

Package ‘laminr’

July 22, 2025

Title Client for 'LaminDB'

Version 1.1.1

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 cli, lifecycle, pak, purrr, R.utils, R6, reticulate (>= 1.38.0), rlang, utils, withr

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

VignetteBuilder knitr

Config/Needs/website rmarkdown

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.2

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 2025-05-15 12:10:05 UTC

Contents

get_current_lamin_instance	2
get_current_lamin_user	2
import_module	3
lamin_connect	4
lamin_delete	5
lamin_disconnect	5
lamin_init	6
lamin_login	7
lamin_logout	7
lamin_save	8
require_module	8
Index	10

get_current_lamin_instance
<i>Get current LaminDB instance</i>

Description

Get the currently connected LaminDB instance

Usage

get_current_lamin_instance()

Details

This is done via a system call to lamin settings to avoid importing Python lamindb

Value

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

get_current_lamin_user
<i>Get current LaminDB user</i>

Description

Get the currently logged in LaminDB user

Usage

get_current_lamin_user()

Details

This is done via a system call to `lamin settings` to avoid importing Python `lamindb`

Value

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

import_module	<i>Import Python modules</i>
---------------	------------------------------

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 <code>git+https://github.com/owner/module.g</code>
python_version	A string defining the Python version to require. Passed to reticulate::py_require()

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")`).

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("bionty", "wetlab"))

# Import other LaminDB modules
bt <- import_module("bionty")
wl <- import_module("wetlab")
cc <- import_module("clinicore")

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

## End(Not run)
```

lamin_connect

Connect to a LaminDB instance

Description

Connect to a LaminDB instance by calling `lamin connect` on the command line

Usage

```
lamin_connect(instance)
```

Arguments

instance	Either a slug giving the instance to connect to (<owner>/<name>) or an instance URL (https://lamin.ai/owner/name)
----------	---

Details

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

Examples

```
## Not run:
lamin_connect("laminlabs/cellxgene")

## End(Not run)
```

lamin_delete	<i>LaminDB delete</i>
--------------	-----------------------

Description

Delete a LaminDB entity. Currently only supports instances.

Usage

```
lamin_delete(instance, force = FALSE)
```

Arguments

instance	Identifier for the instance to delete (e.g. "owner/name")
force	Whether to force deletion without asking for confirmation

Examples

```
## Not run:  
lamin_init("to-delete")  
lamin_delete("to-delete")  
  
## End(Not run)
```

lamin_disconnect	<i>Disconnect from a LaminDB instance</i>
------------------	---

Description

Disconnect from the current LaminDB instance by calling `lamin connect` on the command line

Usage

```
lamin_disconnect()
```

Examples

```
## Not run:  
lamin_disconnect()  
  
## End(Not run)
```

lamin_init	<i>Initialise LaminDB</i>
------------	---------------------------

Description

Initialise a new LaminDB instance

Usage

```
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()
)
```

Arguments

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 <code>withr::defer()</code>

Details

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:
lamin_init("mydata", modules = c("bionty", "wetlab"))

## End(Not run)
```

`lamin_login`*Log into LaminDB*

Description

Log in as a LaminDB user

Usage

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

Arguments

<code>user</code>	Handle for the user to login as
<code>api_key</code>	API key for a user

Details

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_logout`*Log out of LaminDB*

Description

Log out of LaminDB

Usage

```
lamin_logout()
```

lamin_save	<i>Save to a LaminDB instance</i>
------------	-----------------------------------

Description

Save a file or folder to a LaminDB instance by calling `lamin save` on the command line

Usage

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

Arguments

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

See `lamin save --help` for details of what database entries are created for different file types

Examples

```
## Not run:  
my_file <- tempfile()  
lamin_save(my_file)  
  
## End(Not run)
```

require_module	<i>Require a Python module</i>
----------------	--------------------------------

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  
)
```


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 <code>git+https://github.com/owner/module.git</code>
python_version	A string defining the Python version to require. Passed to <code>reticulate::py_require()</code>

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 = ">=1.2")

# Require lamindb with options
require_module("lamindb", options = c("bionty", "wetlab"))

# 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)
```

Index

`get_current_lamin_instance`, [2](#)
`get_current_lamin_user`, [2](#)

`import_module`, [3](#)

`lamin_connect`, [4](#)
`lamin_delete`, [5](#)
`lamin_delete()`, [6](#)
`lamin_disconnect`, [5](#)
`lamin_init`, [6](#)
`lamin_init()`, [6](#)
`lamin_init_temp(lamin_init)`, [6](#)
`lamin_init_temp()`, [6](#)
`lamin_login`, [7](#)
`lamin_logout`, [7](#)
`lamin_save`, [8](#)

`require_module`, [3](#), [8](#)
`require_module()`, [3](#)
`reticulate::py_require`, [9](#)
`reticulate::py_require()`, [3](#), [9](#)

`withr::defer()`, [6](#)