Package 'One4All'

January 20, 2025

Type Package

Title Validate, Share, and Download Data

Version 0.5

Date 2024-07-02

Description Designed to enhance data validation and management processes by employing a set of functions that read a set of rules from a 'CSV' or 'Excel' file and apply them to a dataset. Funded by the National Renewable Energy Laboratory and Possibility Lab, maintained by the Moore Institute for Plastic Pollution Research.

URL https://github.com/Moore-Institute-4-Plastic-Pollution-Res/One4All, https://moore-institute-4-plastic-pollution-res.github.io/One4All/

BugReports https://github.com/Moore-Institute-4-Plastic-Pollution-Res/One4All/issues

License MIT + file LICENSE

Encoding UTF-8

LazyLoad true

LazyData true

VignetteBuilder knitr

Depends R (>= 4.0.0)

Imports shiny, dplyr, validate, digest, data.table, ckanr, openxlsx, lexicon, readr, readxl, tibble, aws.s3, rlang, jsonlite, mongolite, httr

Suggests knitr, rmarkdown, testthat (>= 3.0.0), config, DT, shinythemes, shinyWidgets, bs4Dash, shinyjs, listviewer, RCurl, purrr, stringr

RoxygenNote 7.3.1

Config/testthat/edition 3

NeedsCompilation no

Author Hannah Sherrod [cre, aut] (<https://orcid.org/0009-0001-0497-8693>),
Nick Leong [aut] (<https://orcid.org/0009-0008-3313-4132>),
Hannah Hapich [aut] (<https://orcid.org/0000-0003-0000-6632>),
Fabian Gomez [aut],
Shelly Moore [aut],
Win Cowger [aut] (<https://orcid.org/0000-0001-9226-3104>),
Scott Coffin [ctb],
Tony Hale [ctb],
Diana Lin [ctb],
Gemma Shusterman [ctb],
Rebecca Sutton [ctb],
Adam Wong [ctb],
Haig Minasian [ctb],
Holden Ford [ctb],
Anja Oca [ctb],
Richard Nelson [ctb],
Leah Thornton Hampton [ctb],
Libby Heeren [ctb],
Gabriel Daiess [ctb]

Maintainer Hannah Sherrod <hannah@mooreplasticresearch.org>

Repository CRAN

Date/Publication 2024-07-02 23:00:02 UTC

Contents

certificate_df . . . 3
checkLuhn . . . 3
check_exists_in_zip . . . 4
check_for_malicious_files . . . 5
check_images . . . 5
check_other_hyperlinks . . . 6
create_valid_excel . . . 7
download_all . . . 8
invalid_example . . . 9
is.POSIXct . . . 9
name_data . . . 10
query_document_by_object_id . . . 10
read_data . . . 11
read_rules . . . 12
reformat_rules . . . 12
remote_download . . . 13
remote_raw_download . . . 14
remote_share . . . 15
rows_for_rules . . . 17
rules_broken . . . 18
run_app . . . 19
test_profanity . . . 20

test_rules 21
validate_data 21
valid_example 22

Index **24**

`certificate_df` *Generate a data frame with certificate information*

Description

This function creates a data frame with certificate information including the current time, data and rule hashes, package version, and web hash.

Usage

```
certificate_df(x, time = Sys.time())
```

Arguments

`x`	A list containing 'data_formatted' and 'rules' elements.
`time`	the time the certificate is generated, can be passed a value or uses current system time.

Value

A data frame with certificate information.

Examples

```
certificate_df(x = list(data_formatted = data.frame(a = 1:3, b = 4:6),
                        rules = validate::validator(a > 0, b > 0)),
               time = Sys.time())
```

`checkLuhn` *Check if a number passes the Luhn algorithm*

Description

This function checks if a given number passes the Luhn algorithm. It is commonly used to validate credit card numbers.

Usage

```
checkLuhn(number)
```

Arguments

number	A character string of the number to check against the Luhn algorithm.

Value

A logical value indicating whether the number passes the Luhn algorithm (TRUE) or not (FALSE).

