

# Package ‘shinyscholar’

March 20, 2025

**Version** 0.3.0

**Title** A Template for Creating Reproducible 'shiny' Applications

**Description** Create a skeleton 'shiny' application with `create_template()` that is reproducible, can be saved and meets academic standards for attribution. Forked from 'wallace'. Code is split into modules that are loaded and linked together automatically and each call one function. Guidance pages explain modules to users and flexible logging informs them of any errors. Options enable asynchronous operations, viewing of source code, interactive maps and data tables. Use to create complex analytical applications, following best practices in open science and software development. Includes functions for automating repetitive development tasks and an example application at `run_shinyscholar()` that requires `install.packages(`shinyscholar`, dependencies = TRUE)`. A guide to developing applications can be found on the package website.

**Depends** R ( $\geq 3.5.0$ ), gargoyle, leaflet ( $\geq 2.0.2$ ), shiny ( $\geq 1.8.1$ )

**Imports** curl, devtools, glue, knitr, magrittr, tools, zip

**Suggests** bslib, DT ( $\geq 0.5$ ), dplyr ( $\geq 1.0.2$ ), future, htr2, knitcitations, leaflet.extras ( $\geq 1.0.0$ ), markdown, promises, R6, RColorBrewer, renv, rintrojs, rmarkdown, shinyAce, shinyalert, shinybusy, shinyjs, shinytest2, shinyWidgets ( $\geq 0.6.0$ ), terra, testthat, xml2, withr

**SystemRequirements** pandoc is required for generating reproducible reports

**License** GPL-3

**URL** <https://simon-smart88.github.io/shinyscholar/>

**BugReports** <https://github.com/simon-smart88/shinyscholar/issues>

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**NeedsCompilation** no

**Author** Simon E. H. Smart [aut, cre, cph],  
Tim Lucas [aut] (<<https://orcid.org/0000-0003-4694-8107>>),

Jamie M. Kass [ctb],  
 Gonzalo E. Pinilla-Buitrago [ctb],  
 Andrea Paz [ctb],  
 Bethany A. Johnson [ctb],  
 Valentina Grisales-Betancur [ctb],  
 Dean Attali [ctb],  
 Matthew E. Aiello-Lammens [ctb],  
 Cory Merow [ctb],  
 Mary E. Blair [ctb],  
 Robert P. Anderson [ctb],  
 Sarah I. Meenan [ctb],  
 Olivier Broennimann [ctb],  
 Peter J. Galante [ctb],  
 Brian S. Maitner [ctb],  
 Hannah L. Owens [ctb],  
 Sara Varela [ctb],  
 Bruno Vilela [ctb],  
 Robert Muscarella [ctb]

**Maintainer** Simon E. H. Smart <simon.smart@cantab.net>

**Repository** CRAN

**Date/Publication** 2025-03-20 12:20:02 UTC

## Contents

|                                |    |
|--------------------------------|----|
| shinyscholar-package . . . . . | 2  |
| create_module . . . . .        | 3  |
| create_template . . . . .      | 4  |
| get_nasa_token . . . . .       | 6  |
| metadata . . . . .             | 6  |
| plot_hist . . . . .            | 7  |
| plot_scatter . . . . .         | 8  |
| register_module . . . . .      | 9  |
| run_shinyscholar . . . . .     | 10 |
| save_and_load . . . . .        | 11 |
| select_async . . . . .         | 12 |
| select_query . . . . .         | 13 |
| select_user . . . . .          | 14 |

**Index** **15**

---

shinyscholar-package    shinyscholar: *A modular platform for creating reproducible applications*

---

**Description**

*shinyscholar* is a template for creating reproducible shiny applications to academic standards. *shinyscholar* is forked from 'wallace' and provides a template for producing applications that are interactive, reproducible, adaptable and built to high standards. It was created to ease the process of creating complex workflows in a single, streamlined GUI interface. In addition, executable session code (R Markdown format) can be downloaded to share with others or use as supplementary information for scientific papers and reports. An example application is run via the function [run\\_shinyscholar](#) and requires `install.packages("shinyscholar", dependencies = TRUE)`.

---

|               |                                     |
|---------------|-------------------------------------|
| create_module | <i>Create a shinyscholar module</i> |
|---------------|-------------------------------------|

---

**Description**

Create the template of a new shinyscholar module.

