Package 'nemoR'

July 6, 2026

Title Access Open 'NeMO Archive' Datasets

Version 0.99.3

Author Maciej Pietrzak [aut, cre] (ORCID:
<https://orcid.org/0000-0001-5768-7884>),
AbdElrahman Amer [aut]

Maintainer Maciej Pietrzak <pietrzak.20@osu.edu>

Description Provides helpers for discovering, planning, and downloading open-access datasets from the Neuroscience Multi-Omic Archive ('NeMO'; <https://nemoarchive.org/>). The package builds reproducible file manifests that record search parameters, file metadata, download URLs, checksums, and local file paths. It supports exploratory 'NeMO' metadata queries and provides first-pass bridges from downloaded files into 'SingleCellExperiment' and 'Seurat' workflows.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.3)

Imports httr2, jsonlite, tibble

Suggests BiocFileCache, BiocStyle, DropletUtils, knitr, Matrix, rmarkdown, Seurat, SingleCellExperiment, S4Vectors, testthat (>= 3.0.0), zellkonverter

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

NeedsCompilation no

Repository CRAN

Date/Publication 2026-07-06 12:30:02 UTC

Contents

`nemo_access_types` *List available file access categories*

Description

Returns values from `file.file_access`. Use `access = "open"` in search and fetch workflows when you want files that can be downloaded without controlled-access authorization.

Usage

```
nemo_access_types(...)
```

Arguments

`...`	Filters passed to `nemo_facet_values()`.

Value

A tibble of access categories and counts.

Examples

```
# Queries the external NeMO Data Portal; not run during package checks.
## Not run:
nemo_access_types(file_format = "h5ad")

## End(Not run)
```

`nemo_assays` *List available assays*

Description

Returns values from `sample.assay`, optionally filtered by organism, technique, modality, file format, or other search arguments.

Usage

```
nemo_assays(...)
```

Arguments

`...`	Filters passed to `nemo_facet_values()`.

Value

A tibble of assays and counts.

Examples

```
# Queries the external NeMO Data Portal; not run during package checks.
## Not run:
nemo_assays(taxon = "house mouse")

## End(Not run)
```

nemo_as_sce *Convert downloaded NeMO files to a SingleCellExperiment*

Description

Dispatches from manifest file paths to the supported reader for the first compatible file. H5AD files are preferred over 10x HDF5 files when both are present. NeMO provenance from the manifest is attached to the returned object when `S4Vectors` is available.

Usage

```
nemo_as_sce(manifest, ...)
```

Arguments

`manifest`	Manifest with a populated `local_path` column.
`...`	Additional reader arguments.

Value

A `SingleCellExperiment` object.

Examples

```
# Loading a single-cell dataset requires local downloaded .h5ad or
# 10x .h5 files, so this code is not run in examples.
## Not run:
downloaded <- nemo_read_manifest("nemo_downloads/nemo_manifest.tsv")
sce <- nemo_as_sce(downloaded)

## End(Not run)
```

nemo_as_seurat *Convert a SingleCellExperiment to a Seurat object*

Description

Thin wrapper around `Seurat::as.Seurat()` used by `nemo_load()` when `format = "Seurat"`. Seurat is optional and must be installed separately.

Usage

```
nemo_as_seurat(object)
```

Arguments

`object`	SingleCellExperiment object.

Value

A Seurat object.

Examples

```
# Requires optional Seurat conversion and is not run in examples.
## Not run:
if (requireNamespace("SingleCellExperiment", quietly = TRUE) &&
    requireNamespace("Seurat", quietly = TRUE) &&
    requireNamespace("Matrix", quietly = TRUE)) {
  mat <- Matrix::Matrix(c(1, 0, 2, 0, 3, 0), nrow = 3, sparse = TRUE)
  rownames(mat) <- paste0("gene", seq_len(nrow(mat)))
  colnames(mat) <- paste0("cell", seq_len(ncol(mat)))
  sce <- SingleCellExperiment::SingleCellExperiment(
    list(counts = mat, logcounts = mat)
  )
  seurat_obj <- nemo_as_seurat(sce)
  dim(seurat_obj)
}

## End(Not run)
```

---

nemo_brain_regions *List available brain regions/anatomies*

Description

Returns values from `sample.anatomies`, which can be used as the `anatomy` or `brain_region` filter in search and fetch functions.

