for the filtered count before `limit` cut it.

See Also

`perturbation_summary()`, `perturbation_facets()`, `seqout_counts()` to read the matrices of a study with `has_matrix`.

Examples

```
# studies to rely on that have both a matrix and FASTQ
PerturbationProjects(
  min_confidence = "medium", data_availability = "both", limit = 100
)

# CRISPRi screens in human
PerturbationProjects(
  perturbation_type = "genetic", genetic_subtype = "CRISPRi",
  organism = "Homo sapiens", min_confidence = "medium", limit = 10
)

# drug studies that name trametinib
PerturbationProjects(compound = "trametinib", has_matrix = TRUE, limit = 10)
```

`perturbation_summary` *Perturbation-study corpus totals*

Description

Totals for single-cell studies with genetic or chemical perturbation evidence, the same population `perturbation_projects()` lists. `studies_high_medium` counts the studies whose evidence is strong enough to rely on; `studies` also counts weak single-signal calls.

Usage

```
perturbation_summary(con = .con())

PerturbationSummary(con = .con())
```

Arguments

`con`	A `seqout_connection`. Defaults to the shared REST connection; a Parquet connection reads the dump.

Value

A one-row tibble of totals.

See Also

`perturbation_facets()` for the filter values, `perturbation_projects()` for the studies themselves.

Examples

```
PerturbationSummary()
```

`project_longread_chemistry` *Get PacBio/Oxford Nanopore chemistry calls for a study*

Description

Every PacBio/Oxford Nanopore run in the study, independent of single-cell status and Pentimento linkage. `chemistry_confidence` is `exact` when read from the submitted BAM's own header, `declared` (ONT only) when parsed from the depositor's own protocol text, `bucket` when inferred from `instrument_model` alone, `unknown` otherwise.

Usage

```
project_longread_chemistry(accession, con = .con())

ProjectLongreadChemistry(accession, con = .con())
```

Arguments

`accession`	A study or series accession, or one resolving to a study.
`con`	A `seqout_connection`. Defaults to the shared REST connection; a Parquet connection reads the dump.

Value

A run tibble, empty when the study has no long-read runs.

See Also

`project_single_cell()`, whose `longread_chemistry` attribute answers the same question restricted to the study's first page of samples.

Examples

```
ProjectLongreadChemistry("GSE297547")
```

`project_single_cell` *Per-sample cell and gene counts for a study*

Description

One row per sample with matrix dimensions and read-derived calls.

Usage

```
project_single_cell(accession, limit = NULL, offset = 0, con = .con())

ProjectSingleCell(accession, limit = NULL, offset = 0, con = .con())
```

Arguments

`accession`	A study or series accession, or one resolving to a study.
`limit`	Maximum samples; `NULL` reads all, in pages of up to 1000.
`offset`	Number of samples to skip.
`con`	A `seqout_connection`. Defaults to the shared REST connection; a Parquet connection reads the dump.

Details

`cells` counts matrix columns; for unfiltered 10x matrices these are barcodes, so the number is an upper bound and sums overcount.
`has_*_reads = NA` means unscreened; `FALSE` means screened with no hit. `flags` preserves the same distinction.

Value

A sample tibble, empty when the study is unprocessed. The `study` and `n_samples_total` attributes carry the study row and its total. The `longread_chemistry` attribute carries every PacBio/Oxford Nanopore run in the study (empty, not `NULL`, when there are none); the server only computes it when `offset = 0`.

See Also

`sample_microbes()` for the per-organism detections behind the viral and bacterial flags, `project_longread_chemistry()` to read the same run list without an `offset` argument.

Examples

```
sc <- ProjectSingleCell("GSE168652")
sc[, c("sample_accession", "cells", "genes")]
attr(sc, "study")$study_cells
attr(sc, "longread_chemistry")
```

`project_single_cell_status` *Determine if a study is single-cell*

Description

`kind` describes the evidence behind the call:

`matrix_and_reads` counts and reads, both confirmed

`matrix_reads_unscanned` counts plus linked FASTQ no read-derived call has confirmed yet

`matrix_only` counts, no linked reads

`reads_only` no parsed matrix

Usage

```
project_single_cell_status(accession, con = .con())

ProjectSingleCellStatus(accession, con = .con())
```

Arguments

`accession`	A study or series accession, or one resolving to a study.
`con`	A `seqout_connection`. Defaults to the shared REST connection; a Parquet connection reads the dump.

Value

A one-row tibble, or an empty one when the accession is unknown to both the unified catalogue and the Pentimento.

See Also

`project_single_cell_summary()` for the study-level rollup, `project_single_cell()` for the per-sample counts.

Examples

```
st <- ProjectSingleCellStatus("GSE168652")
st$single_cell_status
st$n_independent_evidence
```

`project_single_cell_summary` *Study-level single-cell summary: cells, reads scanned, microbes*

Description

Study-level single-cell summary: cells, reads scanned, microbes

Usage

```
project_single_cell_summary(accession, con = .con())

ProjectSingleCellSummary(accession, con = .con())
```

Arguments

`accession`	A study or series accession, or one resolving to a study.
`con`	A `seqout_connection`. Defaults to the shared REST connection; a Parquet connection reads the dump.

Value

A one-row tibble, empty when the study is not in the Pentimento.

See Also

`project_single_cell_status()` when you need an answer even for a study absent from the Pentimento.

Examples

```
s <- ProjectSingleCellSummary("GSE168652")
s$study_cells
s$microbe_measured
```

query *Run arbitrary SQL on the Seqout DuckDB connection*

Description

The DuckDB database includes remote views and cached local tables.

Usage

```
query(sql, params = NULL, con = .con())

Query(sql, params = NULL, con = .con())
```

Arguments

sql	SQL query to execute.
params	Parameters for a parameterised query.
con	A Parquet seqout_connection from seqout_connect().

Value

A tibble with query results.

Examples

```
## Not run:
con <- SeqoutConnect("parquet")
Query("SELECT accession, title FROM geo_series LIMIT 10", con = con)

SeqoutDefault(con) # or make it the session default
Query("
  SELECT dominant_scientific_name AS organism, count(*) AS n
  FROM unified_metadata
  WHERE dominant_scientific_name IS NOT NULL
  GROUP BY organism
  ORDER BY n DESC
  LIMIT 20
")

## End(Not run)
```

`quick_annotation` *Label clusters by their strongest marker set*

Description

Scores markers per cell, averages by cluster, and labels by the top set. Marker sets with no genes in `x` are dropped with a warning.

Usage

```
quick_annotation(x, clusters, markers, normalize = "auto")

QuickAnnotation(x, clusters, markers, normalize = "auto")
```

Arguments

`x`	A features by cells matrix, as `SeqoutMatrix(...)$X` and `bind_counts()` return.
`clusters`	Cluster assignment per cell, one entry per column of `x`.
`markers`	A named list of marker gene vectors, one per cell type.
`normalize`	Scale each cell to 10,000 counts and `log1p()`. `"auto"` does it when `x` holds whole numbers.

Details

Scores are unscaled; housekeeping-heavy sets can win.

Value

A character vector of labels, named by cluster, with the cluster by marker-set score matrix attached as attribute `scores`.

Examples

```
## Not run:
markers <- list(
  neuron = c("Neurod2", "Tbr1"),
  microglia = c("Cx3cr1", "C1qa")
)
expr <- Seurat::GetAssayData(obj, assay = "RNA", slot = "data") # Seurat v4
expr <- SeuratObject::LayerData(Seurat::JoinLayers(obj), layer = "data") # v5

labels <- QuickAnnotation(expr, obj$seurat_clusters, markers)
obj$celltype <- unname(labels[as.character(obj$seurat_clusters)])

## End(Not run)
```

register_tables *Register the remote Parquet views*

Description

Views are lazy; catalog reads need them registered first.

Usage

```
register_tables(con = .con(), tables = NULL, progress = interactive())

RegisterTables(con = .con(), tables = NULL, progress = interactive())
```

Arguments

con	A Parquet seqout_connection from seqout_connect().
tables	Which tables to register. Defaults to all of them.
progress	Show a progress bar. Defaults to TRUE interactively.

Value

The names of every registered view, invisibly.

Examples

```
## Not run:
con <- SeqoutConnect("parquet")
RegisterTables(con, "unified_metadata")
dplyr::tbl(con$db, "unified_metadata")

# all views
SeqoutConnect("parquet") |> RegisterTables()

## End(Not run)
```

resolve_study *Find the study an accession belongs to*

Description

Resolves a run, experiment or sample to its study or series.