Examples

```
checkLuhn("4532015112830366") # TRUE
checkLuhn("4532015112830367") # FALSE
```

---

check_exists_in_zip *Check if a file exists in a zip file*

Description

This function checks if a file with a given name exists in a specified zip file.

Usage

```
check_exists_in_zip(zip_path, file_name)
```

Arguments

zip_path	A character string representing the path of the zip file.
file_name	A character string representing the name of the file to check.

Value

A logical value indicating whether the file exists in the zip file (TRUE) or not (FALSE).

Examples

```
## Not run:
check_exists_in_zip(zip_path = "/path/to/your.zip", file_name = "file/in/zip.csv")

## End(Not run)
```

check_for_malicious_files *Check for malicious files*

Description

This function checks for the presence of files with extensions known to be associated with malicious activities. The function can be used to screen zip files or individual files for these potentially dangerous file types.

Usage

```
check_for_malicious_files(files)
```

Arguments

files	A character vector of file paths. These can be paths to zip files or individual files.

Value

A logical value indicating if any of the files in the input have a malicious file extension. Returns 'TRUE' if any malicious file is found, otherwise 'FALSE'.

Examples

```
## Not run:
  check_for_malicious_files("path'(s)'/to/your/files")
  check_for_malicious_files(utils::unzip("path/to/your/file.zip", list = TRUE)$Name)

## End(Not run)
```

check_images *Check and format image URLs*

Description

This function checks if the input string contains an image URL (PNG or JPG) and formats it as an HTML img tag with a specified height.

Usage

```
check_images(x)
```

Arguments

x	A character string to check for image URLs.

Value

A character string with the HTML img tag if an image URL is found, otherwise the input string.

Examples

```
check_images("https://example.com/image.png")
check_images("https://example.com/image.jpg")
check_images("https://example.com/text")
```

---

check_other_hyperlinks

Check and format non-image hyperlinks

Description

This function checks if the input string contains a non-image hyperlink and formats it as an HTML anchor tag.

Usage

```
check_other_hyperlinks(x)
```

Arguments

x	A character string to check for non-image hyperlinks.

Value

A character string with the HTML anchor tag if a non-image hyperlink is found, otherwise the input string.

Examples

```
check_other_hyperlinks("https://example.com/page")
check_other_hyperlinks("https://example.com/image.png")
check_other_hyperlinks("https://example.com/image.jpg")
```

`create_valid_excel` *Create a formatted Excel file based on validation rules*

Description

This function creates an Excel file with conditional formatting and data validation based on the given validation rules in a CSV or Excel file. This function is currently compatible with Windows and Linux operating systems. When using a macOS system, the excel file is able to download, but has some bugs with formatting the LOOKUP sheet.

Usage

```
create_valid_excel(
  file_rules,
  negStyle = createStyle(fontColour = "#9C0006", bgFill = "#FFC7CE"),
  posStyle = createStyle(fontColour = "#006100", bgFill = "#C6EFCE"),
  row_num = 1000
)
```

Arguments

`file_rules`	A CSV or Excel file containing validation rules.
`negStyle`	Style to apply for negative conditions (default is red text on a pink background).
`posStyle`	Style to apply for positive conditions (default is green text on a light green background).
`row_num`	Number of rows to create in the output file (default is 1000).

Value

A workbook object containing the formatted Excel file.

Examples

```
data("test_rules")
create_valid_excel(file_rules = test_rules)
```

download_all *Download all data alternative*

Description

This function allows users to download all data rather than one data set at a time.

Usage

```
download_all(
  file_path = NULL,
  s3_key_id = NULL,
  s3_secret_key = NULL,
  s3_region = NULL,
  s3_bucket = NULL,
  callback = NULL
)
```

Arguments

file_path	location and name of the zip file to create.
s3_key_id	A character string representing the AWS S3 access key ID.
s3_secret_key	A character string representing the AWS S3 secret access key.
s3_region	A character string representing the AWS S3 region.
s3_bucket	A character string representing the AWS S3 bucket name.
callback	Prints if the download was a success.