Usage

```
nemo_brain_regions(...)
```

Arguments

`...`	Filters passed to `nemo_facet_values()`.

Value

A tibble of anatomical regions and counts.

Examples

```
# Queries the external NeMO Data Portal; not run during package checks.
## Not run:
nemo_brain_regions(taxon = "house mouse")

## End(Not run)
```

nemo_cache_dir *Get the nemoR cache directory*

Description

Returns the default directory used by `nemo_download()` when `destdir` is not supplied. The location can be overridden with the `NEMOR_CACHE_DIR` environment variable.

Usage

```
nemo_cache_dir()
```

Value

Path to the cache directory. The directory is created if needed.

Examples

```
nemo_cache_dir()
```

nemo_collection *Retrieve a NeMO collection*

Description

Retrieves one collection record from the NeMO Assets API by collection identifier. Use `nemo_files()` to list file records attached to a collection.

Usage

```
nemo_collection(id)
```

Arguments

`id`	NeMO collection identifier, such as `"nemo:col-rmf5gdy"`.

Value

Parsed collection metadata from the Assets API.

Examples

```
# Queries the external NeMO Assets API; not run during package checks.
## Not run:
nemo_collection("nemo:col-rmf5gdy")

## End(Not run)
```

nemo_collections *List NeMO collections*

Description

The public NeMO Assets API is currently identifier-based. This helper returns documented example public collections that are useful for API tests and demonstrations. It is not intended to be a complete collection catalog.

Usage

```
nemo_collections(...)
```

Arguments

...	Reserved for future use.

Value

A tibble with `name`, `collection_id`, and `api_url` columns.

Examples

```
nemo_collections()
```

nemo_data_types *List available file data types*

Description

Returns values from `file.data_type`, such as processed counts, parameters, raw data, or analysis outputs, depending on what NeMO records match the filters.

Usage

```
nemo_data_types(...)
```

Arguments

...	Filters passed to `nemo_facet_values()`.

Value

A tibble of file data types and counts.

Examples

```
# Queries the external NeMO Data Portal; not run during package checks.
## Not run:
nemo_data_types(taxon = "house mouse", file_format = "h5ad")

## End(Not run)
```

`nemo_download` *Download files from a NeMO manifest*

Description

Downloads each row of a NeMO manifest to a local folder and returns the same manifest with `local_path`, `download_status`, and `checksum_verified` updated. Existing files are skipped unless `overwrite = TRUE`. NeMO datasets can contain large raw or processed files. Use `nemo_download_plan()` and the `max_size_gb` guard before downloading, and increase `max_size_gb` only when the transfer is intentional.

Usage

```
nemo_download(
  manifest,
  destdir = nemo_cache_dir(),
  overwrite = FALSE,
  max_size_gb = 2,
  verify_checksum = FALSE,
  quiet = FALSE
)
```

Arguments

`manifest`	Manifest tibble or data frame produced by `nemo_manifest()`, `nemo_search_manifest()`, `nemo_manifest_from_urls()`, or `nemo_fetch(dry_run = TRUE)`.
`destdir`	Destination directory for downloaded files.
`overwrite`	Whether to overwrite existing local files with the same basename.
`max_size_gb`	Maximum total manifest size to download, in gigabytes. Set to `Inf` to disable this guard.
`verify_checksum`	Whether to verify MD5 checksums after download when checksums are available in the manifest.
`quiet`	Whether to suppress progress messages from `download.file()`.

Value

Updated manifest tibble. `download_status` is set to values such as `"downloaded"`, `"skipped_existing"`, `"failed"`, or `"failed_missing_url"`. `checksum_verified` is `TRUE`, `FALSE`, or `NA`.

Examples

```
source_file <- tempfile()
writeLines("small local file", source_file)
manifest <- nemo_manifest_from_urls(
  paste0("file://", normalizePath(source_file)),
  collection_id = "local-example"
)
manifest$size <- file.info(source_file)$size
nemo_download_plan(manifest, max_size_gb = 1)

# Downloads write files to disk and may transfer large external data.
## Not run:
destdir <- tempfile()
dir.create(destdir)
downloaded <- nemo_download(manifest, destdir = destdir, quiet = TRUE)
downloaded[, c("file_name", "download_status", "local_path")]

## End(Not run)
```

---

nemo_download_plan *Summarize a planned NeMO download*

Description

Use this before downloading to check how many files will be downloaded and how large they are. The summary includes file formats, data types, known total size, whether the size guard would pass, and how many rows have checksums available.