Usage

```
resolve_study(accession, con = .con())

ResolveStudy(accession, con = .con())
```

Arguments

accession	A run, experiment or sample accession.
con	A seqout_connection. Defaults to the shared REST connection; a Parquet connection reads the dump.

Value

The study accession, or NA_character_ when nothing links back.

Examples

```
resolve_study("SRR13927092")
```

---

run_files *The files each run would download as, without downloading them*

Description

One row per file: a paired run gives two. Uses the same choice of copy as download_runs(), so this is what that call would fetch.

Usage

```
run_files(x, mode = NULL, con = .con())

RunFiles(x, mode = NULL, con = .con())
```

Arguments

x	A runs table (e.g. seqout_get(acc)$runs) or any accession Seqout holds.
mode	Which copy to take: "fastq", "sra" (NCBI's full-quality copy) or "sra_lite" (the same reads with binned quality scores). NULL takes the first each run offers, in that order.
con	A seqout_connection. Defaults to the shared REST connection.

Value

A tibble with run, mode, url, name (the file name on disk), bytes and md5; bytes/md5 are NA where the archive publishes none. Runs with no downloadable copy are left out.

Examples

```
files <- RunFiles("SRP148597", mode = "fastq")
sum(files$bytes, na.rm = TRUE) / 1e9 # GB to fetch
"SRR7191948" |> RunFiles()
```

`sample_microbes` *Microbial reads found in a sample*

Description

Returns one row per organism, ordered by k-mer mass.

Usage

```
sample_microbes(
  accession,
  kind = c("all", "viral", "bacterial", "both"),
  min_breadth = NULL,
  min_kmer_mass = NULL,
  validated_only = FALSE,
  include_background = FALSE,
  detail = FALSE,
  limit = 500,
  con = .con()
)

SampleMicrobes(
  accession,
  kind = c("all", "viral", "bacterial", "both"),
  min_breadth = NULL,
  min_kmer_mass = NULL,
  validated_only = FALSE,
  include_background = FALSE,
  detail = FALSE,
  limit = 500,
  con = .con()
)
```

Arguments

`accession`	A GSM, or the archive sample accession behind one.
`kind`	Detection class: `"viral"`, `"bacterial"`, `"both"` or `"all"`.
`min_breadth`	Keep only detections covering at least this fraction of the reference, overriding the stored gate.

min_kmer_mass	Keep only detections at or above this weighted k-mer mass.
validated_only	Keep only detections that pass the stored gates.
include_background	Include reagent kitome and skin flora, which are excluded by default.
detail	Return per-run rows; the organism view stays on the by_organism attribute and costs no second request.
limit	The maximum number of detection rows to read, at most 1000.
con	A seqout_connection. Defaults to the shared REST connection; a Parquet connection reads the dump.

Details

Panel-unitig alignments lack raw-read, UMI and replicate confirmation. measurable = FALSE or NA means unscreened and triggers a warning.

kmer_mass weights k-mers within a run, so it carries no absolute scale. n_unitigs counts mapped unitigs.

Spike-in, negative-control and background classes are excluded from totals.

Value

An organism tibble, or a run-by-organism tibble when detail = TRUE. measurable, n_runs, totals, by_kingdom, control_kingdoms and the unpicked view are carried as attributes.

See Also

project_single_cell() for the study-wide flags.

Examples

```
m <- SampleMicrobes("GSM5155196", kind = "viral")
m[, c("organism", "n_unitigs", "max_breadth_frac")]
attr(m, "measurable")
attr(m, "detections")
```

---

sample_search *Search samples across every project*

Description

Searches harmonised sample metadata. Unharmonised samples are absent.

Usage

```
sample_search(
  filters = NULL,
  ...,
  single_cell = NULL,
  include_descendants = TRUE,
  sort = "sample",
  order = "asc",
  limit = NULL,
  con = .con()
)

SampleSearch(
  filters = NULL,
  ...,
  single_cell = NULL,
  include_descendants = TRUE,
  sort = "sample",
  order = "asc",
  limit = NULL,
  con = .con()
)
```

Arguments

filters	Filters as a named list. Names in ... override it.
...	The filters, by name, from the set above. At least one is required, here or in filters; an unfiltered call returns the whole corpus.
single_cell	How to decide a sample is single-cell: "assay" reads assay text, "flag" reads assay_is_single_cell, "either" accepts both. NULL, the default, does not filter. "flag" (and single_cell_only) drops samples with NA flags. "assay" and "either" read every page and filter locally.
include_descendants	Expand an ontology filter through the ontology graph, so disease_ontology_id = "MONDO:0005061" also matches the subtypes of that term. TRUE by default. Set FALSE for the exact term only.
sort	"sample" (the default), "study_accession", "age_days", "cell_count" or "gene_count".
order	"asc", the default, or "desc".
limit	The maximum number of samples. The default, NULL, returns every match.
con	A seqout_connection. Defaults to the shared REST connection; a Parquet connection reads the dump.

Details

Substring `tissue`, `disease`, `cell_type`, `assay`, `assay_category`, `phenotype`, `treatment`, `development_stage`, `sample_type`, `genetic_modification`, `strain`, `cell_line`, `ethnicity`, `tissue_primary_site`. `"liver"` matches `"liver, left lobe"`.

Exact `organism`, `sex`, `taxid`, `study_accession`. Exact matching keeps `"male"` from matching `"female"`.

Harmonised ontology ID `disease_ontology_id`, `tissue_ontology_id`, `cell_type_ontology_id`, `assay_ontology_id`, `development_stage_ontology_id`. Give a CURIE such as `"MONDO:0005061"`. See `include_descendants`.

Range `age_min_years`, `age_max_years`, `min_cell_count`, `max_cell_count`, `min_gene_count`, `max_gene_count`. An age filter excludes a sample without a recorded age, so `age_min_years = 0` means "has a recorded age".

Matrix `has_matrix` keeps samples with a parsed cell-by-gene matrix.

Read-derived `single_cell_only` (but see `single_cell`), `has_viral_reads`, `has_bacterial_reads`, `hpv_type`, `microbe`, `microbe_class`, `microbe_min_breadth`, `microbe_min_kmer_mass`, `microbe_validated_only`.

A `microbe*` filter keeps samples with matching detections and attaches them.

Requires REST. The harmonised sample table is absent from the Parquet dump.

Value

A sample tibble with `total` and `filters` attributes.

See Also

`sample_microbes()` for every detection in one sample, `seqout_search()` when the answer should be projects.

Examples

```
# single-cell samples by assay text
SampleSearch(tissue = "liver", disease = "steato", single_cell = "assay", limit = 10)

# female human liver samples over 50
SampleSearch(
  organism = "Homo sapiens", sex = "female",
  tissue = "liver", age_min_years = 50, limit = 10
)

# HPV references use HPV16, HPV18, and related names
# "papillomavirus" matches nothing
hpv <- SampleSearch(
  tissue = "cervix", microbe = "HPV",
  sort = "cell_count", order = "desc", limit = 10
)
attr(hpv, "total")

# filters built in code
```

```
list(tissue = "cervix", microbe = "HPV") |> SampleSearch(limit = 10)

# ontology term and subtypes
SampleSearch(disease_ontology_id = "MONDO:0005061", limit = 10)

# exact term
SampleSearch(
  disease_ontology_id = "MONDO:0005061",
  include_descendants = FALSE, limit = 10
)
```

`search_facets` *Count the matches of a search*

Description

Groups the full `seqout_search()` match set by facet.

Usage

```
search_facets(
  query,
  ...,
  structured = FALSE,
  exclude_ontology = NULL,
  con = .con()
)

SearchFacets(
  query,
  ...,
  structured = FALSE,
  exclude_ontology = NULL,
  con = .con()
)
```

Arguments

`query`	Character. The text to count the matches of. Required.
`...`	Filters, by name, to count *within*: `db`, `organism`, `country`, `library_strategy`, `library_source`, `instrument_model`, `platform`, `journal`, `multi_platform`, `year_from`, `year_to` and `case_sensitive`.
`structured`	Read `query` as a boolean expression, as `seqout_search()`.
`exclude_ontology`	Ontologies to keep out of the query expansion. See `seqout_search()`.
`con`	A `seqout_connection`. Defaults to the shared REST connection; a Parquet connection reads the dump.

Details

Values order by `score`, the summed match rank. `score` is 0 without a query.

