

# Package ‘geeLite’

December 4, 2025

**Type** Package

**Title** Building and Managing Local Databases from 'Google Earth Engine'

**Version** 1.0.5

**Description** Simplifies the creation, management, and updating of local databases using data extracted from 'Google Earth Engine' ('GEE'). It integrates with 'GEE' to store, aggregate, and process spatio-temporal data, leveraging 'SQLite' for efficient, serverless storage. The 'geeLite' package provides utilities for data transformation and supports real-time monitoring and analysis of geospatial features, making it suitable for researchers and practitioners in geospatial science. For details, see Kurbucz and Andrée (2025) ``Building and Managing Local Databases from Google Earth Engine with the geeLite R Package'' <<https://hdl.handle.net/10986/43165>>.

**License** MPL-2.0 | file LICENSE

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.3.2

**VignetteBuilder** knitr

**Imports** rnaturalearthdata,  
rnaturalearth,  
googledrive,  
data.table,  
reticulate,  
rstudioapi,  
geojsonio,  
lubridate,  
jsonlite,  
magrittr,  
progress,  
reshape2,  
RSQLite,  
stringr,  
crayon,  
gargle,  
dplyr,  
h3jsr,  
knitr,  
utils,  
purrr,

stats,  
tidyr,  
rgee,  
cli,  
sf

**Suggests** testthat (>= 3.0.0),  
rmarkdown,  
leaflet,  
withr

**Config/testthat.edition** 3

## Contents

fetch_regions . . . . .	2
fetch_vars . . . . .	3
gee_install . . . . .	4
get_config . . . . .	5
get_state . . . . .	5
init_postp . . . . .	6
modify_config . . . . .	7
read_db . . . . .	7
run_geelite . . . . .	9
set_cli . . . . .	10
set_config . . . . .	10

<b>Index</b>	12
--------------	----

---



---

fetch_regions	<i>Fetch ISO 3166 Country and Subdivision Codes</i>
---------------	---

---

### Description

Returns a data frame containing ISO 3166-1 country codes and ISO 3166-2 subdivision codes for the specified administrative level.

### Usage

```
fetch_regions(admin_lvl = 0)
```

### Arguments

admin_lvl	[optional] (integer) Administrative level to retrieve: 0 for country-level (ISO 3166-1), 1 for first-level subdivisions (ISO 3166-2), or NULL to include both (default: 0).
-----------	---

### Value

A data frame containing region names, ISO 3166-2 codes, and the corresponding administrative levels.

## Examples

```
# Example: Fetch ISO 3166-1 country codes
## Not run:
fetch_regions()

## End(Not run)
```

---

fetch\_vars

*Fetch Variable Information from an SQLite Database*

---

## Description

Displays information on the available variables in the SQLite database (`data/geelite.db`).

## Usage

```
fetch_vars(
  path,
  format = c("data.frame", "markdown", "latex", "html", "pipe", "simple", "rst")
)
```

## Arguments

path	[mandatory] (character) Path to the root directory of the generated database.
format	[mandatory] (character) A character string. Possible values are "data.frame" (default) to return a <code>data.frame</code> object, or one of "markdown", "latex", "html", "pipe" (Pandoc's pipe tables), "simple" (Pandoc's simple tables), and "rst" to be passed on to knitr for formatting.

## Value

Returns the variable information in the selected format. If `format = "data.frame"`, a `data.frame` is returned. For other formats, the output is printed in the specified format and NULL is

## Examples

```
# Example: Printing the available variables
## Not run:
fetch_vars(path = "path/to/db")

## End(Not run)
```

**gee\_install***Install and Configure a Conda Environment for 'rgee'***Description**

Sets up a Conda environment with all required Python and R dependencies for using the `rgee` package, including a specific version of the `earthengine-api`. If Conda is not available, the user will be prompted to install Miniconda. The created environment is automatically registered for use with `rgee`.

**Usage**

```
gee_install(conda = "rgee", python_version = "3.11", force_recreate = FALSE)
```

**Arguments**

- |                             |   |
|-----------------------------|---|
| <code>conda</code>          | [optional] (character) Name of the Conda environment to create or use. Defaults to "rgee".                              |
| <code>python_version</code> | [optional] (character) Python version to use when creating the Conda environment. Defaults to "3.11".                   |
| <code>force_recreate</code> | [optional] (logical) If TRUE, deletes and recreates the Conda environment even if it already exists. Defaults to FALSE. |

**Value**

Invisibly returns the name of the Conda environment used or created.

**Note**

Even after installation, users must manually accept the Conda Terms of Service (ToS) using the ‘`conda tos accept`’ command before package installation can proceed. Clear instructions will be provided if ToS acceptance is needed.