Usage

```
nemo_download_plan(manifest, max_size_gb = 2)
```

Arguments

manifest	Manifest tibble or data frame.
max_size_gb	Maximum allowed known total size in gigabytes.

Value

A one-row tibble with columns n_files, total_size_gb, largest_file_gb, unknown_size_files, within_size_limit, file_formats, data_types, access, download_statuses, and files_with_checksum.

Examples

```
manifest <- nemo_manifest_from_urls(
  "https://example.com?file=sample.h5ad",
  collection_id = "example"
)
manifest$size <- 1024^2
```

```
nemo_download_plan(manifest, max_size_gb = 1)
nemo_download_plan(manifest, max_size_gb = 0.0001)
```

`nemo_example_collections` *Example public NeMO collection identifiers*

Description

These identifiers are small, stable examples useful for documentation, smoke tests, and learning the identifier-based Assets API. They are not a complete catalog of NeMO collections; use the portal search functions for broad discovery.

Usage

```
nemo_example_collections()
```

Value

A named character vector of public collection identifiers from the NeMO Assets API documentation.

Examples

```
nemo_example_collections()
```

`nemo_facet_values` *List available values for NeMO portal facets*

Description

This is the general discovery helper behind convenience functions such as `nemo_species()` and `nemo_platforms()`. It asks the NeMO Data Portal for aggregation buckets, so it is useful for learning valid filter values before running `nemo_search()` or `nemo_fetch()`.

Usage

```
nemo_facet_values(
  facets,
  target = c("files", "samples"),
  organism = NULL,
  taxon = NULL,
  species = NULL,
  technique = NULL,
  modality = NULL,
  assay = NULL,
```

```
  brain_region = NULL,
  anatomy = NULL,
  specimen_type = NULL,
  study = NULL,
  project = NULL,
  program = NULL,
  lab = NULL,
  file_format = NULL,
  data_type = NULL,
  access = NULL,
  filters = NULL,
  size = 1
)
```

Arguments

facets	Character vector of portal fields, such as "subject.taxon" or "sample.technique".
target	Portal target to aggregate over: "files" or "samples".
organism, taxon, species	Optional organism/taxon filter.
technique	Optional sample technique/platform filter.
modality	Optional sample modality filter.
assay	Optional sample assay filter.
brain_region, anatomy	Optional anatomical region filter.
specimen_type	Optional specimen type filter.
study, project	Optional study/grant filter.
program	Optional program filter.
lab	Optional lab filter.
file_format	Optional file format filter.
data_type	Optional file data type filter.
access	Optional file access filter.
filters	Optional raw portal filter object.
size	Number of records to request with the aggregation query. The default is 1 because the portal can return server errors for some filtered aggregation-only requests with size = 0.

Value

A tibble with facet, value, and count columns. value is a usable filter value and count is the number of matching portal records in the requested target.

Examples

```
# Queries the external NeMO Data Portal; not run during package checks.
## Not run:
nemo_facet_values("subject.taxon")
nemo_facet_values("file.format", taxon = "house mouse", access = "open")

## End(Not run)
```

---

nemo_fetch *Search NeMO and download matching files*

Description

`nemo_fetch()` is the highest-level data fetching helper. It searches the NeMO Data Portal, builds a manifest, checks the planned download size, and downloads files to a selected folder. Use `dry_run = TRUE` to inspect the matching files and attached download plan without downloading anything. Downloading real NeMO files can transfer large external datasets, so examples that perform downloads are wrapped in `\dontrun{}`. Inspect a dry run and adjust `max_size_gb` before starting a download.