**Usage**

```
create_module(
  id,
  dir,
  map = FALSE,
  result = FALSE,
  rmd = FALSE,
  save = FALSE,
  download = FALSE,
  async = FALSE,
  init = FALSE
)
```

**Arguments**

|                       |  |
|-----------------------|--|
| <code>id</code>       | character. The id of the module.   |
| <code>dir</code>      | character. Path to the parent directory containing the application                                       |
| <code>map</code>      | logical. Whether or not the module should support modifying the map.                                     |
| <code>result</code>   | logical. Whether or not the module should support showing information in the Result tab.                 |
| <code>rmd</code>      | logical. Whether or not the module should add Rmd code to the Session Code download.                     |
| <code>save</code>     | logical. Whether or not the module has some custom data to save when the user saves the current session. |
| <code>download</code> | logical. Whether or not the module should add code to handle downloading a file.                         |
| <code>async</code>    | logical. Whether or not the module will operate asynchronously.  |
| <code>init</code>     | logical. Whether or not the function is being used inside of the init function                           |

**Value**

No return value, called for side effects

**See Also**

[register\\_module](#)

---

|                 |   |
|-----------------|---|
| create_template | <i>Create a skeleton application containing empty modules</i> |
|-----------------|---|

---

**Description**

Creates a skeleton app containing empty modules with options controlling objects in common and whether to include a map, code and tables

**Usage**

```
create_template(
  path,
  name,
  common_objects,
  modules,
  author,
  include_map = TRUE,
  include_table = TRUE,
  include_code = TRUE,
  install = FALSE,
  logger = NULL
)
```

**Arguments**

|                |   |
|----------------|---|
| path           | character. Path to where the app should be created  |
| name           | character. Name of the app which will be used as the package name. Must be only characters and numbers and not start with a number.   |
| common_objects | character vector. Names of objects which will be shared between modules. The objects meta, logger and state are included by default and if include_map is TRUE, the object poly is included to store polygons drawn on the map.   |
| modules        | dataframe. Containing one row for each module in the order to be included and with the following column names: <ul style="list-style-type: none"> <li>• component character. Single word descriptor for the component used to name files</li> <li>• long_component character. Full component name displayed to the user, formatted appropriately</li> <li>• module character. Single word descriptor for the module used to name files</li> </ul> |

- long\_module character. Full module name displayed to the user, formatted appropriately
- map logical. Whether or not the module interacts with the map
- result logical. Whether or not the module produces results
- rmd logical. Whether or not the module is included in the markdown
- save logical. Whether or not the input values of the model should be saved
- download logical. Whether or not the input values of the model should be saved
- async logical. Whether or not the module will run asynchronously

|               |   |
|---------------|---|
| author        | character. Name of the author(s)  |
| include_map   | logical. Whether to include a leaflet map. Default TRUE   |
| include_table | logical. Whether to include a table tab. Default TRUE   |
| include_code  | logical. Whether to include a tab for viewing module code. Default TRUE   |
| install       | logical. Whether to install the package. Default FALSE  |
| logger        | Stores all notification messages to be displayed in the Log Window. Insert the logger reactive list here for running in shiny, otherwise leave the default NULL |

**Value**

No return value, called for side effects

**Author(s)**

Simon E. H. Smart [simon.smart@cantab.net](mailto:simon.smart@cantab.net)

**Examples**

```
td <- tempfile()
dir.create(td, recursive = TRUE)

modules <- data.frame(
  "component" = c("data", "data", "plot", "plot"),
  "long_component" = c("Load data", "Load data", "Plot data", "Plot data"),
  "module" = c("user", "database", "histogram", "scatter"),
  "long_module" = c("Upload your own data", "Query a database to obtain data",
    "Plot the data as a histogram", "Plot the data as a scatterplot"),
  "map" = c(TRUE, TRUE, FALSE, FALSE),
  "result" = c(FALSE, FALSE, TRUE, TRUE),
  "rmd" = c(TRUE, TRUE, TRUE, TRUE),
  "save" = c(TRUE, TRUE, TRUE, TRUE),
  "download" = c(FALSE, FALSE, TRUE, TRUE),
  "async" = c(TRUE, FALSE, FALSE, FALSE))

common_objects = c("raster", "histogram", "scatter")

create_template(path = td, name = "demo",
  common_objects = common_objects, modules = modules,
  author = "Simon E. H. Smart", include_map = TRUE, include_table = TRUE,
```

```
include_code = TRUE, install = FALSE)
```

