Package 'ladder'

July 22, 2025

Title Get on to the Slides

Version 0.0.3

Description Create tables from within R directly on Google Slides presentations. Currently supports matrix, data.frame and 'flextable' objects.

License MIT + file LICENSE

URL https://www.r-ladder.com

BugReports https://github.com/igrave/ladder/issues

Imports checkmate, cli, curl, flextable, gargle, httpuv, httr, rlang

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

Language en-GB

RoxygenNote 7.3.2

Collate 'api_choose.R' 'api_functions.R' 'api_gluestick.R' 'api_object_request.R' 'api_objects.R' 'api_slides_auth.R' 'api_utils.R' 'create_slides.R' 'generics.R' 'data.frame.R' 'flextable.R' 'flextable_borders.R' 'flextable_cell_properties.R' 'flextable_column_row_sizes.R' 'flextable_merge.R' 'flextable_paragraph_style.R' 'flextable_text_style.R' 'functions.R' 'gauth_token.R' 'get_slide_ids.R' 'helpers.R' 'images.R' 'matrix.R' 'package.R' 'run_script.R' 'thumbnail.R' 'zzz.R'

NeedsCompilation no

Author Isaac Gravestock [aut, cre] (ORCID: <https://orcid.org/0000-0003-0283-2065>)

Maintainer Isaac Gravestock <isaac.gravestock@gmail.com>

Repository CRAN

Date/Publication 2025-05-02 14:30:14 UTC

Contents

`add_to_slides` *Add Object to Slides*

Description

Add Object to Slides

Usage

```
add_to_slides(
  object,
  presentation_id,
  on = NULL,
  object_id,
  overwrite,
  from_top_left = NULL,
  ...
)

## S3 method for class 'data.frame'
add_to_slides(
  object,
  presentation_id,
  on = NULL,
  object_id = new_id("table"),
  overwrite = FALSE,
```

```
  from_top_left = NULL,
  digits = NULL,
  ...
)

## S3 method for class 'flextable'
add_to_slides(
  object,
  presentation_id,
  on = NULL,
  object_id = new_id("table"),
  overwrite = FALSE,
  from_top_left = NULL,
  ...
)

## S3 method for class 'matrix'
add_to_slides(
  object,
  presentation_id,
  on = NULL,
  object_id = new_id("table"),
  overwrite = FALSE,
  from_top_left = NULL,
  digits = NULL,
  ...
)
```

Arguments

`object`	An object to add to slides
`presentation_id`	The id from the Slides presentation
`on`	The id or number of the slide to add `object` to
`object_id`	A unique id for the new object on the slides
`overwrite`	If TRUE and an object with `object_id` exists it will deleted and replaced.
`from_top_left`	Numerical vector of length two giving the position of the table as the distance in from the left and down from the top of the slide in EMU. Use `cm(x)` or `inches(x)` to convert to EMU. If `NULL` a default position is used.
`...`	Other arguments used in methods
`digits`	Number of digits to passed to format for numeric matrices and data frame columns.

Details

A data.frame object is added as a table with the column names in bold as the first row. For other formatting use the `flextable` package and add_to_slides.flextable.

A flextable object is added with all formatting.

A matrix object is added as a table without any row or column names.

Value

A presentation object updated with the new object. This function is used for its side effect of adding an object to the slides. The returned object in R is mostly for inspection.

Examples

```
## Add a data.frame
s <- choose_slides()
obj <- iris[1:5, ]
add_to_slides(obj, s, on = 1)


## Add a flextable
s <- choose_slides()
library(flextable)
ft <- flextable(iris[1:5, ])
ft <- theme_box(ft)
ft <- color(ft, i = 1:3, j = 1:2, "pink", part = "body")
ft <- autofit(ft)
add_to_slides(ft, s, on = 1)


## Add a matrix
s <- choose_slides()
obj <- cov(iris[, 1:4])
add_to_slides(obj, s, on = 1)
```

---

choose_slides *Choose Slides presentation*

Description

Opens a webpage for a user to authenticate with Google and select a presentation. This presentation is then authorised for use with ladder.

Usage

```
choose_slides(presentation = NULL)
```

Arguments

presentation	A string containing the presentation link/URL or the presentation ID.