Value

A tibble of `facet`, `value`, `count` and `score`, with `total` and `max_rank` attributes.

See Also

`seqout_search()` for the results themselves, `search_suggest()` when a query returns nothing.

Examples

```
f <- SearchFacets("liver cancer")
attr(f, "total")

# organism facet
f[f$facet == "organism", ]

# count within a narrower set
SearchFacets("liver cancer", organism = "Homo sapiens")
```

---

search_suggest *Suggest spelling corrections for a query*

Description

Suggest spelling corrections for a query

Usage

```
search_suggest(query, con = .con())

SearchSuggest(query, con = .con())
```

Arguments

`query`	Character. The query as it was typed.
`con`	A `seqout_connection`. Defaults to the shared REST connection; a Parquet connection reads the dump.

Value

A tibble of suggestions, empty when the query needs no correction.

See Also

seqout_search(), and search_facets() for a query that matched too much.

Examples

```
SearchSuggest("livre cancr")

# already spelled correctly
SearchSuggest("liver cancer")
```

---

seqout_close *Close a Seqout connection*

Description

Parquet closes its DuckDB handle; REST has no handle.

Usage

```
seqout_close(con = .con())

SeqoutClose(con = .con())
```

Arguments

con	A seqout_connection returned by seqout_connect().

Value

NULL, invisibly. Called for its side effect.

Examples

```
con <- SeqoutConnect(quiet = TRUE)
SeqoutClose(con)
```

`seqout_connect` *Connect to Seqout*

Description

Two backends are supported:

Usage

```
seqout_connect(
  backend = c("api", "parquet"),
  base_url = Sys.getenv("SEQOUT_BASE_URL", "https://seqout.org"),
  data_dir = NULL,
  read_only = FALSE,
  eager = FALSE,
  quiet = FALSE
)

SeqoutConnect(
  backend = c("api", "parquet"),
  base_url = Sys.getenv("SEQOUT_BASE_URL", "https://seqout.org"),
  data_dir = NULL,
  read_only = FALSE,
  eager = FALSE,
  quiet = FALSE
)
```

Arguments

`backend`	`"api"`, the default, or `"parquet"`. See the description.
`base_url`	Seqout server URL. Defaults to `SEQOUT_BASE_URL`, then `"https://seqout.org"`.
`data_dir`	Directory or URL holding the `.parquet` files. `NULL` reads `"<base_url>/data"`. The `"api"` backend ignores it.
`read_only`	Make the DuckDB connection read-only. The default is `FALSE`, so that `cache_table()` can write views to local storage. The `"api"` backend ignores this argument.
`eager`	Register every Parquet view at connection time. The `"api"` backend ignores it.
`quiet`	Do not show the message at the start.

Details

- `"api"`, the default, reads https://seqout.org with HTTP.
- `"parquet"` reads the Parquet dump with DuckDB through `httpfs`. It answers SQL and can read a local dump. Requires the duckdb package.

Select Parquet explicitly; one lookup can become a dump scan.

Value

A seqout_connection object.

Examples

```
## Not run:
# REST API default
SeqoutGet("GSE297547")

# Parquet for SQL and local dumps
con <- SeqoutConnect("parquet")
Query("SELECT * FROM geo_series LIMIT 5", con = con)
SeqoutClose(con)

# local dump
con <- SeqoutConnect("parquet", data_dir = file.path(tempdir(), "seqout-dump"))
Query("SELECT count(*) FROM geo_series", con = con)
SeqoutClose(con)

## End(Not run)
```

---

seqout_correct_names *Repair Excel-damaged gene names in a counts matrix*

Description

Uses genevintage::correct_genenames() to resolve date-like gene names against the chosen species and annotation. Tokens with multiple candidates or no candidate are retained and reported. This cannot reconstruct every Excel error or recover information lost through rounding.

Usage

```
seqout_correct_names(
  x,
  ids = NULL,
  species = NULL,
  release = NULL,
  assembly = NULL,
  source = NULL,
  mapping = NULL,
  quiet = FALSE
)

SeqoutCorrectNames(
  x,
  ids = NULL,
  species = NULL,
  release = NULL,
```

```
  assembly = NULL,
  source = NULL,
  mapping = NULL,
  quiet = FALSE
)
```

Arguments

x	A seqout_matrix from seqout_matrix(), or a numeric matrix (including a sparse Matrix), with genes on rows.
ids	Gene names to repair, in matrix row order. Defaults to row names. Pass xvarsymbol when damaged symbols live in a feature column.
species, release, assembly, source	Annotation choices passed to genevintage. NULL detects them from the identifiers. source can be "ensembl", "gencode", or "gencode_all".
mapping	Optional annotation table with id and name columns. Bypasses detection and downloads. Excel correction additionally requires chr and biotype columns, as in genevintage::fetch_mapping().
quiet	Suppress genevintage's progress messages and, with a supplied mapping, mapping warnings.

Details

Supply mapping, or both species and release; genevintage refuses damaged input otherwise, since symbols alone cannot establish species or annotation vintage. Choosing a release explicitly also distinguishes historical names such as SEPT9 from SEPTIN9. Only call this on gene labels: real dates and numeric identifiers can look like Excel damage. Neither this function nor the matrix reader repairs arbitrary spreadsheet cells.

Value

See seqout_gene_names(). Status is "corrected", "ambiguous", "unresolved", or "unchanged". The audit preserves damaged input tokens.

See Also

seqout_gene_names()

Examples

```
x <- matrix(1:4,
  nrow = 2,
  dimnames = list(c("9-Sep", "TP53"), c("s1", "s2"))
)
mapping <- data.frame(
  id = c("ENSG00000184640", "ENSG00000141510"),
  name = c("SEPTIN9", "TP53"), chr = "17", biotype = "protein_coding"
)
SeqoutCorrectNames(x, mapping = mapping)
```

seqout_counts *List count-matrix units for a GEO accession*

Description

Groups supplementary files into readable units without downloading them.

Usage

```
seqout_counts(
  accession,
  assay = "rna",
  feature_type = NULL,
  cache_dir = NULL,
  con = .con()
)

SeqoutCounts(
  accession,
  assay = "rna",
  feature_type = NULL,
  cache_dir = NULL,
  con = .con()
)

SeqoutListCounts(
  accession,
  assay = "rna",
  feature_type = NULL,
  cache_dir = NULL,
  con = .con()
)

seqout_list_counts(
  accession,
  assay = "rna",
  feature_type = NULL,
  cache_dir = NULL,
  con = .con()
)
```

Arguments

accession	A GSE or GSM accession.
assay	Assay preference: "rna" (default), "adt", "hto", "atac", or NULL.
feature_type	10x feature class to keep, overriding the one implied by assay. NULL keeps every row.

cache_dir	Where downloads land. Defaults to a per-accession directory under tools::R_user_dir(), reused across sessions.
	assay names the modality read from the file names, NA when none is named. A sample carrying several becomes a multimodal object.
con	A seqout_connection. Defaults to the shared REST connection.

Details

A unit is a 10x triplet, CellRanger .h5, .h5ad, .rds, tar, or table.

has_metadata means embedded metadata or a named sidecar was found.

preferred ranks requested assay, filtered output, then format.

10x feature class follows assay; feature_type overrides it.

Value

A tibble, one row per readable unit.

Examples

```
counts <- SeqoutListCounts("GSE297547", cache_dir = tempdir())
counts[, c("unit", "sample", "format", "preferred")]


## Not run:
# downloads the matrix from NCBI GEO
m <- SeqoutMatrix(counts, sample = "GSM8994520")

"GSE297547" |>
  SeqoutListCounts() |>
  SeqoutMatrix(sample = "GSM8994520")

## End(Not run)
```

---

seqout_counts_files *Download a unit's files without parsing them*

Description

Download a unit's files without parsing them

Usage

```
seqout_counts_files(counts, sample = NULL)

SeqoutCountsFiles(counts, sample = NULL)
```

Arguments

counts	A seqout_counts tibble from seqout_counts().
sample	A unit label or sample accession.

Value

The paths of the downloaded files.

Examples

```
## Not run:
# downloads the files of one unit from NCBI GEO
counts <- SeqoutListCounts("GSE135251", cache_dir = tempdir())
SeqoutCountsFiles(counts, sample = "GSM3998167")

## End(Not run)
```

---

seqout_default *Set the process-wide default connection*

Description

Set the process-wide default connection

Usage

```
seqout_default(con)

SeqoutDefault(con)
```

Arguments

con	A seqout_connection, or NULL to go back to the built-in REST default.