Value

Any return objects from the downloads.

Examples

```
## Not run:
    download_all_data <- download_all(file_path = "your/path/file.zip",
                                      s3_key_id = "your_s3_key_id",
                                      s3_secret_key = "your_s3_secret_key",
                                      s3_region = "your_s3_region",
                                      s3_bucket = "your_s3_bucket",
                                      callback = NULL)

## End(Not run)
```

invalid_example *Invalid example data*

Description

This is a list containing three data frames as an example of invalid_example.

Format

A list with 3 data frames:

data_frame_1 A data frame with 18 variables: MethodologyID, SamplingDevice, AirFiltration, AirFiltrationType, ClothingPolicy, NonplasticPolicy, SealedEnvironment, SealedEnvironmentType, SieveMeshSizes, FilterType, FilterDiameter, FilterPoreSize, VisIDMethod, VisualSoftware, PickingStrategy, VisMagnification, MatIDMethod, MatIDSoftware

data_frame_2 A data frame with 8 variables: SampleID, OwnerOrganization, AnalysisOrganization, ReportingOrganization, Latitude, Longitude, CollectionDate, SampleVolume

data_frame_3 A data frame with 17 variables: ParticleID, MethodologyID, SampleID, PhotoID, SpectraID, FinalAnalysisDate, Comments, Polymer, Morphology, Color, Length, Width, Height, Mass, SurfaceArea, Volume, Tactile

Examples

```
data("invalid_example")
```

is.POSIXct *Check if an object is of class POSIXct*

Description

This function checks if the given object is of class POSIXct. It returns TRUE if the object inherits the POSIXct class, otherwise FALSE.

Usage

```
is.POSIXct(x)
```

Arguments

x	An object to be tested for POSIXct class inheritance.

Value

A logical value indicating if the input object is of class POSIXct.

Examples

```
x <- as.POSIXct("2021-01-01")
is.POSIXct(x) # TRUE

y <- Sys.Date()
is.POSIXct(y) # FALSE
```

`name_data` *Name datasets*

Description

This function extracts the names of the datasets provided in the input files. If specific data names are provided, they are used, otherwise the function tries to extract the names from the files themselves.

Usage

```
name_data(files_data, data_names = NULL)
```

Arguments

`files_data`	A vector of file paths or list of data frames.
`data_names`	A vector of names to be assigned to datasets.

Value

A vector of dataset names.

Examples

```
name_data(files_data = c("path/to/data1.csv", "path/to/data2.csv"))
name_data(files_data = c("path/to/data.xlsx"), data_names = c("sheet1", "sheet2"))
```

`query_document_by_object_id`
Query a MongoDB document by an ObjectID

Description

This function queries a mongodb database using its API to retrieve a document by its ObjectID. Use the MongoDB Atlas Data API to create an API key.

Usage

```
query_document_by_object_id(apiKey, collection, database, dataSource, objectId)
```

Arguments

apiKey	The API key for accessing the MongoDB API.
collection	The name of the collection in the MongoDB database.
database	The name of the MongoDB database.
dataSource	The data source in MongoDB.
objectId	The object ID of the document to query.

Value

The queried document.

Examples

```
## Not run:
apiKey <- 'your_mongodb_api_key'
collection <- 'your_mongodb_collection'
database <- 'your_database'
dataSource <- 'your_dataSource'
objectId <- 'example_object_id'
query_document_by_object_id(apiKey, collection, database, dataSource, objectId)

## End(Not run)
```

---

read_data *Read and format data from csv or xlsx files*

Description

Read and format data from csv or xlsx files

Usage

```
read_data(files_data, data_names = NULL)
```

Arguments

files_data	List of files to be read
data_names	Optional vector of names for the data frames

Value

A list of data frames

Examples

```
read_data(files_data = valid_example, data_names = c("methodology", "particles", "samples"))
```

read_rules *Read rules from a file*

Description

This function reads rules from a file or a data frame. The file can be in csv or xlsx format. The data should have the column names "name", "description", "dataset", "valid example", "severity", "rule". The function also checks that the rules do not contain sensitive words and that all the rules fields are character type.

