Package: dlr (via r-universe)

September 12, 2024

Title Download and Cache Files Safely

Version 1.0.1.9001

Description The goal of dlr is to provide a friendly wrapper around the common pattern of downloading a file if that file does not already exist locally.

License Apache License (>= 2)

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.1

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

Imports cli, digest, fs, glue, rappdirs, rlang, utils

URL https://github.com/macmillancontentscience/dlr

BugReports https://github.com/macmillancontentscience/dlr/issues

VignetteBuilder knitr

Repository https://macmillancontentscience.r-universe.dev

RemoteUrl https://github.com/macmillancontentscience/dlr

RemoteRef HEAD

RemoteSha 3c0473e61f2fe8e05b8ad4622fc25bcb843fcb54

Contents

`app_cache_dir` *Path to an App Cache Directory*

Description

App cache directories can depend on the user's operating system and an overall `R_USER_CACHE_DIR` environment variable. We also respect a per-app option (`appname.dir`), and a per-app environment variable (`APPNAME_CACHE_DIR`). This function returns the path that will be used for a given app's cache.

Usage

```
app_cache_dir(appname, verbose = interactive())
```

Arguments

`appname`	Character; the name of the application that will "own" the cache, such as the name of a package.

Value

The full path to the app's cache directory.

Examples

```
app_cache_dir("myApp")
```

`construct_cached_file_path`
Construct Cache Path

Description

Construct the full path to the cached version of a file within a particular app's cache, using the source path of the file to make sure the cache filename is unique.

Usage

```
construct_cached_file_path(source_path, appname, extension = "")
```

Arguments

source_path	Character scalar; the full path to the source file.
appname	Character; the name of the application that will "own" the cache, such as the name of a package.
extension	Character scalar; an optional filename extension.

Value

The full path to the processed version of source_path in the app's cache directory.

Examples

```
construct_cached_file_path(
  source_path = "my/file.txt",
  appname = "dlr",
  extension = "rds"
)
```

---

construct_processed_filename

Construct Processed Filename

Description

Given the path to a file, construct a unique filename using the hash of the path.

Usage

```
construct_processed_filename(source_path, extension = "")
```

Arguments

source_path	Character scalar; the full path to the source file.
extension	Character scalar; an optional filename extension.

Value

A unique filename for a processed version of the file.

Examples

```
construct_processed_filename(
  source_path = "my/file.txt",
  extension = "rds"
)
```

create_app_cache_dir *Create a Cache Directory for an App*

Description

Create the default path expected by `app_cache_dir()`.

Usage

```
create_app_cache_dir(appname)
```

Arguments

appname	Character; the name of the application that will "own" the cache, such as the name of a package.

Value

A normalized path to a cache directory. The directory is created if the user has write access and the directory does not exist.

Examples

```
# Executing this function creates a cache directory.
create_app_cache_dir("dlr")
```

maybe_cache *Cache a File if Necessary*

Description

This function wraps `maybe_process()`, specifying the app's cache directory.

Usage

```
maybe_cache(
  source_path,
  appname,
  filename = construct_processed_filename(source_path),
  process_f = readRDS,
  process_args = NULL,
  write_f = saveRDS,
  write_args = NULL,
  force_process = FALSE
)
```

Arguments

source_path	Character scalar; the path to the raw file. Paths starting with http://, https://, ftp://, or ftps:// will be downloaded to a temp file if the processed version is not already available.
appname	Character; the name of the application that will "own" the cache, such as the name of a package.
filename	Character; an optional filename for the cached version of the file. By default, a filename is constructed using construct_processed_filename().
process_f	A function or one-sided formula to use to process the source file. source_path will be passed as the first argument to this function. Defaults to read_f.
process_args	An optional list of additional arguments to process_f.
write_f	A function or one-sided formula to use to save the processed file. The processed object will be passed as the first argument to this function, and target_path will be passed as the second argument. Defaults to base::saveRDS().
write_args	An optional list of additional arguments to write_f.
force_process	A logical scalar indicating whether we should process the source file even if the target already exists. This can be particularly useful if you wish to redownload a file.

