Package: DataPackageR (via r-universe)

July 3, 2024

Type Package

Title Construct Reproducible Analytic Data Sets as R Packages

Version 0.16.0

Description A framework to help construct R data packages in a reproducible manner. Potentially time consuming processing of raw data sets into analysis ready data sets is done in a reproducible manner and decoupled from the usual 'R CMD build' process so that data sets can be processed into R objects in the data package and the data package can then be shared, built, and installed by others without the need to repeat computationally costly data processing. The package maintains data provenance by turning the data processing scripts into package vignettes, as well as enforcing documentation and version checking of included data objects. Data packages can be version controlled on 'GitHub', and used to share data for manuscripts, collaboration and reproducible research.

License MIT + file LICENSE

URL https://github.com/ropensci/DataPackageR, https://docs.ropensci.org/DataPackageR/

BugReports https://github.com/ropensci/DataPackageR/issues

Depends R (>= 3.5.0)

Imports crayon, desc, digest, futile.logger, knitr, pkgbuild, pkgload, rmarkdown, roxygen2, rprojroot, usethis, utils, yaml

Suggests covr, data.tree, spelling, testthat, withr

VignetteBuilder knitr

Encoding UTF-8

Language en-US

RoxygenNote 7.3.1

SystemRequirements pandoc - https://pandoc.org

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

RemoteUrl https://github.com/ropensci/DataPackageR

RemoteRef main

RemoteSha 37d517aeea2445779c1d21b5424374dee0112f52

Contents

`assert_data_version` *Assert that a data version in a data package matches an expectation.*

Description

Assert that a data version in a data package matches an expectation.

Usage

```
assert_data_version(
  data_package_name = NULL,
  version_string = NULL,
  acceptable = "equal",
  ...
)
```

Arguments

data_package_name	character Name of the package.
version_string	character Version string in "x.y.z" format.
acceptable	character one of "equal", "equal_or_greater", describing what version match is acceptable.
...	additional arguments passed to data_version (such as lib.loc)

Details

Tests the DataVersion string in data_package_name against version_string testing the major, minor and revision portion.

Tests "data_package_name version equal version_string" or "data_package_name version equal_or_greater version_string".

Value

invisible logical TRUE if success, otherwise stop on mismatch.

Examples

```
if(rmarkdown::pandoc_available()){
f <- tempdir()
f <- file.path(f, "foo.Rmd")
con <- file(f)
writeLines("```{r}\n vec = 1:10 \n```\n",con = con)
close(con)
pname <- basename(tempfile())
datapackage_skeleton(name = pname,
   path=tempdir(),
   force = TRUE,
   r_object_names = "vec",
   code_files = f)
package_build(file.path(tempdir(),pname), install = FALSE)

pkgload::load_all(file.path(tempdir(),pname))

assert_data_version(data_package_name = pname,version_string = "0.1.0",acceptable = "equal")
}
```

---

construct_yml_config *Construct a datapackager.yml configuration*

Description

Constructs a datapackager.yml configuration object from a vector of file names and a vector of object names (all quoted). Can be written to disk via yml_write. render_root is set to a randomly generated named subdirectory of tempdir().

Usage

```
construct_yml_config(code = NULL, data = NULL, render_root = NULL)
```

Arguments

code	A vector of filenames
data	A vector of quoted object names
render_root	The root directory where the package data processing code will be rendered. Defaults to is set to a randomly generated named subdirectory of `tempdir()`.

Value

a datapackager.yml configuration represented as an R object

Examples

```
conf <- construct_yml_config(code = c('file1.rmd','file2.rmd'), data=c('object1','object2'))
tmp <- normalizePath(tempdir(), winslash = "/")
yml_write(conf,path=tmp)
```

DataPackageR-defunct *Defunct functions in package* **DataPackageR**.

Description

These functions are defunct and no longer supported. Calling them will result in an error.

When possible, alternatives are suggested.

Usage

```
datapackage.skeleton(...)

dataVersion(...)

keepDataObjects(...)
```

Arguments

...	All arguments are now ignored.

Value

Defunct function. No return value.

datapackager_object_read *Read an object created in a previously run processing script.*

Description

Read an object created in a previously run processing script.

Usage

```
datapackager_object_read(name)
```

Arguments

name	character the name of the object. Must be a name available in the configuration objects. Other objects are not saved.