**Examples**

```
# Example: Creating a Conda environment with 'rgee' dependencies
## Not run:
gee_install()

## End(Not run)
```

---

get_config	<i>Print the Configuration File</i>
------------	-------------------------------------

---

## Description

Reads and prints the configuration file from the database's root directory in a human-readable format.

## Usage

```
get_config(path)
```

## Arguments

path [mandatory] (character) The path to the root directory of the generated database.

## Value

A character string representing the formatted JSON content of the configuration file.

## Examples

```
# Example: Printing the configuration file
## Not run:
get_config(path = "path/to/db")

## End(Not run)
```

---

get_state	<i>Print the State File</i>
-----------	-----------------------------

---

## Description

Reads and prints the state file from the database's root directory in a human-readable format.

## Usage

```
get_state(path)
```

## Arguments

path [mandatory] (character) The path to the root directory of the generated database.

## Value

A character string representing the formatted JSON content of the state file.

## Examples

```
# Example: Printing the state file
## Not run:
get_state(path = "path/to/db")

## End(Not run)
```

**init\_postp**

*Initialize Post-Processing Folder and Files*

## Description

Creates a postp folder at the specified path and adds two empty files: `structure.json` and `functions.R`.

## Usage

```
init_postp(path, verbose = TRUE)
```

## Arguments

path	[mandatory] character The path to the root directory where the postp folder should be created.
verbose	[optional] (logical) Display messages (default: TRUE).

## Details

The `structure.json` file is initialized with a default JSON structure: `"default": null`. This file is intended for mapping variables to post-processing functions. The `functions.R` file is created with a placeholder comment indicating where to define the R functions for post-processing. If the postp folder already exists, an error will be thrown to prevent overwriting existing files.

## Value

No return value, called for side effects.

## Examples

```
# Example: Initialize post-processing files in the database directory
## Not run:
init_postp("path/to/db")

## End(Not run)
```

---

<b>modify_config</b>	<i>Modify Configuration File</i>
----------------------	----------------------------------

---

### Description

Modifies the configuration file located in the specified root directory of the generated database (config/config.json) by updating values corresponding to the specified keys.

### Usage

```
modify_config(path, keys, new_values, verbose = TRUE)
```

### Arguments

path	[mandatory] (character) The path to the root directory of the generated database.
keys	[mandatory] (list) A list specifying the path to the values in the configuration file that need updating. Each path should correspond to a specific element in the configuration.
new_values	[mandatory] (list) A list of new values to replace the original values at the locations specified by 'keys'. The length of new_values must match the length of keys.
verbose	[optional] (logical) If TRUE, displays messages about the updates made (default: TRUE).

### Value

No return value, called for side effects.

### Examples

```
# Example: Modifying the configuration file
## Not run:
modify_config(
  path = "path/to/db",
  keys = list("limit", c("source", "MODIS/061/MOD13A2", "NDVI")),
  new_values = list(1000, "mean")
)
## End(Not run)
```

---

<b>read_db</b>	<i>Reading, Aggregating, and Processing the SQLite Database</i>
----------------	---

---

### Description

Reads, aggregates, and processes the SQLite database (data/geelite.db).

## Usage

```
read_db(
  path,
  variables = "all",
  freq = c("month", "day", "week", "bimonth", "quarter", "season", "halfyear", "year"),
  prep_fun = NULL,
  aggr_funs = function(x) mean(x, na.rm = TRUE),
  postp_funs = NULL
)
```

## Arguments

path	[mandatory] (character) Path to the root directory of the generated database.
variables	[optional] (character or integer) Names or IDs of the variables to be read. Use the <code>fetch_vars</code> function to identify available variables and IDs (default: "all").
freq	[optional] (character or NULL) The frequency for data aggregation. Options include "day", "week", "month", "bimonth", "quarter", "season", "halfyear", "year" (default: "month"). If NULL, no aggregation is performed and native dates are returned.
prep_fun	[optional] (function or NULL) A function for pre-processing time series data prior to aggregation. If NULL, a default linear interpolation (via <code>linear_interp</code> ) will be used for daily-frequency data. If non-daily, the default behavior simply returns the vector without interpolation.
aggr_funs	[optional] (function or list) A function or a list of functions for aggregating data to the specified frequency ( <code>freq</code> ). Users can directly refer to variable names or IDs. The default function is the mean: <code>function(x) mean(x, na.rm = TRUE)</code> .
postp_funs	[optional] (function or list) A function or list of functions applied to the time series data of a single bin after aggregation. Users can directly refer to variable names or IDs. The default is NULL, indicating no post-processing.