Usage

```
read_rules(file_rules)
```

Arguments

file_rules	The file containing the rules. Can be a CSV or XLSX file, or a data frame.

Value

A data frame containing the rules.

Examples

```
## Not run:
read_rules("path/to/rules")

## End(Not run)
```

reformat_rules *Reformat the rules*

Description

This function is responsible for handling the rule reformating, dataset handling and foreign key checks.

Usage

```
reformat_rules(rules, data_formatted, zip_data = NULL)
```

Arguments

rules	A data.frame containing rules to be reformatted.
data_formatted	A named list of data.frames with data.
zip_data	A file path to a zip folder with additional data to check.

Value

A data.frame with reformatted rules.

Examples

```
data("test_rules")
data("valid_example")
reformat_rules(rules = test_rules, data_formatted = valid_example)
```

`remote_download` *Download structured data from remote sources*

Description

This function downloads data from remote sources like CKAN, AWS S3, and MongoDB. It retrieves the data based on the hashed_data identifier and assumes the data is stored using the same naming conventions provided in the 'remote_share' function.

Usage

```
remote_download(
  hashed_data = NULL,
  ckan_url,
  ckan_key,
  ckan_package,
  s3_key_id,
  s3_secret_key,
  s3_region,
  s3_bucket,
  mongo_key,
  mongo_collection
)
```

Arguments

`hashed_data`	A character string representing the hashed identifier of the data to be downloaded.
`ckan_url`	A character string representing the CKAN base URL.
`ckan_key`	A character string representing the CKAN API key.
`ckan_package`	A character string representing the CKAN package identifier.
`s3_key_id`	A character string representing the AWS S3 access key ID.
`s3_secret_key`	A character string representing the AWS S3 secret access key.
`s3_region`	A character string representing the AWS S3 region.
`s3_bucket`	A character string representing the AWS S3 bucket name.
`mongo_key`	A character string representing the mongo key.
`mongo_collection`	A character string representing the mongo collection.

Value

A named list containing the downloaded datasets.

Examples

```
## Not run:
  downloaded_data <- remote_download(hashed_data = "example_hash",
                                     ckan_url = "https://example.com",
                                     ckan_key = "your_ckan_key",
                                     ckan_package = "your_ckan_package",
                                     s3_key_id = "your_s3_key_id",
                                     s3_secret_key = "your_s3_secret_key",
                                     s3_region = "your_s3_region",
                                     s3_bucket = "your_s3_bucket",
                                     mongo_key = "mongo_key",
                                     mongo_collection = "mongo_collection")

## End(Not run)
```

---

remote_raw_download *Download raw data from remote sources*

Description

This function downloads data from remote sources like CKAN and AWS S3. It retrieves the data based on the hashed_data identifier and assumes the data is stored using the same naming conventions provided in the ‘remote_share‘ function.

Usage

```
remote_raw_download(
  hashed_data = NULL,
  file_path = NULL,
  ckan_url = NULL,
  ckan_key = NULL,
  ckan_package = NULL,
  s3_key_id = NULL,
  s3_secret_key = NULL,
  s3_region = NULL,
  s3_bucket = NULL
)
```

Arguments

hashed_data	A character string representing the hashed identifier of the data to be downloaded.
file_path	location and name of the zip file to create.

ckan_url	A character string representing the CKAN base URL.
ckan_key	A character string representing the CKAN API key.
ckan_package	A character string representing the CKAN package identifier.
s3_key_id	A character string representing the AWS S3 access key ID.
s3_secret_key	A character string representing the AWS S3 secret access key.
s3_region	A character string representing the AWS S3 region.
s3_bucket	A character string representing the AWS S3 bucket name.

Value

Any return objects from the downloads.

