Package ‘dpkg’

January 8, 2025

Title Create, Stow, and Read Data Packages

Version 0.6.1

Description Data frame, tibble, or tbl objects are converted to data package objects using specific metadata labels (name, version, title, homepage, description). A data package object ('dpkg') can be written to disk as a 'parquet' file or released to a 'GitHub' repository. Data package objects can be read into R from online repositories and downloaded files are cached locally across R sessions.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Suggests dplyr, geoarrow, gert, gh, sf, testthat (>= 3.0.0), usethis, withr

Config/testthat/edition 3

URL `https://github.com/cole-brokamp/dpkg`, `https://cole-brokamp.github.io/dpkg/`

BugReports `https://github.com/cole-brokamp/dpkg/issues`

Imports arrow, cli, fs, glue, httr2, rlang, tibble

NeedsCompilation no

Author Cole Brokamp [aut, cre, cph] (<`https://orcid.org/0000-0002-0289-3151`>)

Maintainer Cole Brokamp <`cole@colebrokamp.com`>

Repository CRAN

Date/Publication 2025-01-08 21:30:07 UTC

Contents

as_dpkg *Use a data.frame and metadata to create a data package*

Description

Convert a data frame into a data package (dpkg) by providing specific metadata in the arguments.

Usage

```
as_dpkg(
  x,
  name = deparse(substitute(x)),
  version = "0.0.0.9000",
  title = character(),
  homepage = character(),
  description = character()
)
```

Arguments

x	a tibble or data frame
name	a lowercase character string consisting of only a-z, 0-9, -, _, or . to be used as a data package identifier
version	a character string representing a semantic version (e.g., "0.2.1")
title	a character string that is a title of the data package for humans
homepage	a valid URL that links to a webpage with code or descriptions related to creation of the data package
description	a character string (markdown encouraged!) of more details about how the data was created, including the data sources, references to code or packages used, relevant details for any specific columns, and notes about (mis)usage of the data

Details

name should be specified, but if is not will be deparsed from code defining x; this might not result in a valid name (e.g., when piping code to create a data frame)

Value

a dpkg object

Examples

```
x <- as_dpkg(mtcars, name = "mtcars", title = "Motor Trend Road Car Tests")
attr(x, "description") <- "This is a data set all about characteristics of different cars"
attr(x, "homepage") <- "https://github.com/cole-brokamp/dpkg"
x
```

`dpkg_gh_release` *Use a dpkg to create a github release*

Description

A GitHub release will be created based on the current commit, tagged and named according to the `name` and `version` of the dpkg. The dpkg `description` is used for the release body.

Usage

```
dpkg_gh_release(x, draft = TRUE)
```

Arguments

`x`	a data package (dpkg) object
`draft`	logical; mark release as draft?

Details

The `GITHUB_PAT` environment variable must be set and the working directory must be inside of a git repository with a GitHub remote.

The GitHub release will *not* be set to the latest release in order to prevent problems with other automated actions that rely on the latest release, like R universe or remotes `"*release"` syntax or other GitHub actions.

Release tags are required to be unique, so this will fail if a release with the same name and version already exists.

Value

the URL to the release (invisibly)

Examples

```
## Not run:
dpkg_gh_release(
  as_dpkg(mtcars,
    version = "0.0.0.9001", title = "Foofy Cars",
    homepage = "https://github.com/cole-brokamp/dpkg",
    description =
      paste("# Foofy Cars\n",
        "This is a test for the [dpkg](https://github.com/cole-brokamp/dpkg) package.",
        collapse = "\n"
      )
    ),
    draft = FALSE
  )

## End(Not run)
#> created release at: https://github.com/cole-brokamp/dpkg/releases/tag/mtcars-v0.0.0.9001
```

---

dpkg_meta *get the metadata associated with a data package*

Description

get the metadata associated with a data package

Usage

```
dpkg_meta(x)
```

Arguments

`x`	a dpkg object

Value

a list of metadata key value pairs

Examples

```
x <- as_dpkg(mtcars, name = "mtcars", title = "Motor Trend Road Car Tests")
attr(x, "description") <- "This is a data set all about characteristics of different cars"
attr(x, "homepage") <- "https://github.com/cole-brokamp/dpkg"
x

dpkg_meta(x)
```

`read_dpkg_metadata` *read (meta)data from dpkg on disk*

Description

read (meta)data from dpkg on disk

Usage

```
read_dpkg_metadata(x)

read_dpkg(x)
```

Arguments

x	path to data package (`.parquet` file) on disk

Value

for `read_dpkg()`, a dpkg object; for `read_dpkg_metadata()`, a list of metadata

Examples

```
d <- as_dpkg(mtcars, version = "0.1.0", title = "Motor Trend Road Car Tests")
attr(d, "description") <- "This is a data set all about characteristics of different cars"
attr(d, "homepage") <- "https://github.com/cole-brokamp/dpkg"

write_dpkg(d, dir = tempdir()) |>
  read_dpkg()

# geo objects are supported via the `geoarrow_vctr` in the geoarrow package
library(geoarrow)
sf::read_sf(system.file("gpkg/nc.gpkg", package = "sf")) |>
  as_dpkg(name = "nc_data") |>
  write_dpkg(tempdir())
d <- read_dpkg(fs::path_temp("nc_data-v0.0.0.9000.parquet"))
d

# as a simple features collection
d$geom <- sf::st_as_sfc(d$geom)
sf::st_as_sf(d)

# read just the metadata
read_dpkg_metadata(fs::path_temp("nc_data-v0.0.0.9000.parquet"))
```

stow_gh_release — *download a github release asset to the* stow *R user directory*

Description

Use stow to abstract away the process of downloading a file or a GitHub release asset to a user's data directory, only downloading files that have not already been downloaded.