Details

This function is only accessible within an R or Rmd file processed by DataPackageR. It searches for an environment named ENVS within the current environment, that holds the object with the given name. Such an environment is constructed and populated with objects specified in the yaml objects property and passed along to subsequent R and Rmd files as DataPackageR processes them in order.

Value

An R object.

Examples

```
if(rmarkdown::pandoc_available()){
ENVS <- new.env() # ENVS would be in the environment
                  # where the data processing is run. It is
                  # handled automatically by the package.
assign("find_me", 100, ENVS) #This is done automatically by DataPackageR

find_me <- datapackager_object_read("find_me") # This would appear in an Rmd processed by
                               # DataPackageR to access the object named "find_me" created
                                     # by a previous script. "find_me" would also need to
                                     # appear in the objects property of datapackager.yml
}
```

DataPackageR_options *Options consulted by DataPackageR*

Description

User-configurable options consulted by DataPackageR, which provide a mechanism for setting default behaviors for various functions.

If the built-in defaults don't suit you, set one or more of these options. Typically, this is done in the `.Rprofile` startup file, which you can open for editing with `usethis::edit_r_profile()` - this will set the specified options for all future R sessions. The following setting is recommended to not be prompted upon each package build for a NEWS update:

```
options(DataPackageR_interact = FALSE)
```

Options for the DataPackageR package

- `DataPackageR_interact`: Upon package load, this defaults to the value of `interactive()`, unless the option has been previously set (e.g., in `.Rprofile`). TRUE prompts user interactively for a NEWS update on `package_build()`. See the example above and the rOpenSci blog post for more details on how to set this to FALSE, which will never prompt user for a NEWS update. FALSE is also the setting used for DataPackageR internal package tests.

- `DataPackageR_verbose`: Default upon package load is TRUE. FALSE suppresses all console output and is currently only used for automated unit tests of the DataPackageR package.

- `DataPackageR_packagebuilding`: Default upon package load is FALSE. This option is used internally for package operations and changing it is not recommended.

datapackage_skeleton *Create a Data Package skeleton for use with DataPackageR.*

Description

Creates a package skeleton directory structure for use with DataPackageR. Adds the DataVersion string to DESCRIPTION, creates the DATADIGEST file, and the data-raw directory. Updates the Read-and-delete-me file to reflect the additional necessary steps.

Usage

```
datapackage_skeleton(
  name = NULL,
  path = ".",
  force = FALSE,
  code_files = character(),
  r_object_names = character(),
  raw_data_dir = character(),
  dependencies = character()
)
```

Arguments

name	character name of the package to create.
path	A character path where the package is located. See package.skeleton
force	logical Force the package skeleton to be recreated even if it exists. see package.skeleton
code_files	Optional character vector of paths to Rmd files that process raw data into R objects.
r_object_names	vector of quoted r object names , tables, etc. created when the files in code_files are run.
raw_data_dir	character pointing to a raw data directory. Will be moved with all its subdirectories to "inst/extdata"
dependencies	vector of character, paths to R files that will be moved to "data-raw" but not included in the yaml config file. e.g., dependency scripts.

Value

No return value, called for side effects

Examples

```
if(rmarkdown::pandoc_available()){
f <- tempdir()
f <- file.path(f,"foo.Rmd")
con <- file(f)
writeLines("```{r}\n tbl = data.frame(1:10) \n```\n",con=con)
close(con)
pname <- basename(tempfile())
datapackage_skeleton(name = pname,
   path = tempdir(),
   force = TRUE,
   r_object_names = "tbl",
   code_files = f)
   }
```

---

data_version *Get the DataVersion for a package*

Description

Retrieves the DataVersion of a package if available

Usage

```
data_version(pkg, lib.loc = NULL)
```

Arguments

pkg	character the package name
lib.loc	character path to library location.

Value

Object of class 'package_version' and 'numeric_version' specifying the DataVersion of the package

See Also

packageVersion

Examples

```
if(rmarkdown::pandoc_available()){
f <- tempdir()
f <- file.path(f,"foo.Rmd")
con <- file(f)
writeLines("```{r}\n vec = 1:10 \n```\n",con=con)
close(con)
pname <- basename(tempfile())
datapackage_skeleton(name = pname,
   path=tempdir(),
   force = TRUE,
   r_object_names = "vec",
   code_files = f)

   package_build(file.path(tempdir(),pname), install = FALSE)

   pkgload::load_all(file.path(tempdir(),pname))
   data_version(pname)
}
```

---

document *Build documentation for a data package using DataPackageR.*

Description

Build documentation for a data package using DataPackageR.