Examples

```
## Not run:
  downloaded_data <- remote_raw_download(hashed_data = "example_hash",
                                         file_path = "your/path/file.zip",
                                         ckan_url = "https://example.com",
                                         ckan_key = "your_ckan_key",
                                         ckan_package = "your_ckan_package",
                                         s3_key_id = "your_s3_key_id",
                                         s3_secret_key = "your_s3_secret_key",
                                         s3_region = "your_s3_region",
                                         s3_bucket = "your_s3_bucket")

## End(Not run)
```

---

remote_share *Share your validated data*

Description

This function uploads validated data to specified remote repositories, such as CKAN, Amazon S3, and/or MongoDB.

Usage

```
remote_share(
  validation,
  data_formatted,
  files,
  verified,
  valid_rules,
  valid_key_share,
  ckan_url,
  ckan_key,
```

```
  ckan_package,
  url_to_send,
  rules,
  results,
  s3_key_id,
  s3_secret_key,
  s3_region,
  s3_bucket,
  mongo_key,
  mongo_collection,
  old_cert = NULL
)
```

Arguments

validation	A list containing validation information.
data_formatted	A list containing formatted data.
files	A vector of file paths to upload.
verified	The secret key provided by the portal maintainer.
valid_rules	A list of valid rules for the dataset.
valid_key_share	A valid key to share data.
ckan_url	The URL of the CKAN instance.
ckan_key	The API key for the CKAN instance.
ckan_package	The CKAN package to which the data will be uploaded.
url_to_send	The URL to send the data.
rules	A set of rules used for validation.
results	A list containing results of the validation.
s3_key_id	AWS ACCESS KEY ID
s3_secret_key	AWS SECRET ACCESS KEY
s3_region	AWS DEFAULT REGION
s3_bucket	The name of the Amazon S3 bucket.
mongo_key	mongo connection url
mongo_collection	collection name
old_cert	(Optional) An old certificate to be uploaded alongside the new one to override the previous submission with.

Value

A list containing the status and message of the operation.

Examples

```
## Not run:
shared_data <- remote_share(validation = result_valid,
                            data_formatted = result_valid$data_formatted,
                            files = test_file,
                            verified = "your_verified_key",
                            valid_key_share = "your_valid_key_share",
                            valid_rules = digest::digest(test_rules),
                            ckan_url = "https://example.com",
                            ckan_key = "your_ckan_key",
                            ckan_package = "your_ckan_package",
                            url_to_send = "https://your-url-to-send.com",
                            rules = test_rules,
                            results = valid_example$results,
                            s3_key_id = "your_s3_key_id",
                            s3_secret_key = "your_s3_secret_key",
                            s3_region = "your_s3_region",
                            s3_bucket = "your_s3_bucket",
                            mongo_key = "your_mongo_key",
                            mongo_collection = "your_mongo_collection",
                            old_cert = NULL
)

## End(Not run)
```

---

rows_for_rules *Check which rows in the data violated the rules*

Description

Get the rows in the data that violate the specified rules.

Usage

```
rows_for_rules(data_formatted, report, broken_rules, rows)
```

Arguments

data_formatted	A formatted data frame.
report	A validation report generated by the 'validate' function.
broken_rules	A data frame with broken rules information.
rows	A vector of row indices specifying which rules from the suite of rules with errors to check for violations.

Value

A data frame with rows in the data that violate the specified rules.

Examples

```
data("invalid_example")
data("test_rules")
# Generate a validation report
result_invalid <- validate_data(files_data = invalid_example,
                        data_names = c("methodology", "particles", "samples"),
                        file_rules = test_rules)

# Find the broken rules
broken_rules <- rules_broken(results = result_invalid$results[[1]], show_decision = TRUE)

# Get rows for the specified rules
violating_rows <- rows_for_rules(data_formatted = result_invalid$data_formatted[[1]],
                                 report = result_invalid$report[[1]],
                                 broken_rules = broken_rules,
                                 rows = 1)
```

`rules_broken` *Check which rules were broken*

Description

Filter the results of validation to show only broken rules, optionally including successful decisions.

Usage

```
rules_broken(results, show_decision)
```

Arguments

`results`	A data frame with validation results.
`show_decision`	A logical value to indicate if successful decisions should be included in the output.