Value

A presentation id

Examples

```
id <- choose_slides()
```

`create_slides` *Create a new Google Slides presentation*

Description

Create a new Google Slides presentation

Usage

```
create_slides()
```

Value

A presentation id

Examples

```
create_slides()
```

`get_layouts` *Get layouts from a presentation*

Description

Get layouts from a presentation

Usage

```
get_layouts(presentation_id)
```

Arguments

`presentation_id`
: The presentation id

Value

A data frame with columns `layout_objectId`, `name`, `displayName`, and `placeholders_df`

Examples

```
s <- choose_slides()
get_layouts(s)
```

get_object_ids *Get ids of objects on Slides*

Description

Get ids of objects on Slides

Usage

```
get_object_ids(presentation_id)
```

Arguments

presentation_id	character, the presentation id

Value

A list of character vectors of object ids. The list has elements for each page. If a slide page has no objects the list element is NULL otherwise a character vector containing all object ids on that page. Contains ids for all tables, images, lines, shapes, etc.

Examples

```
s <- choose_slides()
get_object_ids(s)
```

get_slide_ids *Get ids of Slides pages*

Description

Get ids of Slides pages

Usage

```
get_slide_ids(presentation_id)
```

Arguments

presentation_id	character, the presentation id

Value

A vector of slide ids.

Examples

```
s <- choose_slides()
get_slide_ids(s)
```

`inches` *Convert lengths to EMU*

Description

Convert lengths to EMU

Usage

```
inches(...)

cm(...)
```

Arguments

`...`	One or more numeric values to convert

Value

A numeric vector of lengths converted to EMU

Examples

```
inches(2, 0)
cm(3, 2)
```

`ladder_auth` *Authorize ladder*

Description

Authorize ladder to view and manage your presentations. This function is a wrapper around `gargle::token_fetch()`.

By default, you are directed to a web browser, asked to sign in to your Google account, and to grant ladder permission to operate on your behalf with Google Slides. By default, with your permission, these user credentials are cached in a folder below your home directory, from where they can be automatically refreshed, as necessary. Storage at the user level means the same token can be used across multiple projects and tokens are less likely to be synced to the cloud by accident.

Usage

```
ladder_auth(
  email = gargle::gargle_oauth_email(),
  path = NULL,
  subject = NULL,
  scopes = c("https://www.googleapis.com/auth/drive.file",
    "https://www.googleapis.com/auth/presentations.currentonly"),
  cache = gargle::gargle_oauth_cache(),
  use_oob = gargle::gargle_oob_default(),
  token = NULL
)
```

Arguments

email	Optional. If specified, email can take several different forms: • "jane@gmail.com", i.e. an actual email address. This allows the user to target a specific Google identity. If specified, this is used for token lookup, i.e. to determine if a suitable token is already available in the cache. If no such token is found, email is used to pre-select the targeted Google identity in the OAuth chooser. (Note, however, that the email associated with a token when it's cached is always determined from the token itself, never from this argument). • "*@example.com", i.e. a domain-only glob pattern. This can be helpful if you need code that "just works" for both alice@example.com and bob@example.com. • TRUE means that you are approving email auto-discovery. If exactly one matching token is found in the cache, it will be used. • FALSE or NA mean that you want to ignore the token cache and force a new OAuth dance in the browser. Defaults to the option named "gargle_oauth_email", retrieved by gargle_oauth_email() (unless a wrapper package implements different default behavior).
path	JSON identifying the service account, in one of the forms supported for the txt argument of jsonlite::fromJSON() (typically, a file path or JSON string).
subject	An optional subject claim. Specify this if you wish to use the service account represented by path to impersonate the subject, who is a normal user. Before this can work, an administrator must grant the service account domain-wide authority. Identify the user to impersonate via their email, e.g. subject = "user@example.com". Note that gargle automatically adds the non-sensitive "https://www.googleapis.com/auth/userinfo.email" scope, so this scope must be enabled for the service account, along with any other scopes being requested.
scopes	A character vector of scopes to request. Pick from those listed at https://developers.google.com/identity/protocols/oauth2/scopes.
cache	Specifies the OAuth token cache. Defaults to the option named "gargle_oauth_cache", retrieved via gargle_oauth_cache().

use_oob — Whether to use out-of-band authentication (or, perhaps, a variant implemented by gargle and known as "pseudo-OOB") when first acquiring the token. Defaults to the value returned by `gargle_oob_default()`. Note that (pseudo-)OOB auth only affects the initial OAuth dance. If we retrieve (and possibly refresh) a cached token, `use_oob` has no effect.

If the OAuth client is provided implicitly by a wrapper package, its type probably defaults to the value returned by `gargle_oauth_client_type()`. You can take control of the client type by setting `options(gargle_oauth_client_type = "web")` or `options(gargle_oauth_client_type = "installed")`.

token — A token with class Token2.0 or an object of httr's class `request`, i.e. a token that has been prepared with `httr::config()` and has a Token2.0 in the `auth_token` component.