Usage

```
document(path = ".", install = FALSE, ...)
```

Arguments

path	character the path to the data package source root.
install	logical install the package. (default FALSE)
...	additional arguments to install

Value

Called for side effects. Returns TRUE on successful exit.

Examples

```
# A simple Rmd file that creates one data object
# named "tbl".
if(rmarkdown::pandoc_available()){
f <- tempdir()
f <- file.path(f,"foo.Rmd")
con <- file(f)
writeLines("```{r}\n tbl = data.frame(1:10) \n```\n",con=con)
close(con)

# construct a data package skeleton named "MyDataPackage" and pass
# in the Rmd file name with full path, and the name of the object(s) it
# creates.

pname <- basename(tempfile())
datapackage_skeleton(name=pname,
   path=tempdir(),
   force = TRUE,
   r_object_names = "tbl",
   code_files = f)

# call package_build to run the "foo.Rmd" processing and
# build a data package.
package_build(file.path(tempdir(), pname), install = FALSE)
document(path = file.path(tempdir(), pname), install = FALSE)

}
```

---

package_build *Pre-process, document and build a data package*

Description

Combines the preprocessing, documentation, and build steps into one.

Usage

```
package_build(
  packageName = NULL,
  vignettes = FALSE,
  log = INFO,
  deps = TRUE,
  install = FALSE,
  ...
)
```

Arguments

packageName	character path to package source directory. Defaults to the current path when NULL.
vignettes	logical specify whether to build vignettes. Default FALSE.
log	log level INFO,WARN,DEBUG,FATAL
deps	logical should we pass data objects into subsequent scripts? Default TRUE
install	logical automatically install and load the package after building. Default FALSE
...	additional arguments passed to install.packages when install=TRUE.

Details

Note that if package_build returns an error when rendering an .Rmd internally, but that same .Rmd can be run successfully manually using rmarkdown::render, then the following code facilitates debugging. Set options(error = function(){ sink(); recover()}) before running package_build . This will enable examination of the active function calls at the time of the error, with output printed to the console rather than knitr's default sink. After debugging, evaluate options(error = NULL) to revert to default error handling. See section "22.5.3 RMarkdown" at https://adv-r.hadley.nz/debugging.html for more details.

Value

Character vector. File path of the built package.

Examples

```
if(rmarkdown::pandoc_available()){
f <- tempdir()
f <- file.path(f,"foo.Rmd")
con <- file(f)
writeLines("```{r}\n tbl = data.frame(1:10) \n```\n",con=con)
close(con)
pname <- basename(tempfile())
datapackage_skeleton(name=pname,
   path=tempdir(),
   force = TRUE,
   r_object_names = "tbl",
   code_files = f)

package_build(file.path(tempdir(),pname), install = FALSE)
}
```

project_data_path *Get DataPackageR data path*

Description

Get DataPackageR data path

Usage

```
project_data_path(file = NULL)
```

Arguments

file	character or NULL (default).

Details

Returns the path to the data package data subdirectory, or constructs a path to a file in the data subdirectory from the file argument.

Value

character

Examples

```
if(rmarkdown::pandoc_available()){
project_data_path( file = "data.rda" )
}
```

project_extdata_path *Get DataPackageR extdata path*

Description

Get DataPackageR extdata path

Usage

```
project_extdata_path(file = NULL)
```

Arguments

file	character or NULL (default).

Details

Returns the path to the data package extdata subdirectory, or constructs a path to a file in the extdata subdirectory from the file argument.

Value

`character`

Examples

```
if(rmarkdown::pandoc_available()){
project_extdata_path(file = "mydata.csv")
}
```

`project_path` *Get DataPackageR Project Root Path*

Description

Get DataPackageR Project Root Path

Usage

```
project_path(file = NULL)
```

Arguments

`file`	`character` or `NULL` (default).

Details

Returns the path to the data package project root, or constructs a path to a file in the project root from the file argument.

Value

`character`

Examples

```
if(rmarkdown::pandoc_available()){
project_path( file = "DESCRIPTION" )
}
```

use_data_object *Add a data object to a data package.*

Description

The data object will be added to the yml configuration file.

Usage

```
use_data_object(object_name = NULL)
```

Arguments

object_name	Name of the data object. Should be created by a processing script in data-raw. character vector of length 1.

Value

invisibly returns TRUE for success.