---

|                |  |
|----------------|--|
| get_nasa_token | <i>Fetch a token from the NASA Earthdata API</i> |
|----------------|--|

---

### Description

Uses the Earthdata API to fetch a token using the user's username and password

### Usage

```
get_nasa_token(username, password)
```

### Arguments

|          |                                    |
|----------|------------------------------------|
| username | character. NASA Earthdata username |
| password | character. NASA Earthdata password |

### Value

A character string containing the token

### Author(s)

Simon Smart [simon.smart@cantab.net](mailto:simon.smart@cantab.net)

---

|          |                                      |
|----------|--------------------------------------|
| metadata | <i>Add metadata lines to modules</i> |
|----------|--------------------------------------|

---

### Description

Adds lines to modules and their associated rmarkdown files to semi-automate reproducibility. By default all the modules in the application are edited or you can specify a single module. If metadata lines are already present, the file will not be edited. This function is currently experimental and only semi-automates the process. To ensure that the code is functional complete the following steps:

- Check that any inputs created by packages other than 'shiny' are included
- Add any inputs created dynamically i.e. those without an explicit line of code to generate them, for example those created inside a loop in a renderUI or from a 'leaflet' or 'DT' object.
- Use the objects in each .Rmd file to call the module's function.

### Usage

```
metadata(folder_path, module = NULL)
```

**Arguments**

folder\_path      character. Path to the parent directory containing the application  
module            character. (optional) Name of a single module to edit

**Value**

No return value, called for side effects

**Author(s)**

Simon E. H. Smart [simon.smart@cantab.net](mailto:simon.smart@cantab.net)

**Examples**

```
td <- tempfile()
dir.create(td, recursive = TRUE)

modules <- data.frame(
  "component" = c("demo"),
  "long_component" = c("demo"),
  "module" = c("demo"),
  "long_module" = c("demo"),
  "map" = c(FALSE),
  "result" = c(TRUE),
  "rmd" = c(TRUE),
  "save" = c(TRUE),
  "download" = c(TRUE),
  "async" = c(FALSE))

create_template(path = td, name = "demo",
               common_objects = c("demo"), modules = modules,
               author = "demo", include_map = FALSE,
               include_table = FALSE, include_code = FALSE, install = FALSE)

test_files <- list.files(
  system.file("extdata", package = "shinyscholar"),
  pattern = "test_test*", full.names = TRUE)

module_directory <- file.path(td, "demo", "inst", "shiny", "modules")
file.copy(test_files, module_directory, overwrite = TRUE)

metadata(file.path(td, "demo"), module = "test_test")
```

**Description**

Called by the plot\_hist module in the example app and extracts values from a raster image, returning a histogram of density

**Usage**

```
plot_hist(raster, bins, logger = NULL)
```

**Arguments**

|        |   |
|--------|---|
| raster | SpatRaster object   |
| bins   | The number of breaks in the histogram   |
| logger | Stores all notification messages to be displayed in the Log Window. Insert the logger reactive list here for running in shiny, otherwise leave the default NULL |

**Value**

a list of class histogram

**Author(s)**

Simon Smart [simon.smart@cantab.net](mailto:simon.smart@cantab.net)

**Examples**

```
if (check_suggests(example = TRUE)) {
  raster <- terra::rast(ncol = 8, nrow = 8)
  raster[] <- sapply(1:terra::ncell(raster), function(x){
    rnorm(1, ifelse(x %% 8 != 0, x %% 8, 8), 3)})
  histogram <- plot_hist(raster, bins = 10)
} else {
  message('reinstall with install.packages("shinyscholar", dependencies = TRUE)
  to run this example')
}
```

---

plot\_scatter

*Extract values from a raster to produce a scatterplot*

---

**Description**

Called by the plot\_scatter module in the example app and samples values from a raster along with either the x or y coordinates of the points sampled

**Usage**

```
plot_scatter(raster, sample, axis, logger = NULL)
```

**Arguments**

|        |   |
|--------|---|
| raster | SpatRaster. Raster to be sampled  |
| sample | numeric. Number of points to sample   |
| axis   | character. Which axis coordinates of the raster to return   |
| logger | Stores all notification messages to be displayed in the Log Window. Insert the logger reactive list here for running in shiny, otherwise leave the default NULL |

