Package 'laminr'

March 13, 2026

Title Client for 'LaminDB'

Version 1.3.0

Description Interact with 'LaminDB'. 'LaminDB' is an open-source data framework for biology. This package allows you to query and download data from 'LaminDB' instances.

License Apache License (>= 2)

URL https://laminr.lamin.ai, https://github.com/laminlabs/laminr

BugReports https://github.com/laminlabs/laminr/issues

Depends R (>= 4.1.0)

Imports callr, cli, lifecycle, pkgload, purrr, R.utils, R6, reticulate (>= 1.41.0), rlang, sessioninfo, utils, withr

Suggests anndata, arrow, IRkernel, jsonlite, knitr, magick, readr, rmarkdown, rstudioapi, rsvg, testthat (>= 3.0.0), yaml

VignetteBuilder knitr

Config/Needs/website rmarkdown, Seurat

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.3

NeedsCompilation no

Author Robrecht Cannoodt [aut, cre] (ORCID: <https://orcid.org/0000-0003-3641-729X>), Luke Zappia [aut] (ORCID: <https://orcid.org/0000-0001-7744-8565>), Data Intuitive [aut], Lamin Labs [aut, cph]

Maintainer Robrecht Cannoodt <robrecht@lamin.ai>

Repository CRAN

Date/Publication 2026-03-13 15:20:03 UTC

Contents

`get_current_lamin_instance` *Get current LaminDB instance*

Description

Get the currently connected LaminDB instance

Usage

```
get_current_lamin_instance(ignore_none = TRUE, silent = FALSE)
```

Arguments

`ignore_none`	Whether to ignore the `"none/none"` virtual instance as a valid instance and return `NULL`
`silent`	Whether to suppress messages

Value

The slug of the current LaminDB instance, or `NULL` invisibly if no instance is found

---

`get_current_lamin_settings` *Get current LaminDB settings*

Description

Get the current LaminDB settings as an R list

Usage

```
get_current_lamin_settings(minimal = FALSE, silent = FALSE)
```

Arguments

minimal	If TRUE, quickly extract a minimal list of important settings instead of converting the complete settings object
silent	Whether to suppress messages

Value

A list of the current LaminDB settings

---

get_current_lamin_user *Get current LaminDB user*

Description

Get the currently logged in LaminDB user

Usage

```
get_current_lamin_user(silent = FALSE)
```

Arguments

silent	Whether to suppress messages

Value

The handle of the current LaminDB user, or NULL invisibly if no user is found

---

import_module *Import Python modules*

Description

This function can be used to import **LaminDB** Python modules with additional checks and nicer error messages.

Usage

```
import_module(module, ...)
```

Arguments

module	The name of the Python module to import
...	Arguments passed on to require_module options A vector of defined optional dependencies for the module that is being required version A string specifying the version of the module to require source A source for the module requirement, for example git+https://github.com/owner/module.g python_version A string defining the Python version to require. Passed to reticulate::py_require() silent Whether to suppress the message showing what has been required

Details

Python dependencies are set using require_module() before importing the module and used to create an ephemeral environment unless another environment is found (see vignette("versions", package = "reticulate")).

Requirements for the lamindb module can be controlled using environment variables differently, see https://docs.lamin.ai/setup-laminr for details.

Value

An object representing a Python package

See Also

- require_module() and reticulate::py_require() for defining Python dependencies
- vignette("versions", package = "reticulate") for setting the Python environment to use (or online here)

Examples

```
## Not run:
# Import lamindb to start interacting with an instance
ln <- import_module("lamindb")

# Import lamindb with optional dependencies
ln <- import_module("lamindb", options = c("dev"))

# Import other LaminDB modules
bt <- import_module("bionty")
pt <- import_module("pertdb")
cc <- import_module("clinicore")

# Import any Python module
np <- import_module("numpy")

## End(Not run)
```

laminr_status *laminr status*

Description

Overview of the current status of the laminr package and its dependencies. Can be useful for debugging.

Usage

```
laminr_status()
```

Details

Provides information that can be useful for debugging. To run the function when an error occurs, set `options(error = function() { print(laminr::laminr_status() })`. Note that this should be used with some caution as it will print the status whenever any error occurs.

Value

A `laminr_status` object

Examples

```
laminr_status()
```

lamin_cli *Lamin CLI functions*

Description

[Deprecated]

Lamin CLI calls are available from R by importing the **lamin_cli** Python module using `lc <- import_module("lamin_cli")`. The previous CLI functions are now deprecated, see examples for how to switch to the new syntax.

Usage

```
# Import the module instead of using deprecated functions
# lc <- import_module("lamin_cli")

# Deprecated functions

lamin_connect(instance)

lamin_disconnect()
```

```
lamin_login(user = NULL, api_key = NULL)

lamin_logout()

lamin_init(storage, name = NULL, db = NULL, modules = NULL)

lamin_init_temp(
  name = "laminr-temp",
  db = NULL,
  modules = NULL,
  add_timestamp = TRUE,
  envir = parent.frame()
)

lamin_delete(instance, force = FALSE)

lamin_save(filepath, key = NULL, description = NULL, registry = NULL)

lamin_settings()
```

Arguments

instance	Either a slug giving the instance to connect to (<owner>/<name>) or an instance URL (https://lamin.ai/owner/name). For lamin_delete(), the slug for the instance to delete.
user	Handle for the user to login as
api_key	API key for a user
storage	A local directory, AWS S3 bucket or Google Cloud Storage bucket
name	A name for the instance
db	A Postgres database connection URL, use NULL for SQLite
modules	A vector of modules to include (e.g. "bionty")
add_timestamp	Whether to append a timestamp to name to make it unique
envir	An environment passed to withr::defer()
force	Whether to force deletion without asking for confirmation
filepath	Path to the file or folder to save
key	The key for the saved item
description	The description for the saved item
registry	The registry for the saved item

Details

lamin_connect():

Running this will set the LaminDB auto-connect option to True so you auto-connect to instance when importing Python lamindb.