Usage

```
nemo_fetch(
  destdir,
  organism = NULL,
  taxon = NULL,
  species = NULL,
  technique = NULL,
  modality = NULL,
  assay = NULL,
  brain_region = NULL,
  anatomy = NULL,
  specimen_type = NULL,
  study = NULL,
  project = NULL,
  program = NULL,
  lab = NULL,
  file_format = NULL,
  data_type = NULL,
  access = "open",
  file_name = NULL,
  file_id = NULL,
  sample_id = NULL,
  filters = NULL,
  max_files = 100,
  max_size_gb = 2,
  overwrite = FALSE,
```

```
  verify_checksum = FALSE,
  quiet = FALSE,
  dry_run = FALSE,
  save_manifest = TRUE,
  manifest_name = "nemo_manifest.tsv",
  ...
)
```

Arguments

destdir	Folder where files should be downloaded and where manifest_name is written when save_manifest = TRUE.
organism, taxon, species	Optional organism/taxon filter. These are aliases for the portal field subject.taxon, for example "house mouse" or "human".
technique	Optional sample technique/platform filter, for example "10x chromium 3' v3 sequencing".
modality	Optional sample modality filter, for example "transcriptome".
assay	Optional sample assay filter.
brain_region, anatomy	Optional anatomical region filter.
specimen_type	Optional specimen type filter.
study, project	Optional study/grant filter.
program	Optional program filter.
lab	Optional lab filter.
file_format	Optional file format filter, such as "h5ad", "h5", "mtx", "json", or "fastq".
data_type	Optional file data type filter, such as "counts", "parameters", or "analysis_metrics".
access	File access filter. Defaults to "open".
file_name	Optional exact file name filter.
file_id	Optional file identifier filter.
sample_id	Optional sample identifier filter.
filters	Optional raw portal filter object created by nemo_portal_filters().
max_files	Maximum number of matching files to fetch.
max_size_gb	Maximum total manifest size to download, in gigabytes.
overwrite	Whether to overwrite existing local files.
verify_checksum	Whether to verify MD5 checksums after download when checksums are available in the manifest.
quiet	Whether to suppress progress messages from download.file().
dry_run	Whether to return the manifest and download plan without downloading files.
save_manifest	Whether to write a manifest TSV into destdir.
manifest_name	File name for the saved manifest.
...	Additional query parameters passed to nemo_search_manifest().

Value

A manifest tibble. For downloads, `local_path`, `download_status`, and `checksum_verified` describe local file state. For dry runs, `download_status` is `"dry_run"` and the planned summary is stored in the `"download_plan"` attribute.

Examples

```
# Searches external NeMO services; not run during package checks.
## Not run:
preview <- nemo_fetch(
  destdir = tempdir(),
  taxon = "house mouse",
  file_format = "h5ad",
  access = "open",
  max_files = 1,
  dry_run = TRUE
)
attr(preview, "download_plan")

# Downloads real external data; inspect the dry-run plan first.
manifest <- nemo_fetch(
  destdir = "nemo_downloads",
  taxon = "house mouse",
  file_format = "h5ad",
  access = "open",
  max_files = 1,
  max_size_gb = 2
)

## End(Not run)
```

`nemo_file` *Retrieve a NeMO file record*

Description

Retrieves one file record from the NeMO Assets API. The response can include file format, size, checksum, access level, collection membership, and available manifest URLs.

Usage

```
nemo_file(id)
```

Arguments

`id`	NeMO file identifier, such as `"nemo:fil-..."`.

Value

Parsed file metadata from the Assets API.

Examples

```
# Queries the external NeMO Assets API; not run during package checks.
## Not run:
nemo_file("nemo:fil-328dskg")

## End(Not run)
```

`nemo_files` *List files attached to a NeMO collection*

Description

Uses the NeMO Assets API to list file records for a known collection ID. For broad discovery by species, assay, file type, or access level, use `nemo_search(target = "files")` or `nemo_search_manifest()`.

Usage

```
nemo_files(collection_id, page = 1, page_size = 100, ...)
```

Arguments

`collection_id`	NeMO collection identifier.
`page`	Results page to retrieve from the Assets API.
`page_size`	Number of file records per page.
`...`	Additional API query parameters passed to the Assets API.

Value

A tibble of file metadata returned by the Assets API.

Examples

```
# Queries the external NeMO Assets API; not run during package checks.
## Not run:
nemo_files("nemo:col-rmf5gdy", page_size = 2)

## End(Not run)
```

nemo_file_formats *List available file formats*

Description

Returns values from `file.format`, such as `"h5ad"`, `"h5"`, `"fastq"`, `"bam"`, `"json"`, `"csv"`, or `"mtx"`, depending on the current filters.

Usage

```
nemo_file_formats(...)
```

Arguments

`...`	Filters passed to `nemo_facet_values()`.