Details

Most users, most of the time, do not need to call `ladder_auth()` explicitly – it is triggered by the first action that requires authorization. Even when called, the default arguments often suffice.

However, when necessary, `ladder_auth()` allows the user to explicitly:

- Declare which Google identity to use, via an `email` specification.
- Use a service account token or workload identity federation via `path`.
- Bring your own `token`.
- Customize `scopes`.
- Use a non-default `cache` folder or turn caching off.
- Explicitly request out-of-band (OOB) auth via `use_oob`.

If you are interacting with R within a browser (applies to RStudio Server, Posit Workbench, Posit Cloud, and Google Colaboratory), you need OOB auth or the pseudo-OOB variant. If this does not happen automatically, you can request it explicitly with `use_oob = TRUE` or, more persistently, by setting an option via `options(gargle_oob_default = TRUE)`.

The choice between conventional OOB or pseudo-OOB auth is determined by the type of OAuth client. If the client is of the "installed" type, `use_oob = TRUE` results in conventional OOB auth. If the client is of the "web" type, `use_oob = TRUE` results in pseudo-OOB auth. Packages that provide a built-in OAuth client can usually detect which type of client to use. But if you need to set this explicitly, use the `"gargle_oauth_client_type"` option:

```
options(gargle_oauth_client_type = "web")       # pseudo-OOB
# or, alternatively
options(gargle_oauth_client_type = "installed") # conventional OOB
```

For details on the many ways to find a token, see `gargle::token_fetch()`. For deeper control over auth, use `ladder_auth_configure()` to bring your own OAuth client or API key. To learn more about gargle options, see gargle::gargle_options.

Value

Called for side-effect. Returns invisible `NULL`.

See Also

Other auth functions: ladder_auth_configure(), ladder_deauth()

Examples

```
# load/refresh existing credentials, if available
# otherwise, go to browser for authentication and authorization
ladder_auth()

# see user associated with current token
ladder_user()

# force use of a token associated with a specific email
ladder_auth(email = "jenny@example.com")
ladder_user()

# force the OAuth web dance
ladder_auth(email = NA)

# use a 'read only' scope, so it's impossible to edit or delete files
ladder_auth(scopes = "slides.readonly")

# use a service account token
ladder_auth(path = "foofy-83ee9e7c9c48.json")
```

---

ladder_auth_configure *Edit and view auth configuration*

Description

These functions give more control over and visibility into the auth configuration than ladder_auth() does. ladder_auth_configure() lets the user specify their own:

- OAuth client, which is used when obtaining a user token.
- API key. If ladder is de-authorized via ladder_deauth(), all requests are sent with an API key in lieu of a token.

See the vignette("get-api-credentials", package = "gargle") for more. If the user does not configure these settings, internal defaults are used.

ladder_oauth_client() and ladder_api_key() retrieve the currently configured OAuth client and API key, respectively.

Usage

```
ladder_auth_configure(client, path, api_key, app)

ladder_api_key()

ladder_oauth_client()
```

Arguments

client	A Google OAuth client, presumably constructed via gargle::gargle_oauth_client_from_json(). Note, however, that it is preferred to specify the client with JSON, using the path argument.
path	JSON downloaded from Google Cloud Console, containing a client id and secret, in one of the forms supported for the txt argument of jsonlite::fromJSON() (typically, a file path or JSON string).
api_key	API key.
app	**[Deprecated]** Replaced by the client argument.

Value

- ladder_auth_configure(): An object of R6 class gargle::AuthState, invisibly.
- ladder_oauth_client(): the current user-configured OAuth client.
- ladder_api_key(): the current user-configured API key.

See Also

Other auth functions: ladder_auth(), ladder_deauth()

Examples

```
# see and store the current user-configured OAuth client (probaby `NULL`)
(original_client <- ladder_oauth_client())

# see and store the current user-configured API key (probaby `NULL`)
(original_api_key <- ladder_api_key())

# the preferred way to configure your own client is via a JSON file
# downloaded from Google Developers Console
# this example JSON is indicative, but fake
path_to_json <- system.file(
  "extdata", "client_secret_installed.googleusercontent.com.json",
  package = "gargle"
)
ladder_auth_configure(path = path_to_json)

# this is also obviously a fake API key
ladder_auth_configure(api_key = "the_key_I_got_for_a_google_API")

# confirm the changes
ladder_oauth_client()
ladder_api_key()

# restore original auth config
ladder_auth_configure(client = original_client, api_key = original_api_key)
```

ladder_deauth *Suspend authorization*

Description

Put ladder into a de-authorized state. Instead of sending a token, ladder will send an API key. This can be used to access public resources for which no Google sign-in is required. This is handy for using ladder in a non-interactive setting to make requests that do not require a token. It will prevent the attempt to obtain a token interactively in the browser. The user can configure their own API key via `ladder_auth_configure()` and retrieve that key via `ladder_api_key()`. In the absence of a user-configured key, a built-in default key is used.