Usage

```
stow_gh_release(owner, repo, dpkg, overwrite = FALSE)

stow(uri, overwrite = FALSE)

stow_url(url, overwrite = FALSE)
```

Arguments

`owner`	string of repo owner
`repo`	string of repo name
`dpkg`	string of gh release tag (will be the same as the filename without the `.parquet` extension)
`overwrite`	logical; re-download the remote file even though a local file with the same name exists?
`uri`	character string universal resource identifier; currently, must begin with `http://`, `https://`, or `gh://`
`url`	a URL string starting with `http://` or `https://`

Details

Supported URI prefixes include:

- `https://`, `http://`: download from a file
- `gh://`: download a github release asset, formatted as `gh://owner/repo/name`

Stow downloads files to the users data directory; see `?tools::R_user_dir`. Specify an alternative download location by setting the `R_USER_DATA_DIR` environment variable. The stow cache works by name only; that is, if a file with the same URI has already been downloaded once, it will not be re-downloaded again (unless `overwrite = TRUE`).

Value

path to the stowed file or url to github release

Examples

```
Sys.setenv(R_USER_DATA_DIR = tempfile("stow"))
# get by using URL
stow("https://github.com/geomarker-io/appc/releases/download/v0.1.0/nei_2020.rds",
     overwrite = TRUE) |>
  readRDS()

# will be faster (even in later R sessions) next time
stow("https://github.com/geomarker-io/appc/releases/download/v0.1.0/nei_2020.rds") |>
  readRDS()

# get a data package from a GitHub release
stow("gh://cole-brokamp/dpkg/mtcars-v0.0.0.9000", overwrite = TRUE) |>
  arrow::read_parquet()

stow("gh://cole-brokamp/dpkg/mtcars-v0.0.0.9000") |>
  arrow::read_parquet()
```

`stow_info` *get info about stowed files*

Description

get info about stowed files

get the path to a stowed file (or the stow directory)

test if a stowed file (or the stow directory) exists

get the size of a stowed file

remove a stowed file (or the stow entire directory)

Usage

```
stow_info(filename = NULL)

stow_path(filename = NULL)

stow_exists(filename = NULL)

stow_size(filename = NULL)

stow_remove(filename = NULL, .delete_stow_dir_confirm = FALSE)
```

Arguments

`filename`	character filename of stowed file; if NULL, then information about *all* stowed files or the directory where files are stowed is returned

`.delete_stow_dir_confirm`
set to TRUE in order to delete the entire stow directory without interactive user confirmation

Value

for `stow_info()`, a tibble of file or folder information; for `stow_path()`, a character path to the stowed file or stow directory; for `stow_exists()`, a logical; for `stow_size()`, a fs::

Examples

```
Sys.setenv(R_USER_DATA_DIR = tempfile("stow"))

stow_path()

stow("https://github.com/geomarker-io/appc/releases/download/v0.1.0/nei_2020.rds")

stow_path("nei_2020.rds")

stow_exists("nei_2020.rds")

stow_size("nei_2020.rds")

stow("https://github.com/geomarker-io/appc/releases/download/v0.1.0/nei_2017.rds")

stow_info("nei_2017.rds")

stow_info()

stow_size()

stow_remove(.delete_stow_dir_confirm = TRUE)
```

---

use_dpkg_badge *Use a markdown badge for a dpkg's latest github release*

Description

The badge relies on shields.io for the images, which will always display to the most recently released version and will link to the releases specific to the dpkg name.

Usage

```
use_dpkg_badge(x)
```

Arguments

`x`	a data package (dpkg) object

Details

Note that this relies on the structure of the release created with `dpkg_gh_release()`, but relies on a dpkg object *before* it is released. This will lead to broken release badges and links until an initial dpkg release is created with `dpkg_gh_release()`.

Value

character string of markdown

Examples

```
## Not run:
as_dpkg(mtcars,
  version = "0.0.0.9000", title = "Foofy Cars",
  homepage = "https://github.com/cole-brokamp/dpkg",
  description =
    paste("# Foofy Cars\n",
      "This is a test for the [dpkg](https://github.com/cole-brokamp/dpkg) package.",
      collapse = "\n"
    )
) |>
  use_dpkg_badge()

## End(Not run)
```

`write_dpkg` *write dpkg to disk*

Description

write dpkg to disk

Usage

```
write_dpkg(x, dir)
```

Arguments

`x`	a data package (dpkg) object
`dir`	path to directory where dpkg parquet file will be written

Value

path to the written file, invisibly

Index