Value

A tibble of file formats and counts.

Examples

```
# Queries the external NeMO Data Portal; not run during package checks.
## Not run:
nemo_file_formats(taxon = "house mouse", access = "open")

## End(Not run)
```

nemo_labs *List available labs*

Description

Returns values from `study.labs`, optionally narrowed by species, technique, modality, file format, or other filters.

Usage

```
nemo_labs(...)
```

Arguments

`...`	Filters passed to `nemo_facet_values()`.

Value

A tibble of labs and counts.

Examples

```
# Queries the external NeMO Data Portal; not run during package checks.
## Not run:
nemo_labs(taxon = "house mouse")

## End(Not run)
```

`nemo_load` *Load downloaded NeMO files into an analysis object*

Description

Uses the local file paths recorded in a NeMO manifest to create a `SingleCellExperiment` or Seurat object. This function assumes the files have already been downloaded and currently supports the first compatible `.h5ad` or 10x `.h5` file in the manifest.

Usage

```
nemo_load(manifest, format = c("SingleCellExperiment", "Seurat"), ...)
```

Arguments

`manifest`	Manifest returned by `nemo_manifest()` and optionally `nemo_download()` or `nemo_fetch()`. The `local_path` column must point to downloaded files.
`format`	Output format: `"SingleCellExperiment"` or `"Seurat"`.
`...`	Additional arguments passed to readers or converters.

Value

A `SingleCellExperiment` object by default, or a Seurat object when `format = "Seurat"`.

Examples

```
# Loading a single-cell dataset requires local downloaded .h5ad or
# 10x .h5 files, so this code is not run in examples.
## Not run:
downloaded <- nemo_read_manifest("nemo_downloads/nemo_manifest.tsv")
sce <- nemo_load(downloaded)

## End(Not run)
```

nemo_manifest *Build a reproducible NeMO file manifest*

Description

Builds a manifest from a known NeMO collection and its Assets API file records. A manifest is the package's reproducibility table: it records file identifiers, names, formats, sizes, checksums, download URLs, local paths, query provenance, and download status.

Usage

```
nemo_manifest(
  collection_id,
  files = NULL,
  pattern = NULL,
  file_format = NULL,
  data_type = NULL,
  max_files = NULL
)
```

Arguments

collection_id	NeMO collection identifier.
files	Optional precomputed file table, usually from nemo_files(). When NULL, nemo_files(collection_id) is called.
pattern	Optional regular expression used to filter file names and download URLs.
file_format	Optional file format filter, such as "h5ad" or "h5".
data_type	Optional file data type filter, such as "counts".
max_files	Optional maximum number of files to keep after filtering.

Value

A manifest tibble suitable for nemo_download() and nemo_write_manifest().

Examples

```
files <- data.frame(
  file_id = "file-1",
  file_name = "sample.h5ad",
  file_format = "h5ad",
  data_type = "counts",
  size = 1024,
  md5 = "abc",
  download_url = "https://example.com?file=sample.h5ad"
)
nemo_manifest("example-collection", files = files)
```

nemo_manifest_from_urls *Build a NeMO manifest from direct URLs*

Description

Use this for public files discovered through the NeMO HTTP browser, such as processed `.h5ad` files, when an Assets API collection is not yet known.

Usage

```
nemo_manifest_from_urls(urls, collection_id = "direct-url", file_id = NULL)
```

Arguments

`urls`	Character vector of HTTP or HTTPS URLs to downloadable files.
`collection_id`	Optional collection, dataset, or analysis label stored in the manifest `collection_id` column.
`file_id`	Optional file identifiers. Defaults to generated URL-derived IDs of the form `"url:1"`, `"url:2"`, and so on.

Value

A manifest tibble with one row per URL.

Examples

```
manifest <- nemo_manifest_from_urls(
  "https://example.com?file=sample.h5ad",
  collection_id = "example"
)
manifest
```

nemo_modalities *List available modalities*

Description

Returns values from `sample.modality`, such as transcriptomic or epigenomic measurement categories, optionally filtered by organism, technique, file type, or other search arguments.

Usage

```
nemo_modalities(...)
```

Arguments

...	Filters passed to nemo_facet_values().

Value

A tibble of modalities and counts.