Value

The normalized target_path.

Examples

```
if (interactive()) {
  target_path <- maybe_cache(
    "https://query.data.world/s/owqxojjiphaypjmlxldsp566lck7co",
    appname = "dlr",
    process_f = read.csv
  )
  target_path

  unlink(target_path)
}
```

---

maybe_process *Process a File if Necessary*

Description

Sometimes you just need to get a processed file to a particular location, without reading the data. For example, you might need to download a lookup table used by various functions in a package, independent of a particular function call that needs the data. This function does the processing if it hasn't already been done.

Usage

```
maybe_process(
  source_path,
  target_path,
  process_f = readRDS,
  process_args = NULL,
  write_f = saveRDS,
  write_args = NULL,
  force_process = FALSE
)
```

Arguments

source_path	Character scalar; the path to the raw file. Paths starting with `http://`, `https://`, `ftp://`, or `ftps://` will be downloaded to a temp file if the processed version is not already available.
target_path	Character scalar; the path where the processed version of the file should be stored.
process_f	A function or one-sided formula to use to process the source file. `source_path` will be passed as the first argument to this function. Defaults to `read_f`.
process_args	An optional list of additional arguments to `process_f`.
write_f	A function or one-sided formula to use to save the processed file. The processed object will be passed as the first argument to this function, and `target_path` will be passed as the second argument. Defaults to `base::saveRDS()`.
write_args	An optional list of additional arguments to `write_f`.
force_process	A logical scalar indicating whether we should process the source file even if the target already exists. This can be particularly useful if you wish to redownload a file.

Value

The normalized `target_path`.

Examples

```
if (interactive()) {
  temp_filename <- tempfile()
  maybe_process(
    "https://query.data.world/s/owqxojjiphaypjmlxldsp566lck7co",
    target_path = temp_filename,
    process_f = read.csv
  )

  unlink(temp_filename)
}
```

read_or_cache *Read or Cache a File*

Description

This function wraps `read_or_process()`, specifying an app's cache directory as the target directory.

Usage

```
read_or_cache(
  source_path,
  appname,
  filename = construct_processed_filename(source_path),
  process_f = readRDS,
  process_args = NULL,
  read_f = readRDS,
  read_args = NULL,
  write_f = saveRDS,
  write_args = NULL,
  force_process = FALSE
)
```

Arguments

source_path	Character scalar; the path to the raw file. Paths starting with `http://`, `https://`, `ftp://`, or `ftps://` will be downloaded to a temp file if the processed version is not already available.
appname	Character; the name of the application that will "own" the cache, such as the name of a package.
filename	Character; an optional filename for the cached version of the file. By default, a filename is constructed using `construct_processed_filename()`.
process_f	A function or one-sided formula to use to process the source file. `source_path` will be passed as the first argument to this function. Defaults to `read_f`.
process_args	An optional list of additional arguments to `process_f`.
read_f	A function or one-sided formula to use to read the processed file. `target_path` will be passed as the first argument to this function. Defaults to `base::readRDS()`.
read_args	An optional list of additional arguments to `read_f`.
write_f	A function or one-sided formula to use to save the processed file. The processed object will be passed as the first argument to this function, and `target_path` will be passed as the second argument. Defaults to `base::saveRDS()`.
write_args	An optional list of additional arguments to `write_f`.
force_process	A logical scalar indicating whether we should process the source file even if the target already exists. This can be particularly useful if you wish to redownload a file.

Value

The processed object.