Value

A data frame with the filtered results.

Examples

```
# Sample validation results data frame
sample_results <- data.frame(
  description = c("Rule 1", "Rule 2", "Rule 3"),
  status = c("error", "success", "error"),
  name = c("rule1", "rule2", "rule3"),
  expression = c("col1 > 0", "col2 <= 5", "col3 != 10"),
  stringsAsFactors = FALSE
)
```

```
# Show only broken rules
broken_rules <- rules_broken(sample_results, show_decision = FALSE)
```

---

run_app *Run any of the apps*

Description

This wrapper function starts the user interface of your app choice.

Usage

```
run_app(
  path = "system",
  log = TRUE,
  ref = "main",
  test_mode = FALSE,
  app = "validator",
  ...
)
```

Arguments

path	to store the downloaded app files; defaults to "system" pointing to system.file(package = "One4All").
log	logical; enables/disables logging to tempdir()
ref	git reference; could be a commit, tag, or branch name. Defaults to "main". Only change this in case of errors.
test_mode	logical; for internal testing only.
app	your app choice
...	arguments passed to runApp().

Details

After running this function the Validator, Microplastic Image Explorer, or Data Visualization GUI should open in a separate window or in your computer browser.

Value

This function normally does not return any value, see runGitHub().

Author(s)

Hannah Sherrod, Nick Leong, Hannah Hapich, Fabian Gomez, Win Cowger

See Also

runGitHub()

Examples

```
## Not run:
run_app(app = "validator")

## End(Not run)
```

`test_profanity` *Test for profanity in a string*

Description

This function checks if the input string contains any profane words.

Usage

```
test_profanity(x)
```

Arguments

x	A character string to check for profanity.

Value

A logical value indicating whether the input string contains no profane words.

Examples

```
test_profanity("This is a clean sentence.")
test_profanity("This sentence contains a badword.")
```

test_rules *Rules data*

Description

A dataset containing rules and their descriptions, datasets, valid examples, severity, and rules.

Format

A data frame with 6 columns:

name Name of the rule (e.g., "MethodologyID_valid")

description Description of the rule (e.g., "URL address is valid and can be found on the internet.")

dataset Dataset associated with the rule (e.g., "methodology")

valid_example A valid example of the rule (e.g., "https://www.waterboards.ca.gov/drinking_water/certlic/drinkingwater/doc

severity Severity of the rule (e.g., "error")

rule The actual rule (e.g., "check_uploadable(MethodologyID) == TRUE")

Examples

```
data("test_rules")
```

validate_data *Validate data based on specified rules*

Description

Validate data based on specified rules

Usage

```
validate_data(
  files_data,
  data_names = NULL,
  file_rules = NULL,
  zip_data = NULL
)
```

Arguments

files_data	A list of file paths for the datasets to be validated.
data_names	(Optional) A character vector of names for the datasets. If not provided, names will be extracted from the file paths.
file_rules	A file path for the rules file, either in .csv or .xlsx format.
zip_data	A file path to a zip folder for validating unstructured data.

Value

A list containing the following elements: - data_formatted: A list of data frames with the validated data. - data_names: A character vector of dataset names. - report: A list of validation report objects for each dataset. - results: A list of validation result data frames for each dataset. - rules: A list of validator objects for each dataset. - status: A character string indicating the overall validation status ("success" or "error"). - issues: A logical vector indicating if there are any issues in the validation results. - message: A data.table containing information about any issues encountered.

Examples

```
# Validate data with specified rules
data("valid_example")
data("invalid_example")
data("test_rules")

result_valid <- validate_data(files_data = valid_example,
                        data_names = c("methodology", "particles", "samples"),
                        file_rules = test_rules)

result_invalid <- validate_data(files_data = invalid_example,
                        data_names = c("methodology", "particles", "samples"),
                        file_rules = test_rules)
```

---

valid_example *Valid example data*

Description

This is a list containing three data frames as an example of valid_example.