Examples

```
if(rmarkdown::pandoc_available()){
myfile <- tempfile()
file <- system.file("extdata", "tests", "extra.Rmd",
                    package = "DataPackageR")
datapackage_skeleton(
  name = "datatest",
  path = tempdir(),
  code_files = file,
  force = TRUE,
  r_object_names = "data")
use_data_object(object_name = "newobject")
}
```

use_ignore *Ignore specific files by git and R build.*

Description

Ignore specific files by git and R build.

Usage

```
use_ignore(file = NULL, path = NULL)
```

Arguments

file	character File to ignore.
path	character Path to the file.

Value

invisibly returns 0.

Examples

```
datapackage_skeleton(name="test",path = tempdir())
use_ignore("foo", ".")
```

---

use_processing_script *Add a processing script to a data package.*

Description

The Rmd or R file or directory specified by `file` will be moved into the data-raw directory. It will also be added to the yml configuration file. Any existing file by that name will be overwritten when overwrite is set to TRUE

Usage

```
use_processing_script(
  file = NULL,
  title = NULL,
  author = NULL,
  overwrite = FALSE
)
```

Arguments

file	character path to an existing file or name of a new R or Rmd file to create.
title	character title of the processing script for the yaml header. Used only if file is being created.
author	character author name for the yaml header. Used only if the file is being created.
overwrite	logical default FALSE. Overwrite existing file of the same name.

Value

invisibly returns TRUE for success. Stops on failure.

Examples

```
if(rmarkdown::pandoc_available()){
myfile <- tempfile()
file <- system.file("extdata", "tests", "extra.Rmd",
                    package = "DataPackageR")
datapackage_skeleton(
  name = "datatest",
  path = tempdir(),
  code_files = file,
  force = TRUE,
  r_object_names = "data")
use_processing_script(file = "newScript.Rmd",
    title = "Processing a new dataset",
    author = "Y.N. Here.")
}
```

`use_raw_dataset` *Add a raw data set to inst/extdata*

Description

The file or directory specified by `path` will be moved into the inst/extdata directory.

Usage

```
use_raw_dataset(path = NULL, ignore = FALSE)
```

Arguments

`path`	`character` path to file or directory.
`ignore`	`logical` whether to ignore the path or file in git and R build.

Value

invisibly returns TRUE for success. Stops on failure.

Examples

```
if(rmarkdown::pandoc_available()){
myfile <- tempfile()
file <- system.file("extdata", "tests", "extra.Rmd",
                    package = "DataPackageR")
raw_data <- system.file("extdata", "tests", "raw_data",
                        package = "DataPackageR")
datapackage_skeleton(
  name = "datatest",
  path = tempdir(),
  code_files = file,
  force = TRUE,
```

```
  r_object_names = "data")
use_raw_dataset(raw_data)
}
```

`yml_find` *Edit DataPackageR yaml configuration*

Description

Edit a yaml configuration file via an API.

Usage

```
yml_find(path)

yml_add_files(config, filenames)

yml_disable_compile(config, filenames)

yml_enable_compile(config, filenames)

yml_add_objects(config, objects)

yml_list_objects(config)

yml_list_files(config)

yml_remove_objects(config, objects)

yml_remove_files(config, filenames)

yml_write(config, path = NULL)
```

Arguments

`path`	Path to the data package source or path to write config file (for `yml_write`)
`config`	an R representation of the datapackager.yml config, returned by yml_find, or a path to the package root.
`filenames`	A vector of filenames.
`objects`	A vector of R object names.

Details

Add, remove files and objects, enable or disable parsing of specific files, list objects or files in a yaml config, or write a config back to a package.

Value

A yaml configuration structured as an R nested list.

Examples

```
if(rmarkdown::pandoc_available()){
f <- tempdir()
f <- file.path(f,"foo.Rmd")
con <- file(f)
writeLines("```{r}\n vec = 1:10\n```\n",con=con)
close(con)
pname <- basename(tempfile())
datapackage_skeleton(name=pname,
   path = tempdir(),
   force = TRUE,
   r_object_names = "vec",
   code_files = f)
yml <- yml_find(file.path(tempdir(),pname))
yml <- yml_add_files(yml,"foo.Rmd")
yml_list_files(yml)
yml <- yml_disable_compile(yml,"foo.Rmd")
yml <- yml_enable_compile(yml,"foo.Rmd")
yml <- yml_add_objects(yml,"data1")
yml_list_objects(yml)
yml <- yml_remove_objects(yml,"data1")
yml <- yml_remove_files(yml,"foo.Rmd")
}
```

Index