**Value**

a dataframe containing the axis values and the cell values

**Author(s)**

Simon Smart [simon.smart@cantab.net](mailto:simon.smart@cantab.net)

**Examples**

```
if (check_suggests(example = TRUE)) {
  raster <- terra::rast(ncol = 8, nrow = 8)
  raster[] <- sapply(1:terra::ncell(raster), function(x){
    rnorm(1, ifelse(x %% 8 != 0, x %% 8, 8), 3)})
  scatterplot <- plot_scatter(raster, sample = 10, axis = "y")
} else {
  message('reinstall with install.packages("shinyscholar", dependencies = TRUE)
  to run this example')
}
```

---

register\_module

*Register a shinyscholar module*

---

**Description**

Currently disabled as cannot be used with apps created by shinyscholar. Before running the shinyscholar application with `run_shinyscholar()`, you can register your own modules to be used in shinyscholar.

**Usage**

```
register_module(config_file)
```

**Arguments**

|             |  |
|-------------|--|
| config_file | The path to a YAML file that contains the information about one or more modules. |
|-------------|--|

**Value**

No return value, called for side effects

**See Also**[create\\_module](#)

---

|                  |                                     |
|------------------|-------------------------------------|
| run_shinyscholar | <i>Run shinyscholar Application</i> |
|------------------|-------------------------------------|

---

**Description**

This function runs the *shinyscholar* application in the user's default web browser.

**Usage**

```
run_shinyscholar(  
  launch.browser = TRUE,  
  port = getOption("shiny.port"),  
  load_file = NULL  
)
```

**Arguments**

`launch.browser` Whether or not to launch a new browser window.  
`port` The port for the shiny server to listen on. Defaults to a random available port.  
`load_file` Path to a saved session file which will be loaded when the app is opened

**Value**

No return value, called for side effects

**Author(s)**

Jamie Kass [jkass@gradcenter.cuny.edu](mailto:jkass@gradcenter.cuny.edu)

Gonzalo E. Pinilla-Buitrago [gpinillabuitrago@gradcenter.cuny.edu](mailto:gpinillabuitrago@gradcenter.cuny.edu)

Simon E. H. Smart [simon.smart@cantab.net](mailto:simon.smart@cantab.net)

**Examples**

```
if(interactive()) {  
  run_shinyscholar()  
}
```

---

|               |   |
|---------------|---|
| save_and_load | <i>Adds lines to modules to save and load input values.</i> |
|---------------|---|

---

### Description

Converts 'shiny' \*Input functions to lines of code required to store and reload the values when the app is saved or loaded. By default all the modules in the application are edited. Currently only input functions from 'shiny' and shinyWidgets::materialSwitch are supported.

### Usage

```
save_and_load(folder_path, module = NULL)
```

### Arguments

|             |  |
|-------------|--|
| folder_path | character. Path to the parent directory containing the application |
| module      | character. (optional) Name of a single module to edit              |

### Value

No return value, called for side effects

### Author(s)

Simon E. H. Smart [simon.smart@cantab.net](mailto:simon.smart@cantab.net)

### Examples

```
td <- tempfile()
dir.create(td, recursive = TRUE)

modules <- data.frame(
  "component" = c("demo"),
  "long_component" = c("demo"),
  "module" = c("demo"),
  "long_module" = c("demo"),
  "map" = c(FALSE),
  "result" = c(TRUE),
  "rmd" = c(TRUE),
  "save" = c(TRUE),
  "download" = c(TRUE),
  "async" = c(FALSE))

create_template(path = td, name = "demo",
  common_objects = c("demo"), modules = modules,
  author = "demo", include_map = FALSE,
  include_table = FALSE, include_code = FALSE, install = FALSE)

test_files <- list.files(
```

```

system.file("extdata", package = "shinyscholar"),
pattern = "test_test*", full.names = TRUE)

module_directory <- file.path(td, "demo", "inst", "shiny", "modules")
file.copy(test_files, module_directory, overwrite = TRUE)

save_and_load(file.path(td, "demo"), module = "test_test")

```

---

select\_async

*Load FAPAR data from NASA asynchronously*


---

### Description

Called by the select\_async module in the example app and loads an FAPAR raster for the selected area via the Earthdata API. This function is identical to select\_query but can be run asynchronously