Examples

```
if (interactive()) {
  austin_smoke_free <- read_or_cache(
    "https://query.data.world/s/owqxojjiphaypjmlxldsp566lck7co",
    appname = "dlr",
    process_f = read.csv
  )
  head(austin_smoke_free)
}

if (interactive()) {
  # Calling the function a second time gives the result instantly.
  austin_smoke_free <- read_or_cache(
    "https://query.data.world/s/owqxojjiphaypjmlxldsp566lck7co",
    appname = "dlr",
    process_f = read.csv
  )
  head(austin_smoke_free)
}

if (interactive()) {
  # Remove the generated file.
  unlink(
    construct_cached_file_path(
      "https://query.data.world/s/owqxojjiphaypjmlxldsp566lck7co"
    )
  )
}
```

---

read_or_process *Read or Process a File*

Description

Often, a file must be processed before being usable in R. It can be useful to save the processed contents of that file in a standard format, such as RDS, so that the file does not need to be processed the next time it is loaded.

Usage

```
read_or_process(
  source_path,
  target_path,
  process_f = readRDS,
  process_args = NULL,
```

```
  read_f = readRDS,
  read_args = NULL,
  write_f = saveRDS,
  write_args = NULL,
  force_process = FALSE
)
```

Arguments

source_path	Character scalar; the path to the raw file. Paths starting with `http://`, `https://`, `ftp://`, or `ftps://` will be downloaded to a temp file if the processed version is not already available.
target_path	Character scalar; the path where the processed version of the file should be stored.
process_f	A function or one-sided formula to use to process the source file. `source_path` will be passed as the first argument to this function. Defaults to `read_f`.
process_args	An optional list of additional arguments to `process_f`.
read_f	A function or one-sided formula to use to read the processed file. `target_path` will be passed as the first argument to this function. Defaults to `base::readRDS()`.
read_args	An optional list of additional arguments to `read_f`.
write_f	A function or one-sided formula to use to save the processed file. The processed object will be passed as the first argument to this function, and `target_path` will be passed as the second argument. Defaults to `base::saveRDS()`.
write_args	An optional list of additional arguments to `write_f`.
force_process	A logical scalar indicating whether we should process the source file even if the target already exists. This can be particularly useful if you wish to redownload a file.

Value

The processed object.

Examples

```
if (interactive()) {
  temp_filename <- tempfile()
  austin_smoke_free <- read_or_process(
    "https://query.data.world/s/owqxojjiphaypjmlxldsp566lck7co",
    target_path = temp_filename,
    process_f = read.csv
  )
  head(austin_smoke_free)
}

# Calling the function a second time gives the result instantly.
if (interactive()) {
  austin_smoke_free <- read_or_process(
    "https://query.data.world/s/owqxojjiphaypjmlxldsp566lck7co",
```

```
    target_path = temp_filename,
    process_f = read.csv
  )
  head(austin_smoke_free)
}

if (interactive()) {
  # Remove the generated file.
  unlink(temp_filename)
}
```

`set_app_cache_dir` *Set a Cache Directory for an App*

Description

Override the default paths used by `app_cache_dir()`.

Usage

```
set_app_cache_dir(appname, cache_dir = NULL)
```

Arguments

`appname`	Character; the name of the application that will "own" the cache, such as the name of a package.
`cache_dir`	Character scalar; a path to a cache directory.

Value

A normalized path to a cache directory. The directory is created if the user has write access and the directory does not exist. An option is also set so future calls to `app_cache_dir()` will respect the change.

Examples

```
# Executing this function creates a cache directory.
set_app_cache_dir(appname = "dlr", cache_dir = "/my/cache/path")
```

`set_timeout` *Set Download Timeout*

Description

The default timeout for downloads is 60 seconds. This is not long enough for many of the files that are downloaded using this package. We therefore supply a convenience function to easily change this setting. You can permanently change this default by setting `R_DEFAULT_INTERNET_TIMEOUT` in your `.Renviron`.

Usage

```
set_timeout(seconds = 600L)
```

Arguments

`seconds`	The number of seconds to set as the timeout (default 600 seconds).

Value

A list with the old `timeout` setting (invisibly).

Examples

```
getOption("timeout")
old_setting <- set_timeout()
getOption("timeout")
options(old_setting)
```

Index