Examples

```
# Queries the external NeMO Data Portal; not run during package checks.
## Not run:
nemo_modalities(taxon = "house mouse", access = "open")

## End(Not run)
```

---

nemo_platforms *List available sequencing or assay platforms*

Description

In NeMO portal metadata, platform-like values are represented by `sample.technique`, for example `"10x chromium 3' v2 sequencing"`.

Usage

```
nemo_platforms(...)
```

Arguments

...	Filters passed to nemo_facet_values().

Value

A tibble of techniques/platforms and counts.

Examples

```
# Queries the external NeMO Data Portal; not run during package checks.
## Not run:
nemo_platforms(taxon = "house mouse")

## End(Not run)
```

nemo_project *Retrieve a NeMO project*

Description

Retrieves one project record from the NeMO Assets API by project identifier. This is a low-level wrapper for users who already know the NeMO project ID.

Usage

```
nemo_project(id)
```

Arguments

id	NeMO project identifier.

Value

Parsed project metadata from the Assets API.

Examples

```
# Queries the external NeMO Assets API; not run during package checks.
## Not run:
nemo_project("nemo:proj-example")

## End(Not run)
```

nemo_read_10x_h5 *Read a downloaded 10x Genomics HDF5 file*

Description

Reads a 10x-style `.h5` count file and returns the object produced by `DropletUtils::read10xCounts()`. This is intended for processed count matrices rather than raw sequencing reads.

Usage

```
nemo_read_10x_h5(path, ...)
```

Arguments

path	Path to a 10x `.h5` file.
...	Additional arguments passed to `DropletUtils::read10xCounts()`.

Value

A SingleCellExperiment object.

Examples

```
# Requires a local 10x HDF5 single-cell file; not run in examples.
## Not run:
sce <- nemo_read_10x_h5("nemo_downloads/sample_10x.h5")

## End(Not run)
```

---

nemo_read_h5ad *Read a downloaded H5AD file as a SingleCellExperiment*

Description

Reads an .h5ad file downloaded from NeMO or supplied manually and returns the object produced by zellkonverter::readH5AD(). This function does not search or download data; use nemo_fetch() or nemo_download() first.

Usage

```
nemo_read_h5ad(path, ...)
```

Arguments

path	Path to an .h5ad file.
...	Additional arguments passed to zellkonverter::readH5AD().

Value

A SingleCellExperiment object.

Examples

```
# Requires a local H5AD file, typically after running nemo_download().
## Not run:
sce <- nemo_read_h5ad("nemo_downloads/sample.h5ad")

## End(Not run)
```

nemo_read_manifest *Read a NeMO manifest*

Description

Reads a manifest previously written by `nemo_write_manifest()`.

Usage

```
nemo_read_manifest(path)
```

Arguments

`path`	Manifest `.tsv` path.

Value

Manifest tibble.

Examples

```
manifest <- nemo_manifest_from_urls("https://example.com?file=sample.h5ad")
path <- tempfile(fileext = ".tsv")
nemo_write_manifest(manifest, path)
nemo_read_manifest(path)
```

nemo_read_mtx *Read a Matrix Market count matrix directory*

Description

Finds `matrix.mtx` or `matrix.mtx.gz` in a directory and reads it with `Matrix::readMM()`. This first-pass reader currently returns only the sparse matrix; barcode and feature annotation files are not yet attached.

Usage

```
nemo_read_mtx(path)
```

Arguments

`path`	Directory containing `matrix.mtx`, `barcodes.tsv`, and features or genes files.

Value

A sparse matrix read from the Matrix Market file.

Examples

```
if (requireNamespace("Matrix", quietly = TRUE)) {
  dir <- tempfile()
  dir.create(dir)
  mat <- Matrix::Matrix(c(1, 0, 2, 0), nrow = 2, sparse = TRUE)
  Matrix::writeMM(mat, file.path(dir, "matrix.mtx"))
  nemo_read_mtx(dir)
}
```

---

nemo_search *Search NeMO portal files, samples, or projects*

Description

This function uses the NeMO Data Portal API for faceted metadata discovery. Use `target = "files"` to find downloadable files, `target = "samples"` to find biospecimen/library records, and `target = "projects"` to summarize studies/grants. Routine users can use biological argument names such as `taxon`, `technique`, and `brain_region` rather than internal portal field names.