Value

The previous default, invisibly.

Examples

```
## Not run:
SeqoutDefault(SeqoutConnect("parquet"))
SeqoutGet("GSE297547") # now reads Parquet
SeqoutDefault(NULL) # back to REST

## End(Not run)
```

seqout_fragments *Fragments files for an ATAC study*

Description

A fragments file holds one bgzipped BED-like row per ATAC fragment, with columns `chrom`, `start`, `end`, `barcode` and `read_support`.

Usage

```
seqout_fragments(counts, sample = NULL, download = FALSE)

SeqoutFragments(counts, sample = NULL, download = FALSE)
```

Arguments

`counts`	A `seqout_counts` tibble from `seqout_counts()`.
`sample`	Sample accessions to keep. `NULL` returns every fragments file in the study.
`download`	Fetch the files and report where they landed. `FALSE`, the default, only lists them.

Details

Files download only when `download = TRUE`.

GEO rarely holds the `.tbi` index Signac wants. Build one after downloading with `Rsamtools::indexTabix(path, format = "bed")`.

That works on a CellRanger fragments file, which is bgzipped and sorted by coordinate. Some deposits are neither: GSE184462 ships plain-gzipped files sorted by barcode, and tabix rejects them with "Chromosome blocks not continuous". Sort and recompress first: `sort -k1,1 -k2,2n`, then `Rsamtools::bgzip()`, then `indexTabix()`.

Value

A tibble of `sample`, `file`, `url`, `indexed` and `path`. `path` is `NA` until the file is downloaded.

See Also

`seqout_counts()` for the matrices, `seqout_counts_files()` for a unit's own files.

Examples

```
counts <- SeqoutListCounts("GSE156478", cache_dir = tempdir())
SeqoutFragments(counts)


## Not run:
# downloads a fragments file from NCBI GEO
```

```
frags <- SeqoutFragments(counts, sample = "GSM5065524", download = TRUE)
Rsamtools::indexTabix(frags$path[1], format = "bed")

## End(Not run)
```

`seqout_gene_names` *Convert matrix gene identifiers to annotation-specific names*

Description

Uses `genevintage::geneid2name()` to detect species and rank compatible annotation releases, or `genevintage::gene_names()` with a supplied mapping. Detection is an estimate, especially for small or filtered gene sets. Pass the known annotation explicitly when available. Unmapped IDs are retained.

Usage

```
seqout_gene_names(
  x,
  ids = NULL,
  species = NULL,
  release = NULL,
  assembly = NULL,
  source = NULL,
  mapping = NULL,
  quiet = FALSE
)

SeqoutGeneNames(
  x,
  ids = NULL,
  species = NULL,
  release = NULL,
  assembly = NULL,
  source = NULL,
  mapping = NULL,
  quiet = FALSE
)
```

Arguments

`x`	A `seqout_matrix` from `seqout_matrix()`, or a numeric matrix (including a sparse `Matrix`), with genes on rows.
`ids`	Gene identifiers in row order. Defaults to the matrix row names; may instead come from a feature annotation column such as `x$var$gene_id`.

species, release, assembly, source
: Annotation choices passed to genevintage. NULL detects them from the identifiers. source can be "ensembl", "gencode", or "gencode_all".

mapping
: Optional annotation table with id and name columns. Bypasses detection and downloads. Excel correction additionally requires chr and biotype columns, as in genevintage::fetch_mapping().

quiet
: Suppress genevintage's progress messages and, with a supplied mapping, mapping warnings.

Value

The same kind of object with renamed rows. Counts, row order and observation metadata are preserved. For a seqout_matrix, var row names are updated together with X; original annotation columns are preserved. An audit is appended to x$gene_annotation (or the gene_annotation attribute of a plain matrix). Each entry records operation, annotation and a features table with the original row, input token, resolved name, unique output row name and status. Colliding row names get .1, .2, etc.; the unsuffixed gene name remains in the audit. Rows are never aggregated.

See Also

seqout_correct_names(), genevintage::detect_release()

Examples

```
x <- matrix(1:4, nrow = 2, dimnames = list(
  c("ENSG00000141510.16", "ENSG00000012048.23"), c("s1", "s2")
))
mapping <- data.frame(
  id = c("ENSG00000141510", "ENSG00000012048"),
  name = c("TP53", "BRCA1")
)
SeqoutGeneNames(x, mapping = mapping)
```

---

seqout_gene_report *The audit a gene-name repair left on a matrix*

Description

Reads the audit seqout_gene_names() or seqout_correct_names() appended, wherever it lives: x$gene_annotation for a seqout_matrix, the gene_annotation attribute for a plain matrix.

Usage

```
seqout_gene_report(x, which = NULL)

SeqoutGeneReport(x, which = NULL)
```

Arguments

x	A result of seqout_gene_names() or seqout_correct_names().
which	Which audit, when x was repaired more than once. NULL, the default, takes the most recent.

Value

The audit's features table: original, input, name, row_name and status, one row per matrix row. The call's operation and annotation are attached as attributes of the same names.

Examples

```
x <- matrix(1:4, nrow = 2, dimnames = list(
  c("ENSG00000141510.16", "ENSG00000012048.23"), c("s1", "s2")
))
mapping <- data.frame(
  id = c("ENSG00000141510", "ENSG00000012048"),
  name = c("TP53", "BRCA1")
)
x |>
  SeqoutGeneNames(mapping = mapping) |>
  SeqoutGeneReport()
```

---

seqout_get *Everything reachable from one accession*

Description

Lazy fields fetch on first access and then cache their result.

Usage

```
seqout_get(accession, con = .con())

SeqoutGet(accession, con = .con())
```

Arguments

accession	Any accession Seqout holds.
con	A seqout_connection. Defaults to the shared REST connection.

Details

The fields are:

`meta` the project record: title, summary, design, dates, files

`samples` one record per sample

`experiments` one record per library preparation

`runs` one record per sequencing run, with its file URLs

`supplementary` one row per supplementary file, series and sample alike; `sample` is `NA` on the ones the series carries itself

`bams` the alignment files a submitter sent, where there are any

`pubs` the publications linked to the dataset

`links` the same data in other archives

`enriched` the harmonised data for the samples: the submitter's free text read into one vocabulary, with ontology IDs

`detail` the record for a sample or run accession

Value

A `seqout_dataset` object with lazy fields.

Examples

```
d <- SeqoutGet("GSE168652")
d$meta$title
nrow(d$samples)
nrow(d$runs)
```

---

seqout_matrix *Read one unit as a counts matrix*

Description

Read one unit as a counts matrix

Usage

```
seqout_matrix(counts, sample = NULL)

SeqoutMatrix(counts, sample = NULL)
```

Arguments

counts	A seqout_counts tibble from seqout_counts().
sample	A unit label or sample accession. Required when the accession holds more than one unit.

Value

A seqout_matrix: a list with X (features by observations), obs, var, kind, fmt and evidence.

Examples

```
## Not run:
# downloads the series matrix from NCBI GEO
counts <- SeqoutListCounts("GSE126848", cache_dir = tempdir())
m <- SeqoutMatrix(counts)
dim(m$X)

## End(Not run)
```

---

seqout_online *Check that the Seqout API is reachable*

Description

Asks the REST root once per session with a short timeout, so scripts, examples and vignettes can skip cleanly while the service is down instead of failing on their first query.

Usage

```
seqout_online(con = .con())

SeqoutOnline(con = .con())
```

Arguments

con	A seqout_connection. Defaults to the shared REST connection.

Value

TRUE when the API answered, FALSE otherwise.

Examples

```
SeqoutOnline()
```

seqout_sce *Convert counts to a SingleCellExperiment*

Description

`X` becomes `counts`; `obs` becomes `colData`; `var` becomes `rowData`. A GSE or GSM accession is resolved through `seqout_counts()`. Requires SingleCellExperiment.