Format

A list with 3 data frames:

data_frame_1 A data frame with 15 variables: MethodID, MatIDMethod, Equipment, Magnification, MethodComments, Protocols, Deployment, SamplingDevice, SmallestParticle, TopParticle, FilterType, FilterDiameter, FilterPoreSize, ImageFile, ImageType

data_frame_2 A data frame with 131 variables: SampleID, SampleSize, Project, Affiliation, Citation, OwnerContributor, AnalysisContributor, ReportingContributor, SiteName, Location, Compartment, SampleComments, SamplingDepth, SamplingVolume, SamplingWeight, BlankContamination, Latitude, Longitude, Matrix, CollectionStartDateTime, CollectionEndDateTime, SpatialFile, Concentration, ConcentrationUnits, StandardizedConcentration, StandardizedConcentrationUnits, Color_Transparent, Color_Blue, Color_Red, Color_Brown, Color_Green, Color_Orange, Color_White, Color_Yellow, Color_Pink, Color_Black, Color_Other, Material_PEST, Material_PE, Material_PP, Material_PA, Material_PE_PS, Material_PS, Material_CA, Material_PVC, Material_ER, Material_PAM, Material_PET, Material_PlasticAdditive, Material_PBT, Material_PU, Material_PET_PEST, Material_PAN, Material_Silicone, Material_Acrylic,

Material_Vinyl, Material_Vinyon, Material_Other, Material_PA_ER, Material_PTT, Material_PE_PP, Material_PPS, Material_Rayon, Material_PAA, Material_PMPS, Material_PI, Material_Olefin, Material_Styrene_Butadiene, Material_PBA, Material_PMMA, Material_Cellophane, Material_SAN, Material_PC, Material_PDMS, Material_PLA, Material_PTFE, Material_SBR, Material_PET_Olefin, Material_PES, Material_ABS, Material_LDPE, Material_PEVA, Material_AR, Material_PVA, Material_PPE, Morphology_Fragment, Morphology_Fiber, Morphology_Nurdle, Morphology_Film, Morphology_Foam, Morphology_Sphere, Morphology_Line, Morphology_Bead, Morphology_Sheet, Morphology_Film_Fragment, Morphology_Rubbery_Fragment, Size_3000um, Size_2_5mm, Size_1_5mm, Size_1_2mm, Size_0.5_1mm, Size_less_than_0.5mm, Size_500um, Size_300_500um, Size_125_300um, Size_100_500um, Size_greater_than_100um, Size_50_150um, Size_50_100um, Size_50um, Size_45_125um, Size_greater_than_25um, Size_20um_5mm, Size_20_100um, Size_20_50um, Size_10_50um, Size_10_45um, Size_10_20um, Size_greater_than_10um, Size_8_316um, Size_5_100um, Size_5_10um, Size_4_10um, Size_1.5_5um, Size_less_than_1.5um, Size_1_100um, Size_1_50um, Size_1_10um, Size_1_5um, Size_110_124nm, Size_0_20um

data_frame_3 A data frame with 19 variables: ParticleID, Amount, Color, Polymer, Shape, PhotoID, ParticleComments, PlasticType, Length, Width, Height, Units, Mass, SurfaceArea, SizeDimension, Volume, Tactile, ArrivalDate, AnalysisDate

Examples

```
data("valid_example")
```

Index

∗ **datasets**
invalid_example, 9
valid_example, 22
∗ **data**
test_rules, 21

certificate_df, 3
check_exists_in_zip, 4
check_for_malicious_files, 5
check_images, 5
check_other_hyperlinks, 6
checkLuhn, 3
create_valid_excel, 7

download_all, 8

invalid_example, 9
is.POSIXct, 9

name_data, 10

query_document_by_object_id, 10

read_data, 11
read_rules, 12
reformat_rules, 12
remote_download, 13
remote_raw_download, 14
remote_share, 15
rows_for_rules, 17
rules_broken, 18
run_app, 19
runApp, *19*
runGitHub, *19, 20*

tempdir, *19*
test_profanity, 20
test_rules, 21

valid_example, 22
validate_data, 21