### Usage

```
select_async(poly, date, token, async = FALSE)
```

### Arguments

|       |   |
|-------|---|
| poly  | matrix. Coordinates of area to load   |
| date  | character. Date of image to load in YYYY-MM-DD format.  |
| token | character. NASA Earthdata API token. <a href="#">Click here</a> to register and then <a href="#">follow these instructions</a> to obtain one. Alternatively supply your username and password to get_nasa_token() |
| async | logical. Whether the function is being run asynchronously   |

### Value

A list containing:

|         |  |
|---------|--|
| raster  | a SpatRaster object when async is FALSE or a PackedSpatRaster when async is TRUE |
| message | Information on the number of missing pixels                                      |

### Author(s)

Simon Smart [simon.smart@cantab.net](mailto:simon.smart@cantab.net)

## Examples

```
## Not run:
if (check_suggests(example = TRUE)) {
  poly <- matrix(c(0.5, 0.5, 1, 1, 0.5, 52, 52.5, 52.5, 52, 52), ncol = 2)
  colnames(poly) <- c("longitude", "latitude")
  date <- "2023-06-20"
  token <- get_nasa_token(username = "<username>", password = "<password>")
  ras <- select_async(poly, date, token)
} else {
  message('reinstall with install.packages("shinyscholar", dependencies = TRUE)
  to run this example')
}

## End(Not run)
```

---

select\_query

*Load FAPAR data from NASA*

---

## Description

Called by the `select_query` module in the example app and loads an FAPAR raster for the selected area via the Earthdata API.

## Usage

```
select_query(poly, date, token, logger = NULL)
```

## Arguments

|        |  |
|--------|--|
| poly   | matrix. Coordinates of area to load  |
| date   | character. Date of image to load in YYYY-MM-DD format.   |
| token  | character. NASA Earthdata API token. <a href="#">Click here</a> to register and then <a href="#">follow these instructions</a> to obtain one. Alternatively supply your username and password to <code>get_nasa_token()</code> |
| logger | Stores all notification messages to be displayed in the Log Window. Insert the logger reactive list here for running in shiny, otherwise leave the default NULL  |

## Value

a SpatRaster object

## Author(s)

Simon Smart [simon.smart@cantab.net](mailto:simon.smart@cantab.net)

**Examples**

```
## Not run:
if (check_suggests(example = TRUE)) {
  poly <- matrix(c(0.5, 0.5, 1, 1, 0.5, 52, 52.5, 52.5, 52, 52), ncol = 2)
  colnames(poly) <- c("longitude", "latitude")
  date <- "2023-06-20"
  token <- get_nasa_token(username = "<username>", password = "<password>")
  ras <- select_query(poly, date, token)
} else {
  message('reinstall with install.packages("shinyscholar", dependencies = TRUE)
  to run this example')
}

## End(Not run)
```

---

select\_user

*Load a raster image*


---

**Description**

Called by the select\_user module in the example app and loads a .tif file as a SpatRaster

**Usage**

```
select_user(raster_path, logger = NULL)
```

**Arguments**

|             |   |
|-------------|---|
| raster_path | character. Path to file to be loaded  |
| logger      | Stores all notification messages to be displayed in the Log Window. Insert the logger reactive list here for running in shiny, otherwise leave the default NULL |

**Value**

a SpatRaster object

**Author(s)**

Simon Smart [simon.smart@cantab.net](mailto:simon.smart@cantab.net)

**Examples**

```
if (check_suggests(example = TRUE)) {
  raster_path <- list.files(system.file("extdata", "wc", package = "shinyscholar"),
  full.names = TRUE)
  raster <- select_user(raster_path)
} else {
  message('reinstall with install.packages("shinyscholar", dependencies = TRUE)
  to run this example')
}
```

# Index

`create_module`, [3](#), [10](#)  
`create_template`, [4](#)  
  
`get_nasa_token`, [6](#)  
  
`metadata`, [6](#)  
  
`plot_hist`, [7](#)  
`plot_scatter`, [8](#)  
  
`register_module`, [4](#), [9](#)  
`run_shiny scholar`, [3](#), [10](#)  
  
`save_and_load`, [11](#)  
`select_async`, [12](#)  
`select_query`, [13](#)  
`select_user`, [14](#)  
`shiny scholar (shiny scholar-package)`, [2](#)  
`shiny scholar-package`, [2](#)