Usage

```
nemo_search(
  organism = NULL,
  taxon = NULL,
  species = NULL,
  technique = NULL,
  modality = NULL,
  assay = NULL,
  brain_region = NULL,
  anatomy = NULL,
  specimen_type = NULL,
  study = NULL,
  project = NULL,
  program = NULL,
  lab = NULL,
  file_format = NULL,
  data_type = NULL,
  access = NULL,
  file_name = NULL,
  file_id = NULL,
  sample_id = NULL,
  target = c("files", "samples", "projects"),
  fields = NULL,
  facets = NULL,
  filters = NULL,
  size = 100,
  from = 0,
```

```
  sort = NULL,
  ...
)
```

Arguments

organism, taxon, species	Optional organism/taxon filter. These are aliases for subject.taxon, for example "house mouse" or "human".
technique	Optional sample technique filter, for example "10x chromium 3' v2 sequencing".
modality	Optional sample modality filter, for example "transcriptome" or "chromatin".
assay	Optional sample assay filter.
brain_region, anatomy	Optional sample anatomical region filter.
specimen_type	Optional sample specimen type filter.
study, project	Optional study/grant name filter.
program	Optional study program filter.
lab	Optional study lab filter.
file_format	Optional file format filter, for example "fastq", "h5ad", "h5", or "mtx".
data_type	Optional file data type filter, for example "raw" or "counts".
access	Optional file access filter, for example "open".
file_name	Optional exact file name filter.
file_id	Optional file identifier filter.
sample_id	Optional sample identifier filter.
target	Search endpoint: "files", "samples", or "projects".
fields	Character vector of fields to return. If NULL, sensible defaults are used for the selected target.
facets	Optional character vector of fields to aggregate.
filters	Optional raw portal filter object. If supplied, it is combined with filters generated from the arguments.
size	Number of records to return. Pagination metadata is stored in the "pagination" attribute.
from	Zero-based result offset.
sort	Sort string, such as "file.file_id:asc".
...	Additional query parameters passed to the portal API.

Value

A tibble of matching portal records. Column names are portal field names such as file.file_id, file.name, or subject.taxon. Pagination metadata is stored in the "pagination" attribute and aggregation metadata in the "aggregations" attribute when requested.

Examples

```
# Queries the external NeMO Data Portal; not run during package checks.
## Not run:
results <- nemo_search(
  taxon = "house mouse",
  file_format = "h5ad",
  access = "open",
  target = "files",
  size = 1
)
results
attr(results, "pagination")

## End(Not run)
```

`nemo_search_fields` *List searchable NeMO portal fields*

Description

Retrieves field metadata from the NeMO portal mapping endpoint. This is most useful when constructing advanced `fields`, `facets`, or raw `filters` arguments.

Usage

```
nemo_search_fields(target = NULL)
```

Arguments

`target`	Optional field group: `"files"`, `"samples"`, `"projects"`, or `NULL` for all known fields.

Value

A tibble of portal fields and metadata returned by the mapping endpoint.

Examples

```
# Queries the external NeMO Data Portal; not run during package checks.
## Not run:
nemo_search_fields("files")

## End(Not run)
```

`nemo_search_manifest` *Search NeMO and build a download-ready manifest*

Description

This function connects discovery to download. It searches the NeMO Data Portal for file records, enriches those file IDs through the NeMO Assets API, and returns a manifest that can be passed to `nemo_download()`. It is usually the best entry point when you want a reproducible list of open NeMO files before downloading them.

Usage

```
nemo_search_manifest(
  organism = NULL,
  taxon = NULL,
  species = NULL,
  technique = NULL,
  modality = NULL,
  assay = NULL,
  brain_region = NULL,
  anatomy = NULL,
  specimen_type = NULL,
  study = NULL,
  project = NULL,
  program = NULL,
  lab = NULL,
  file_format = NULL,
  data_type = NULL,
  access = "open",
  file_name = NULL,
  file_id = NULL,
  sample_id = NULL,
  filters = NULL,
  max_files = 100,
  enrich = TRUE,
  require_download_url = TRUE,
  sort = "file.file_id:asc",
  ...
)
```

Arguments

`organism`, `taxon`, `species`	Optional organism/taxon filter. These are aliases for `subject.taxon`, for example `"house mouse"` or `"human"`.
`technique`	Optional sample technique/platform filter.
`modality`	Optional sample modality filter.