`lamin_login()`:

Depending on the input, one of these commands will be run (in this order):

1. If `user` is set then `lamin login <user>`
2. Else if `api_key` is set then set the `LAMIN_API_KEY` environment variable temporarily with `withr::with_envvar()` and run `lamin login`
3. Else if there is a stored user handle run `lamin login <handle>`
4. Else if the `LAMIN_API_KEY` environment variable is set run `lamin login`

Otherwise, exit with an error

`lamin_init_temp()`:

For `lamin_init_temp()`, a time stamp is appended to `name` (if `add_timestamp = TRUE`) and then a new instance is initialised with `lamin_init()` using a temporary directory. A `lamin_delete()` call is registered as an exit handler with `withr::defer()` to clean up the instance when `envir` finishes.

The `lamin_init_temp()` function is mostly for internal use and in most cases users will want `lamin_init()`.

Examples

```
## Not run:
# Import Lamin modules
lc <- import_module("lamin_cli")
ln <- import_module("lamindb")

# Examples of replacing CLI functions with the lamin_cli module

## End(Not run)
## Not run:
# Connect to a LaminDB instance
lamin_connect(instance)
# ->
lc$connect(instance)

## End(Not run)
## Not run:
# Disconnect from a LaminDB instance
lamin_disconnect()
# ->
lc$disconnect()

## End(Not run)
## Not run:
# Log in as a LaminDB user
lamin_login(...)
# ->
lc$login(...)

## End(Not run)
## Not run:
# Log out of LaminDB
```

```
lamin_logout()
# ->
lc$logout()

## End(Not run)
## Not run:
# Create a new LaminDB instance
lamin_init(...)
# ->
lc$init(...)

## End(Not run)
## Not run:
# Create a temporary LaminDB instance
lamin_init_temp(...)
# ->
create_temporary_instance()

## End(Not run)
## Not run:
# Delete a LaminDB entity
lamin_delete(...)
# ->
lc$delete(...)

## End(Not run)
## Not run:
# Save to a LaminDB instance
lamin_save(...)
# ->
lc$save(...)

## End(Not run)
## Not run:
# Access Lamin settings
lamin_settings()
# ->
ln$setup$settings
# OR
ln$settings
# Alternatively
get_current_lamin_settings()

## End(Not run)
```

`require_module` *Require a Python module*

Description

This function can be used to require that Python modules are available for **laminr** with additional checks and nicer error messages.

Usage

```
require_module(
  module,
  options = NULL,
  version = NULL,
  source = NULL,
  python_version = NULL,
  silent = FALSE
)
```

Arguments

`module`	The name of the Python module to require
`options`	A vector of defined optional dependencies for the module that is being required
`version`	A string specifying the version of the module to require
`source`	A source for the module requirement, for example `git+https://github.com/owner/module.git`
`python_version`	A string defining the Python version to require. Passed to `reticulate::py_require()`
`silent`	Whether to suppress the message showing what has been required

Details

Python dependencies are set using `reticulate::py_require()`. If a connection to Python is already initialized and the requested module is already in the list of requirements then a further call to `reticulate::py_require()` will not be made to avoid errors/warnings. This means that required versions etc. need to be set before Python is initialized.

Arguments:

- Setting `options = c("opt1", "opt2")` results in `"module[opt1,opt2]"`
- Setting `version = ">=1.0.0"` results in `"module>=1.0.0"`
- Setting `source = "my_source"` results in `"module @ my_source"`
- Setting all of the above results in `"module[opt1,opt2]>=1.0.0 @ my_source"`

Value

The result of reticulate::py_require

See Also

`reticulate::py_require()`

Examples

```
## Not run:
# Require lamindb
require_module("lamindb")

# Require a specific version of lamindb
require_module("lamindb", version = ">=2.0.0")

# Require require lamindb with options
require_module("lamindb", options = c("dev"))

# Require the development version of lamindb from GitHub
require_module("lamindb", source = "git+https://github.com/laminlabs/lamindb.git")

# Require lamindb with a specific Python version
require_module("lamindb", python_version = "3.12")

## End(Not run)
```

`use_temporary_instance` *Use a temporary LaminDB instance*

Description

Create and connect to a temporary LaminDB instance to use for the current session. This function is primarily intended for developers to use during testing and documentation but can also be useful for users to debug issues or create reproducible examples.

Usage

```
use_temporary_instance(
  name = "laminr-temp",
  modules = NULL,
  add_timestamp = TRUE,
  envir = parent.frame()
)
```

Arguments

`name`	A name for the temporary instance
`modules`	A vector of modules to include (e.g. "bionty")
`add_timestamp`	Whether to append a time stamp to `name` to make it unique
`envir`	An environment passed to `withr::defer()`

Details

This function creates and connects to a temporary LaminDB instance. A temporary storage folder is created and used to initialize a new instance. An exit handler is registered with `withr::defer()` that deletes the instance and storage, then reconnects to the previous instance when `envir` finishes.

Switching to a temporary instance is not possible when another instance is already connected.

Index