Usage

```
ladder_deauth()
```

Value

Called for side-effect. Returns invisible `NULL`.

See Also

Other auth functions: `ladder_auth()`, `ladder_auth_configure()`

Examples

```
ladder_deauth()
ladder_user()
```

ladder_has_token *Is there a token on hand?*

Description

Reports whether ladder has stored a token, ready for use in downstream requests.

Usage

```
ladder_has_token()
```

Value

Logical.

See Also

Other low-level API functions: `ladder_token()`

Examples

```
ladder_has_token()
```

`ladder_token` *Produce configured token*

Description

For internal use or for those programming around the Slides API. Returns a token pre-processed with `httr::config()`. Most users do not need to handle tokens "by hand" or, even if they need some control, `ladder_auth()` is what they need. If there is no current token, `ladder_auth()` is called to either load from cache or initiate OAuth2.0 flow. If auth has been deactivated via `ladder_deauth()`, `ladder_token()` returns `NULL`.

Usage

```
ladder_token()
```

Value

A `request` object (an S3 class provided by httr).

See Also

Other low-level API functions: `ladder_has_token()`

Examples

```
ladder_token()
```

ladder_user *Get info on current user*

Description

Reveals the email address of the user associated with the current token. If no token has been loaded yet, this function does not initiate auth.

Usage

```
ladder_user()
```

Value

An email address or, if no token has been loaded, NULL.

Returns the username associated with the current token or NULL if not authenticated.

See Also

gargle::token_userinfo(), gargle::token_email(), gargle::token_tokeninfo()

Examples

```
## Not run:
ladder_user()

## End(Not run)
```

new_slide *Add a new slide to a presentation*

Description

Add a new slide to a presentation

Usage

```
new_slide(
  presentation_id,
  layout,
  centered_title = NULL,
  subtitle = NULL,
  title = NULL,
  body = NULL
)
```

Arguments

presentation_id	The presentation id
layout	The layout to use for the slide. See get_layouts.
centered_title	Character vector to be inserted into the "centered_title" placeholders in order as for title
subtitle	Character vector to be inserted into the subtitle placeholders in order as for title
title	Character vector to be inserted into the title placeholders in order. Any NA entries will be skip the corresponding placeholder.
body	Character vector to be inserted into the body placeholders in order as for title

Value

The URL of the new slide. This function is mostly used for its side effect of adding a slide to the presentation.

Examples

```
s <- create_slides()
layout <- get_layouts(s)
layout_p9 <- layout$layout_objectId[20]
new_slide(s, layout_p9, title = "Slide Title", subtitle = "A Subtitle", body = "Body Text")
```

---

slides_url *Print presentation URL*

Description

Print presentation URL

Usage

```
slides_url(presentation_id, slide_id = NULL)
```

Arguments

presentation_id	ID of presentation
slide_id	Optional slide id to link directly to a certain slide. See get_slide_ids.

Value

Prints URL as a link and invisibly returns URL.

Examples

```
slides_url("example_id_won't_work_1234567asdfbg")
slides_url("example_id_won't_work_1234567asdfbg", slide_id = "p")
```

`use_gauth_workflow` *Use a Google token from github auth workflow*

Description

Use a Google token from github auth workflow

Usage

```
use_gauth_workflow(access_token)
```

Arguments

`access_token`	The access token from github auth workflow

Value

Sets the internal token to use the provided `access_token` string and returns the `AuthState` token object.

See https://github.com/google-github-actions/auth/ for more details.

Examples

```
google_access_token <- Sys.getenv("access_token")
use_gauth_workflow("your_access_token")
```

`view_slide` *Preview Slide as Image*

Description

Preview Slide as Image

Usage

```
view_slide(presentation_id, page, size = "MEDIUM", viewer = TRUE)
```

Arguments

presentation_id	character, the presentation id
page	character, the page number or id
size	character, the size of the image. One of "SMALL", "MEDIUM", "LARGE"
viewer	logical, if TRUE opens the image in the viewer

Value

A character string of the file path to the saved image and opens the image in the viewer or browser if `viewer = TRUE`.

Examples

```
s <- choose_slides()
tmp_image <- view_slide(s, 1)
file.remove(tmp_image)
```

Index