Usage

```
seqout_sce(
  x,
  assay_name = "counts",
  sample = NULL,
  max_cells = NULL,
  multimodal = TRUE,
  sample_metadata = FALSE,
  ...
)

Seqout2SCE(
  x,
  assay_name = "counts",
  sample = NULL,
  max_cells = NULL,
  multimodal = TRUE,
  sample_metadata = FALSE,
  ...
)
```

Arguments

`x`	A `seqout_matrix` from `seqout_matrix()`, one GSE/GSM accession, or a `seqout_counts` tibble from `seqout_counts()`.
`assay_name`	Assay name. Defaults to `"counts"`.
`sample`	Unit labels or sample accessions to read when `x` is a GSE or `seqout_counts`. `NULL` reads every preferred unit.
`max_cells`	Cap on cells kept per unit, sampled at random. `NULL` keeps all cells.
`multimodal`	Carry the other modalities as `altExp` entries. `FALSE` reads the preferred unit of each sample.
`sample_metadata`	Join the study's sample record onto every cell, by the accession the cell came from. A column already in the cell metadata wins. The columns of a series-level matrix are matched to GSMs by the sample title, description or a characteristic; without a one-to-one match the join is skipped with a warning.
`...`	Passed to `SingleCellExperiment::SingleCellExperiment()`.

Details

Use it for bulk RNA-seq too: a `SingleCellExperiment` is a `SummarizedExperiment`, so DESeq2 and other bulk tools accept it as it is. A bulk column is named for its GSM. The submitter's own name for it, such as the header of a series-level matrix, is kept in `colData()$label`.

Value

A `SingleCellExperiment` object.

See Also

`seqout_seurat()`, `seqout_matrix()` and `matrices()`.

Examples

```
## Not run:
sce <- Seqout2SCE("GSM8994520")

sce <- Seqout2SCE("GSE297547", sample = "GSM8994520")
counts <- SeqoutListCounts("GSE297547")
sce <- Seqout2SCE(counts, sample = "GSM8994520")

## End(Not run)
```

---

seqout_search *Search every archive*

Description

Searches GEO, SRA, ArrayExpress, ENA, GSA, DRA and GEA together.

Usage

```
seqout_search(
  query = NULL,
  ...,
  sortby = NULL,
  order = "desc",
  limit = NULL,
  structured = FALSE,
  expand = TRUE,
  exclude_ontology = NULL,
  con = .con()
)

SeqoutSearch(
  query = NULL,
  ...,
```

```
  sortby = NULL,
  order = "desc",
  limit = NULL,
  structured = FALSE,
  expand = TRUE,
  exclude_ontology = NULL,
  con = .con()
)
```

Arguments

query	Character. The text to search for; optional if you give a filter.
...	The filters, by name, from the set above.
sortby	"citations", "journal" or "year". The default order is relevance.
order	"desc", the default, or "asc".
limit	Maximum rows. NULL returns every match in 200-row pages.
structured	Read query as a boolean expression with exact terms.
expand	Expand terms with ontology synonyms. FALSE uses exact terms.
exclude_ontology	Character vector. The ontologies to keep out of the expansion, from MONDO, MeSH, HGNC, CHEBI, UBERON, CL, EFO and CVCL. Shared ontology nodes remain while any source is kept.
con	A seqout_connection. Defaults to the shared REST connection; a Parquet connection reads the dump.

Value

A tibble of results, with a took_ms attribute.

See Also

sample_search() when the answer should be samples, and the Search article for the filters and the costs.

Examples

```
SeqoutSearch("liver cancer scRNA", limit = 10)



# filters combine freely
SeqoutSearch("liver cancer", organism = "Homo sapiens", country = "Japan", limit = 10)

# exact words
SeqoutSearch("spinal muscular atrophy", expand = FALSE, limit = 10)
```

```
# force structured parsing
SeqoutSearch("liver cancer", structured = TRUE, limit = 10)

# exclude ontology sources
SeqoutSearch("spinal muscular atrophy", exclude_ontology = c("MeSH", "CVCL"), limit = 10)

# rank by citations
SeqoutSearch("liver cancer scRNA", db = "geo", sortby = "citations", limit = 10)


## Not run:
# slow broad scans on the server
SeqoutSearch('("aging" OR "aged") (gut OR colon) immun*', limit = 10)
SeqoutSearch(organism = "Mus musculus", assay_l1 = "Transcriptomic", limit = 10)

# restrict to long-read (PacBio / Oxford Nanopore) studies
SeqoutSearch("liver fibrosis", long_read = TRUE, limit = 10)

# exact case: LINE retrotransposons, not "cell line"
SeqoutSearch("LINE", case_sensitive = TRUE, limit = 10)

## End(Not run)
```

`seqout_seurat` *Convert counts to a Seurat object*

Description

`X` becomes counts and `obs` becomes `meta.data`. A GSE or GSM accession is resolved through `seqout_counts()`. Requires SeuratObject.

Usage

```
seqout_seurat(
  x,
  sample = NULL,
  max_cells = NULL,
  multimodal = TRUE,
  sample_metadata = FALSE,
  ...
)

Seqout2Seurat(
  x,
  sample = NULL,
  max_cells = NULL,
  multimodal = TRUE,
```

```
  sample_metadata = FALSE,
  ...
)
```

Arguments

x	A seqout_matrix from seqout_matrix(), one GSE/GSM accession, or a seqout_counts tibble from seqout_counts().
sample	Unit labels or sample accessions to read when x is a GSE or seqout_counts. NULL reads every preferred unit.
max_cells	Cap on cells kept per unit, sampled at random. NULL keeps all cells.
multimodal	Build one object per modality found, as extra assays. FALSE reads the preferred unit of each sample.
sample_metadata	Join the study's sample record onto every cell, by the accession the cell came from. A column already in the cell metadata wins.
...	Passed to SeuratObject::CreateSeuratObject().

Value

A Seurat object.

See Also

seqout_matrix() and matrices().

Examples

```
## Not run:
obj <- Seqout2Seurat("GSM8994520", min.cells = 3)

obj <- Seqout2Seurat("GSE297547", sample = "GSM8994520")
counts <- SeqoutListCounts("GSE297547")
obj <- Seqout2Seurat(counts, sample = "GSM8994520")

## End(Not run)
```

---

seqout_sex *Infer sample sex directly from a Seqout counts matrix*

Description

Species is inferred from Ensembl-like row identifiers with genevintage, then sex-chromosome markers are obtained from the same annotation registry used by SexSeek. The returned table contains the dual-evidence coordinates (y_percent and x_percent) used by seqout_sex_plot().

Usage

```
seqout_sex(
  x,
  sample = NULL,
  species = NULL,
  model = "ratio",
  symbol_index = NULL,
  ...
)

SeqoutSex(
  x,
  sample = NULL,
  species = NULL,
  model = "ratio",
  symbol_index = NULL,
  ...
)
```

Arguments

x	A seqout_matrix, numeric matrix, seqout_counts table, or one GSE/GSM accession.
sample	Unit/sample to read when x is an accession or counts table.
species	Optional species override. Usually unnecessary for Ensembl IDs.
model	Model passed to SexSeek::EstimateSex().
symbol_index	Optional genevintage symbol index for standalone symbol matrices.
...	Other arguments passed to SexSeek::EstimateSex().

Value

A data frame of per-unit calls with inferred species and evidence coordinates.

Examples

```
path <- system.file("extdata", "sex", "GSE135251-liver-excerpt.tsv.gz", package = "seqout")
liver <- as.matrix(read.delim(path, row.names = 1, check.names = FALSE))
calls <- SeqoutSex(liver, group = colnames(liver), species = "human")
calls[, c("unit", "verdict", "confidence")]
```

seqout_sex_plot *Plot dual sex-chromosome evidence*

Description

Plot dual sex-chromosome evidence

Usage

```
seqout_sex_plot(x)

SeqoutSexPlot(x)
```

Arguments

x	A result from seqout_sex().

Value

A ggplot object.

Examples

```
path <- system.file("extdata", "sex", "GSE135251-liver-excerpt.tsv.gz", package = "seqout")
liver <- as.matrix(read.delim(path, row.names = 1, check.names = FALSE))
SeqoutSexPlot(SeqoutSex(liver, group = colnames(liver), species = "human"))
```

seqout_units *The readable units behind a counts table*

Description

Nothing is downloaded. A unit is the smallest group of files that reads as one matrix.

Usage

```
seqout_units(counts, preferred_only = TRUE)

SeqoutUnits(counts, preferred_only = TRUE)
```

Arguments

counts	A seqout_counts tibble from seqout_counts().
preferred_only	Return only the preferred unit per sample.

Value

A tibble, one row per unit, with `unit`, `sample`, `platform`, `format`, `assay`, `preferred`, `has_metadata`, `n_files`, `files` and `metadata_files`.