## Value

A list where the first element (`grid`) is a simple feature (`sf`) object, and subsequent elements are data frame objects corresponding to the variables.

## Examples

---

run\_geelite*Build and Update the Grid Statistics Database*

---

**Description**

Collects and stores grid statistics from Google Earth Engine (GEE) data in SQLite format (data/geelite.db), initializes CLI files (cli/...), and initializes or updates the state (state/state.json) and log (log/log.txt) files.

**Usage**

```
run_geelite(
  path,
  conda = "rgee",
  user = NULL,
  rebuild = FALSE,
  mode = "local",
  verbose = TRUE
)
```

**Arguments**

path	[mandatory] (character) The path to the root directory of the generated database. This must be a writable, non-temporary directory. Avoid using the home directory (~), the current working directory, or the package directory.
conda	[optional] (character) Name of the virtual Conda environment used by the rgee package (default: "rgee").
user	[optional] (character) Specifies the Google account directory within ~/.config/earthengine/. This directory stores credentials for a specific Google account (default: NULL).
rebuild	[optional] (logical) If TRUE, the database and its supplementary files are overwritten based on the configuration file (default: FALSE).
mode	[optional] (character) Mode of data extraction. Currently supports "local" or "drive" (for larger exports via Google Drive). Defaults to "local".
verbose	[optional] (logical) Display computation status and messages (default: TRUE).

**Value**

Invisibly returns NULL, called for side effects.

**Examples**

```
# Example: Build a Grid Statistics Database
## Not run:
run_geelite(path = tempdir())

## End(Not run)
```

`set_cli`*Initialize CLI Files***Description**

Creates R scripts to enable the main functions to be called through the Command Line Interface (CLI). These scripts are stored in the `cli/` directory of the generated database.

**Usage**

```
set_cli(path, verbose = TRUE)
```

**Arguments**

<code>path</code>	[mandatory] (character) The path to the root directory of the generated database. This must be a writable, non-temporary directory. Avoid using the home directory (~), the current working directory, or the package directory.
<code>verbose</code>	[optional] (logical) Whether to display messages (default: TRUE).

**Value**

No return value, called for side effects.

**Examples**

```
## Not run:
set_cli(path = tempdir())

## End(Not run)
```

`set_config`*Initialize the Configuration File***Description**

Creates a configuration file in the specified directory of the generated database (`config/config.json`). If the specified directory does not exist but its parent directory does, it will be created.

**Usage**

```
set_config(
  path,
  regions,
  source,
  start = "2020-01-01",
  resol,
  scale = NULL,
  limit = 10000,
  verbose = TRUE
)
```

## Arguments

path	[mandatory] (character) The path to the root directory of the generated database. This must be a writable, non-temporary directory. Avoid using the home directory (~), the current working directory, or the package directory.
regions	[mandatory] (character) ISO 3166-1 alpha-2 country codes or ISO 3166-2 subdivision codes.
source	[mandatory] (list) Description of Google Earth Engine (GEE) datasets of interest (the complete data catalog of GEE is accessible at: <a href="https://developers.google.com/earth-engine/datasets/catalog">https://developers.google.com/earth-engine/datasets/catalog</a> ). It is a nested list with three levels:  <b>names</b> (list) Datasets of interest (e.g., "MODIS/061/MOD13A1"). <b>bands</b> (list) Bands of interest (e.g., "NDVI"). <b>zonal_stats</b> (character) Statistics of interest (options: "mean", "median", "min", "max", "sd").
start	[optional] (date) First date of the data collection (default: "2020-01-01").
resol	[mandatory] (integer) Resolution of the H3 bin.
scale	[optional] (integer) Specifies the nominal resolution (in meters) for image processing. If left as NULL (the default), a resolution of 1000 is used.
limit	[optional] (integer) In "local" mode, 'limit / dates' sets batch size; in "drive" mode, 'limit' is the max features per export (default: 10000).
verbose	[optional] (logical) Display messages (default: TRUE).

## Value

No return value, called for side effects.

## Examples

```
## Not run:
set_config(path = tempdir(),
           regions = c("SO", "YM"),
           source = list(
             "MODIS/061/MOD13A1" = list(
               "NDVI" = c("mean", "sd")
             )
           ),
           resol = 3)

## End(Not run)
```

# Index

fetch\_regions, 2  
fetch\_vars, 3

gee\_install, 4  
get\_config, 5  
get\_state, 5

init\_postp, 6

linear\_interp, 8

modify\_config, 7

read\_db, 7  
run\_geelite, 9

set\_cli, 10  
set\_config, 10