assay	Optional sample assay filter.
brain_region, anatomy	Optional anatomical region filter.
specimen_type	Optional specimen type filter.
study, project	Optional study/grant filter.
program	Optional program filter.
lab	Optional lab filter.
file_format	Optional file format filter, such as "h5ad", "h5", "mtx", "fastq", or "bam".
data_type	Optional file data type filter, such as "counts" or "analysis_metrics".
access	File access filter. Defaults to "open".
file_name	Optional exact file name filter.
file_id	Optional file identifier filter.
sample_id	Optional sample identifier filter.
filters	Optional raw portal filter object created by nemo_portal_filters().
max_files	Maximum number of portal file records to include.
enrich	Whether to query the Assets API for download URLs, checksums, and collection IDs.
require_download_url	Whether to keep only records with HTTP download URLs after enrichment.
sort	Portal sort string.
...	Additional query parameters passed to nemo_search().

Value

A tibble manifest suitable for nemo_download(). The manifest includes search metadata such as taxon, technique, study, query_parameters, retrieved_at, and download fields such as download_url, download_status, and checksum_verified.

Examples

```
# Queries external NeMO services; not run during package checks.
## Not run:
manifest <- nemo_search_manifest(
  taxon = "house mouse",
  file_format = "h5ad",
  access = "open",
  max_files = 1
)
manifest

## End(Not run)
```

nemo_species *List available species or taxa in NeMO*

Description

Returns values from the portal field `subject.taxon`, such as `"house mouse"` or `"human"`, optionally after applying filters passed through `...`.

Usage

```
nemo_species(...)
```

Arguments

`...`	Filters passed to `nemo_facet_values()`.

Value

A tibble of species/taxa and counts.

Examples

```
# Queries the external NeMO Data Portal; not run during package checks.
## Not run:
nemo_species()
nemo_species(file_format = "h5ad", access = "open")

## End(Not run)
```

nemo_studies *List available studies/grants*

Description

Returns values from `study.study_name`, optionally narrowed by species, technique, modality, file format, or other filters.

Usage

```
nemo_studies(...)
```

Arguments

`...`	Filters passed to `nemo_facet_values()`.

Value

A tibble of study names and counts.

Examples

```
# Queries the external NeMO Data Portal; not run during package checks.
## Not run:
nemo_studies(taxon = "house mouse")

## End(Not run)
```

`nemo_taxa` *List available taxa*

Description

Alias for `nemo_species()`.

Usage

```
nemo_taxa(...)
```

Arguments

`...`	Filters passed to `nemo_facet_values()`.

Value

A tibble of taxa and counts.

Examples

```
# Queries the external NeMO Data Portal; not run during package checks.
## Not run:
nemo_taxa()

## End(Not run)
```

nemo_techniques *List available techniques*

Description

Alias for `nemo_platforms()`.

Usage

```
nemo_techniques(...)
```

Arguments

`...`	Filters passed to `nemo_facet_values()`.

Value

A tibble of techniques and counts.

Examples

```
# Queries the external NeMO Data Portal; not run during package checks.
## Not run:
nemo_techniques(taxon = "house mouse")

## End(Not run)
```

nemo_validate_manifest *Validate a NeMO manifest*

Description

Checks that a table has the minimum columns needed by the download and load workflow. This validates structure only; it does not check that URLs are reachable or local files exist.

Usage

```
nemo_validate_manifest(manifest)
```

Arguments

`manifest`	Manifest tibble or data frame.

Value

Invisibly returns `manifest`.

Examples

```
manifest <- nemo_manifest_from_urls("https://example.com?file=sample.h5ad")
nemo_validate_manifest(manifest)
```

`nemo_write_manifest` *Write a NeMO manifest*

Description

Writes a manifest to a tab-separated file for reproducibility. Saved manifests can be restored with `nemo_read_manifest()` and passed back to `nemo_download()` or loading helpers.

Usage

```
nemo_write_manifest(manifest, path)
```

Arguments

`manifest`	Manifest tibble or data frame.
`path`	Output `.tsv` path.

Value

Invisibly returns `path`.

Examples

```
manifest <- nemo_manifest_from_urls("https://example.com?file=sample.h5ad")
path <- tempfile(fileext = ".tsv")
nemo_write_manifest(manifest, path)
```

Index