Examples

```
counts <- SeqoutListCounts("GSE126848", cache_dir = tempdir())
SeqoutUnits(counts)
```

---

single_cell_studies *Studies with single-cell evidence*

Description

`data` selects on the `kind` column, which says what a study carries:

Usage

```
single_cell_studies(
  con = .con(),
  min_evidence = 1,
  data = c("any", "matrix", "fastq", "both"),
  limit = NULL,
  offset = 0
)

SingleCellStudies(
  con = .con(),
  min_evidence = 1,
  data = c("any", "matrix", "fastq", "both"),
  limit = NULL,
  offset = 0
)
```

Arguments

`con`	A `seqout_connection`. Defaults to the shared REST connection; a Parquet connection reads the dump.
`min_evidence`	Minimum independent measurements, 1 to 3.
`data`	What the study must carry: `"any"`, `"matrix"`, `"fastq"` or `"both"`. `"matrix"` filters server-side. Other choices read every page before `limit` counts.
`limit`	Maximum studies; `NULL` reads all, in pages of up to 1000.
`offset`	Number of studies to skip.

Details

`matrix_and_reads` counts and reads, both confirmed

`matrix_reads_unscanned` counts plus linked FASTQ no read-derived call has confirmed yet

`matrix_only` counts, no linked reads

`reads_only` no parsed matrix

`"any"` includes reads-only studies and warns when rows lack matrices.

`reads_only` means Pentimento parsed no matrix. GEO supplementary files may contain a matrix readable by `seqout_counts()`.

Value

A study tibble, one row per study.

Examples

```
# unrestricted, with matrix warning
sc <- SingleCellStudies(limit = 100)

# parsed matrices
SingleCellStudies(data = "matrix", limit = 100)

# linked reads
SingleCellStudies(data = "fastq", limit = 100)
```

---

singlecell_facets *Single-cell facet counts*

Description

Study counts per chemistry, organism, tissue, modality, assay category, year, and has_matrix/has_fastq/is_long_read status.

Usage

```
singlecell_facets(con = .con())

SingleCellFacets(con = .con())
```

Arguments

`con`	A `seqout_connection`. Defaults to the shared REST connection; a Parquet connection reads the dump.

Value

A tibble with `facet`, `value` and `studies` columns.

See Also

`singlecell_projects()`.

Examples

```
f <- SingleCellFacets()
f[f$facet == "chemistry", ]
```

---

singlecell_projects *Studies with single-cell evidence*

Description

One row per study with matrix or read-derived single-cell evidence, or a declared single-cell classification: modality, chemistries (read-derived barcode chemistry), `cell_or_nucleus` (read-derived from intron fraction / mito %, independent of chemistry; `NA` when never scanned or ambiguous), tissues, organisms, sample count, matrix availability, FASTQ/.sra availability, and whether the study also has long-read sequencing.

Usage

```
singlecell_projects(
  chemistry = NULL,
  organism = NULL,
  tissue = NULL,
  cell_or_nucleus = NULL,
  perturbation_method = NULL,
  intervention_kind = NULL,
  modality = NULL,
  assay_l1 = NULL,
  year = NULL,
  has_matrix = NULL,
  has_fastq = NULL,
  has_sra = NULL,
  is_long_read = NULL,
  q = NULL,
  sort = "year",
  order = "desc",
  limit = NULL,
  offset = 0,
```

```
  con = .con()
)

SingleCellProjects(
  chemistry = NULL,
  organism = NULL,
  tissue = NULL,
  cell_or_nucleus = NULL,
  perturbation_method = NULL,
  intervention_kind = NULL,
  modality = NULL,
  assay_l1 = NULL,
  year = NULL,
  has_matrix = NULL,
  has_fastq = NULL,
  has_sra = NULL,
  is_long_read = NULL,
  q = NULL,
  sort = "year",
  order = "desc",
  limit = NULL,
  offset = 0,
  con = .con()
)
```

Arguments

chemistry, organism, tissue, cell_or_nucleus, perturbation_method, intervention_kind, modality, assay_l1	Character. Filter to one value from singlecell_facets().
year	Integer. Publication year.
has_matrix, has_fastq, has_sra, is_long_read	Logical filters.
q	Character. Case-insensitive substring match on title or study_accession.
sort	One of "n_samples", "n_cells", "n_runs", "n_fastq_runs", "year", "title", "study_accession", "organism".
order	"desc", the default, or "asc".
limit	Maximum studies; NULL reads all, in pages of up to 200.
offset	Number of studies to skip.
con	A seqout_connection. Defaults to the shared REST connection; a Parquet connection reads the dump.

Details

perturbation_method is a named method (Perturb-seq, CROP-seq, ECCITE-seq, Mosaic-seq, sci-Plex, CRISPR-detect) mentioned in the study's title or description, not a confirmed design; NA means none was detected. intervention_kind (genetic/chemical) and is_pooled follow from which method matched. Filter values come from singlecell_facets().

Value

A study tibble, with a `total` attribute for the filtered count before `limit` cut it.

See Also

`singlecell_summary()`, `singlecell_facets()`.

Examples

```
SingleCellProjects(tissue = "liver", has_matrix = TRUE, limit = 100)
```

---

singlecell_summary *Corpus-wide single-cell totals*

Description

Totals for studies with matrix or read-derived single-cell evidence, or a declared single-cell classification; the population `singlecell_projects()` lists. `cells` excludes studies whose only matrix is unfiltered (raw 10x barcodes, not real cells).

Usage

```
singlecell_summary(con = .con())

SingleCellSummary(con = .con())
```

Arguments

`con`	A `seqout_connection`. Defaults to the shared REST connection; a Parquet connection reads the dump.

Value

A one-row tibble of totals.

See Also

`singlecell_facets()` for the filter values, `singlecell_projects()` for the studies themselves.

Examples

```
SingleCellSummary()
```

spatial_facets *Spatial transcriptomics facet counts*

Description

Study counts per platform, resolution, data availability, organism, tissue, readout assay, cell line, sample type and year. Each facet holds its top 100 values.

Usage

```
spatial_facets(con = .con())

SpatialFacets(con = .con())
```

Arguments

con	A seqout_connection. Defaults to the shared REST connection; a Parquet connection reads the dump.

Value

A tibble with facet, value and studies columns.

See Also

spatial_projects().

Examples

```
f <- SpatialFacets()
f[f$facet == "platform", ]
```

spatial_projects *Spatial transcriptomics studies*

Description

One row per study the modality classifier flags "Spatial Transcriptomics". The platform is named by regex over the study's own text (Visium, Xenium, MERFISH, CosMx, Slide-seq, Slide-seqV2, Curio Seeker, Stereo-seq, GeoMx DSP, DBiT-seq, STARmap, osmFISH, HDST, Tomo-seq, Pixel-seq, Molecular Cartography).

Usage

```
spatial_projects(
  platform = NULL,
  resolution = NULL,
  technology = NULL,
  data_availability = NULL,
  organism = NULL,
  tissue = NULL,
  readout_assay = NULL,
  cell_line = NULL,
  sample_type = NULL,
  year = NULL,
  has_matrix = NULL,
  has_fastq = NULL,
  is_long_read = NULL,
  q = NULL,
  sort = "year",
  order = "desc",
  limit = NULL,
  offset = 0,
  con = .con()
)

SpatialProjects(
  platform = NULL,
  resolution = NULL,
  technology = NULL,
  data_availability = NULL,
  organism = NULL,
  tissue = NULL,
  readout_assay = NULL,
  cell_line = NULL,
  sample_type = NULL,
  year = NULL,
  has_matrix = NULL,
  has_fastq = NULL,
  is_long_read = NULL,
  q = NULL,
  sort = "year",
  order = "desc",
  limit = NULL,
  offset = 0,
  con = .con()
)
```

Arguments

platform	Character. One value from spatial_facets() (Visium, Xenium, MERFISH, ...).
resolution	One of "single-cell", "spot", "roi".
technology	One of "imaging", "sequencing", "hybrid".
data_availability	One of "both", "matrix_only", "fastq_only", "neither".
organism, tissue, readout_assay, cell_line, sample_type	Character. Filter to one value from spatial_facets().
year	Integer. Publication year.
has_matrix, has_fastq, is_long_read	Logical filters.
q	Character. Case-insensitive substring match on title or study_accession.
sort	One of "year", the default, "n_samples", "n_cells", "title", "study_accession", "organism".
order	"desc", the default, or "asc".
limit	Maximum studies; NULL reads all, in pages of up to 200.
offset	Number of studies to skip.
con	A seqout_connection. Defaults to the shared REST connection; a Parquet connection reads the dump.

Details

platforms is empty when the text never names a platform. A named platform is a text mention: a comparison paper can name one it did not run.

resolution groups platforms into "single-cell" (Xenium, MERFISH, CosMx, seqFISH, STARmap, osmFISH, Molecular Cartography), "spot" (coarser than single-cell: Visium, Slide-seq, Curio Seeker, Stereo-seq, DBiT-seq, HDST, Tomo-seq, Pixel-seq) or "roi" (GeoMx DSP, region-of-interest, not single-cell resolution despite reaching this population). technology is "imaging", "sequencing" or "hybrid" (GeoMx DSP: optical ROI selection, sequencing/nCounter readout); it currently tracks resolution one-to-one but is stored separately. A study naming platforms from two groups (e.g. a GeoMx-vs-Xenium comparison) collapses both columns to whichever group is checked first (single-cell/imaging, then spot/sequencing, then roi/hybrid); platforms still lists every name found. data_availability is "both", "matrix_only", "fastq_only" or "neither", from has_matrix and has_fastq. Filter values come from spatial_facets().

Value

A study tibble, with a total attribute for the filtered count before limit cut it.

See Also

spatial_summary(), spatial_facets(), seqout_counts() to read the matrices of a study with has_matrix.

Examples

```
# Xenium studies in human
SpatialProjects(platform = "Xenium", organism = "Homo sapiens", limit = 10)

# single-cell-resolution spatial studies with both a matrix and FASTQ
SpatialProjects(
  resolution = "single-cell", data_availability = "both", limit = 10
)
```

---

spatial_summary *Spatial transcriptomics corpus totals*

Description

Totals for studies flagged single-cell modality "Spatial Transcriptomics", the same population `spatial_projects()` lists. `studies_named_platform` counts studies whose own text names a specific platform (Visium, Xenium, MERFISH, ...); the rest only say "spatial transcriptomics" generically.

Usage

```
spatial_summary(con = .con())

SpatialSummary(con = .con())
```

Arguments

con	A `seqout_connection`. Defaults to the shared REST connection; a Parquet connection reads the dump.

Value

A one-row tibble of totals.

See Also

`spatial_facets()` for the filter values, `spatial_projects()` for the studies themselves.

Examples

```
SpatialSummary()
```

summaries *Short project records for many accessions*

Description

Batches project lookup.

Usage

```
summaries(accessions, con = .con())

Summaries(accessions, con = .con())
```

Arguments

accessions	A character vector of project accessions.
con	A seqout_connection. Defaults to the shared REST connection.

Value

A tibble, one row per project.

Examples

```
Summaries(c("GSE168652", "GSE100379"))
```

supplementary_files *List a project's supplementary files, with Excel gene-corruption scan results*

Description

Only GEO, ArrayExpress and DDBJ GEA projects expose this endpoint. Each row's gene_corruption is NULL when the file has not been scanned yet for Excel's gene-symbol-to-date autocorrupt bug (e.g. "SEPT1" -> "1-Sep"). NULL means no data, not a clean result.

Usage

```
supplementary_files(accession, con = .con())

SupplementaryFiles(accession, con = .con())
```

Arguments

accession	A GEO (GSE), ArrayExpress (E-*) or DDBJ GEA (E-GEAD-*) accession.
con	A seqout_connection. Defaults to the shared REST connection.

Value

A tibble with one row per supplementary file: accession, source, url, filename, path, download_command, and gene_corruption (a list-column, each element NULL or a list with n_hits, kinds, examples, checked_at).

Examples

```
supplementary_files("GSE139368")
```

---

tables *List available tables and views*

Description

registered is FALSE until a remote view is created in DuckDB.

Usage

```
tables(con = .con())

Tables(con = .con())
```

Arguments

con	A Parquet seqout_connection from seqout_connect().

Value

A tibble with table_name, table_type and registered columns.

Examples

```
## Not run:
# DuckDB installs its httpfs extension under the home directory
con <- SeqoutConnect("parquet", quiet = TRUE)
Tables(con)
SeqoutClose(con)

## End(Not run)
```

tissue_facets *Per-tissue facet counts*

Description

Study counts per organism, assay category, archive, journal, submitter country, year, and FASTQ/single-cell/long-read status for studies matching `term`.

Usage

```
tissue_facets(term, con = .con())

TissueFacets(term, con = .con())
```

Arguments

`term`	Character. A tissue name or UBERON term, e.g. `"liver"`.
`con`	A `seqout_connection`. Defaults to the shared REST connection; a Parquet connection reads the dump.

Value

A tibble with `facet`, `value` and `studies` columns.

See Also

`tissue_projects()`.

Examples

```
f <- TissueFacets("liver")
f[f$facet == "organism", ]
```

tissue_projects *Studies with a sample in a given tissue*

Description

One row per study with a sample annotated with a tissue matching `term`. `has_fastq`/`has_sra` are `NA` when the study is absent from the download-links table (unknown, not unavailable); filtering on them accepts only `TRUE` for that reason. `n_samples_with_tissue` counts every sample the study has any resolved tissue for; `tissue_summary()`'s `matched_samples` counts only the ones matching `term`.

Usage

```
tissue_projects(
  term,
  organism = NULL,
  assay_l1 = NULL,
  source = NULL,
  has_fastq = NULL,
  has_sra = NULL,
  is_single_cell = NULL,
  is_long_read = NULL,
  q = NULL,
  sort = "pub_date",
  order = "desc",
  limit = NULL,
  con = .con()
)

TissueProjects(
  term,
  organism = NULL,
  assay_l1 = NULL,
  source = NULL,
  has_fastq = NULL,
  has_sra = NULL,
  is_single_cell = NULL,
  is_long_read = NULL,
  q = NULL,
  sort = "pub_date",
  order = "desc",
  limit = NULL,
  con = .con()
)
```

Arguments

`term`	Character. A tissue name or UBERON term, e.g. `"liver"`.
`organism`, `assay_l1`, `source`	Character. Filter to one value from `tissue_facets()`.
`has_fastq`, `has_sra`, `is_single_cell`, `is_long_read`	Logical filters.
`q`	Character. Case-insensitive substring match on title or study_accession.
`sort`	One of `"pub_date"` (default), `"n_samples"`, `"n_experiments"`, `"title"`, `"study_accession"`, `"organism"`.
`order`	`"desc"`, the default, or `"asc"`.
`limit`	Maximum studies; `NULL` reads all, in pages of up to 200.
`con`	A `seqout_connection`. Defaults to the shared REST connection; a Parquet connection reads the dump.

Value

A study tibble, with a `total` attribute for the filtered count before `limit` cut it.

Pages by cursor internally, as `seqout_search()`'s `sortby` pagination does. There is no `offset` argument; `limit` caps the rows returned.

See Also

`tissue_summary()`, `tissue_facets()`.

Examples

```
TissueProjects("liver", organism = "Homo sapiens", limit = 100)
```

`tissue_summary` *Corpus totals for studies with a sample in a given tissue*

Description

`term` is resolved as free text against the full UBERON ontology and expanded to its descendants (`"blood"` also reaches `"whole blood"`, `"peripheral blood mononuclear cell"`, ...). `resolution` says whether it matched an UBERON label exactly or only as a substring; `matched_labels` lists what actually matched. `matched_samples` counts only samples resolved to one of the matched ids; `samples` is the matching studies' total sample count across all annotations.

Usage

```
tissue_summary(term, con = .con())

TissueSummary(term, con = .con())
```

Arguments

`term`	Character. A tissue name or UBERON term, e.g. `"liver"`.
`con`	A `seqout_connection`. Defaults to the shared REST connection; a Parquet connection reads the dump.

Details

A term with no match 404s.

Value

A one-row tibble of totals.

See Also

`tissue_facets()` for the filter values, `tissue_projects()` for the studies themselves.

Examples

```
TissueSummary("liver")
```

Index

accession_kind, 4
AccessionKind *(*accession_kind*)*, 4
Author *(*author*)*, 5
author, 5

bind_counts, 5
bind_counts(), *48*
BindCounts *(*bind_counts*)*, 5

cache_table, 7
cache_table(), *8*, *21*, *58*
CacheTable *(*cache_table*)*, 7
Citations *(*citations*)*, 7
citations, 7
clear_cache, 8
ClearCache *(*clear_cache*)*, 8
country_code_to_name, 9
country_facets, 10
country_facets(), *10–12*
country_name_to_code
        *(*country_code_to_name*)*, 9
country_projects, 10
country_projects(), *10*, *12*
country_summary, 12
country_summary(), *12*
CountryCodeToName
        *(*country_code_to_name*)*, 9
CountryFacets *(*country_facets*)*, 10
CountryNameToCode
        *(*country_code_to_name*)*, 9
CountryProjects *(*country_projects*)*, 10
CountrySummary *(*country_summary*)*, 12
counts_design, 13
counts_samples, 14
counts_samples(), *13*
CountsDesign *(*counts_design*)*, 13
CountsSamples *(*counts_samples*)*, 14

disease_aliases, 15
disease_facets, 16
disease_facets(), *18*, *19*
disease_projects, 16
disease_projects(), *15*, *16*, *19*
disease_summary, 19
disease_summary(), *16*, *18*
DiseaseAliases *(*disease_aliases*)*, 15
DiseaseFacets *(*disease_facets*)*, 16
DiseaseProjects *(*disease_projects*)*, 16
DiseaseSummary *(*disease_summary*)*, 19
download_bams, 20
download_dump, 21
download_runs, 22
download_runs(), *20*, *50*
download_supplementary, 23
DownloadBAMs *(*download_bams*)*, 20
DownloadDump *(*download_dump*)*, 21
DownloadRuns *(*download_runs*)*, 22
DownloadSupplementary
        *(*download_supplementary*)*, 23

file_role, 24
FileRole *(*file_role*)*, 24
filter_values, 24

group_key, 26
GroupKey *(*group_key*)*, 26
gsm_series, 27
GSMSeries *(*gsm_series*)*, 27

is_filtered, 27
IsFiltered *(*is_filtered*)*, 27

linked_geo, 28
linked_study, 29
LinkedGEO *(*linked_geo*)*, 28
LinkedStudy *(*linked_study*)*, 29
list_assays *(*filter_values*)*, 24
list_centers *(*filter_values*)*, 24
list_instrument_models *(*filter_values*)*,
        24

list_journals (filter_values), 24
list_library_strategies (filter_values), 24
list_organisms (filter_values), 24
list_platforms (filter_values), 24
ListAssays (filter_values), 24
ListCenters (filter_values), 24
ListInstrumentModels (filter_values), 24
ListJournals (filter_values), 24
ListLibraryStrategies (filter_values), 24
ListOrganisms (filter_values), 24
ListPlatforms (filter_values), 24
longread_facets, 29
longread_facets(), *30–33*
longread_projects, 30
longread_projects(), *30*, *33*
longread_summary, 32
longread_summary(), *32*
LongreadFacets (longread_facets), 29
LongreadProjects (longread_projects), 30
LongreadSummary (longread_summary), 32

map_to_ontology, 33
MapToOntology (map_to_ontology), 33
Matrices (matrices), 35
matrices, 35
matrices(), *6*, *71*, *74*
microbe_detections, 35
MicrobeDetections (microbe_detections), 35

Ontology (ontology), 37
ontology, 37
ontology(), *34*

Paper (paper), 38
paper, 38
paper(), *8*
perturbation_facets, 39
perturbation_facets(), *41–43*
perturbation_projects, 39
perturbation_projects(), *39*, *42*, *43*
perturbation_summary, 42
perturbation_summary(), *42*
PerturbationFacets (perturbation_facets), 39
PerturbationProjects (perturbation_projects), 39
PerturbationSummary (perturbation_summary), 42
project_longread_chemistry, 43
project_longread_chemistry(), *32*, *44*
project_single_cell, 44
project_single_cell(), *43*, *45*, *52*
project_single_cell_status, 45
project_single_cell_status(), *46*
project_single_cell_summary, 46
project_single_cell_summary(), *45*
ProjectLongreadChemistry (project_longread_chemistry), 43
ProjectSingleCell (project_single_cell), 44
ProjectSingleCellStatus (project_single_cell_status), 45
ProjectSingleCellSummary (project_single_cell_summary), 46

Query (query), 47
query, 47
query(), *25*
quick_annotation, 48
QuickAnnotation (quick_annotation), 48

register_tables, 49
RegisterTables (register_tables), 49
resolve_study, 49
ResolveStudy (resolve_study), 49
run_files, 50
RunFiles (run_files), 50

sample_microbes, 51
sample_microbes(), *36*, *44*, *54*
sample_search, 52
sample_search(), *14*, *35*, *36*, *72*
SampleMicrobes (sample_microbes), 51
SampleSearch (sample_search), 52
search_facets, 55
search_facets(), *57*
search_suggest, 56
search_suggest(), *56*
SearchFacets (search_facets), 55
SearchSuggest (search_suggest), 56
Seqout2SCE (seqout_sce), 70
Seqout2Seurat (seqout_seurat), 73

seqout_close, 57
seqout_connect, 58
seqout_connect(), *7, 8*, *21*, *47*, *49*, *57*, *87*
seqout_correct_names, 59
seqout_correct_names(), *66, 67*
seqout_counts, 61
seqout_counts(), *13, 14*, *35*, *42*, *63, 64*, *69, 70*, *73, 74*, *76*, *78*, *84*
seqout_counts_files, 62
seqout_counts_files(), *64*
seqout_default, 63
seqout_fragments, 64
seqout_gene_names, 65
seqout_gene_names(), *60*, *66, 67*
seqout_gene_report, 66
seqout_get, 67
seqout_list_counts *(*seqout_counts*)*, 61
seqout_matrix, 68
seqout_matrix(), *60*, *65*, *70, 71*, *74*
seqout_online, 69
seqout_sce, 70
seqout_search, 71
seqout_search(), *24, 25*, *37*, *54–57*, *90*
seqout_seurat, 73
seqout_seurat(), *71*
seqout_sex, 74
seqout_sex(), *76*
seqout_sex_plot, 76
seqout_sex_plot(), *74*
seqout_units, 76
SeqoutClose *(*seqout_close*)*, 57
SeqoutConnect *(*seqout_connect*)*, 58
SeqoutCorrectNames *(*seqout_correct_names*)*, 59
SeqoutCounts *(*seqout_counts*)*, 61
SeqoutCountsFiles *(*seqout_counts_files*)*, 62
SeqoutDefault *(*seqout_default*)*, 63
SeqoutFragments *(*seqout_fragments*)*, 64
SeqoutGeneNames *(*seqout_gene_names*)*, 65
SeqoutGeneReport *(*seqout_gene_report*)*, 66
SeqoutGet *(*seqout_get*)*, 67
SeqoutListCounts *(*seqout_counts*)*, 61
SeqoutMatrix *(*seqout_matrix*)*, 68
SeqoutOnline *(*seqout_online*)*, 69
SeqoutSearch *(*seqout_search*)*, 71
SeqoutSex *(*seqout_sex*)*, 74
SeqoutSexPlot *(*seqout_sex_plot*)*, 76
SeqoutUnits *(*seqout_units*)*, 76
SeuratObject::CreateSeuratObject(), *74*
single_cell_studies, 77
singlecell_facets, 78
singlecell_facets(), *80, 81*
singlecell_projects, 79
singlecell_projects(), *79*, *81*
singlecell_summary, 81
singlecell_summary(), *81*
SingleCellExperiment::SingleCellExperiment(), *70*
SingleCellFacets *(*singlecell_facets*)*, 78
SingleCellProjects *(*singlecell_projects*)*, 79
SingleCellStudies *(*single_cell_studies*)*, 77
SingleCellSummary *(*singlecell_summary*)*, 81
spatial_facets, 82
spatial_facets(), *84, 85*
spatial_projects, 82
spatial_projects(), *82*, *85*
spatial_summary, 85
spatial_summary(), *84*
SpatialFacets *(*spatial_facets*)*, 82
SpatialProjects *(*spatial_projects*)*, 82
SpatialSummary *(*spatial_summary*)*, 85
Summaries *(*summaries*)*, 86
summaries, 86
supplementary_files, 86
SupplementaryFiles *(*supplementary_files*)*, 86

Tables *(*tables*)*, 87
tables, 87
tables(), *21*
tissue_facets, 88
tissue_facets(), *89–91*
tissue_projects, 88
tissue_projects(), *88*, *91*
tissue_summary, 90
tissue_summary(), *88*, *90*
TissueFacets *(*tissue_facets*)*, 88
TissueProjects *(*tissue_projects*)*, 88
TissueSummary *(*tissue_summary*)*, 90
tools::R_user_dir(), *62*