Package 'rrefine'

November 15, 2022

Type Package

Title r Client for OpenRefine API

Version 2.1.0

Date 2022-11-01

Maintainer VP Nagraj <nagraj@nagraj.net>

Description 'OpenRefine' (formerly 'Google Refine') is a popular, open source data cleaning software. This package enables users to programmatically trigger data transfer between R and 'OpenRefine'. Available functionality includes project import, export and deletion.

License GPL-3

LazyData TRUE

RoxygenNote 7.1.1

Imports httr (>= 1.1.0), readr, jsonlite

Suggests knitr, rmarkdown

VignetteBuilder knitr

URL https://github.com/vpnagraj/rrefine

BugReports https://github.com/vpnagraj/rrefine/issues

Encoding UTF-8

NeedsCompilation no

Author VP Nagraj [aut, cre]

Repository CRAN

Date/Publication 2022-11-15 19:30:10 UTC

R topics documented:

`lateformeeting` *a "dirty" data set to demonstrate rrefine features*

Description

This data is a simulated collection of dates, days of the week, numbers of hours slept and indicators of whether or not the subject was on time for work. All observations appearing in this data set are fictitious, and any resemblance to actual arrival times for work is purely coincidental.

Usage

```
lateformeeting
```

Format

A data frame with 63 rows and 4 variables

- theDate date of observation in varying formats
- what.day.whas.it day of the week in varying formats
- sleephours number of hours slept
- was.i.on.time.for.work indicator of on-time arrival to work

Examples

```
head(lateformeeting)
```

lfm_clean *a "clean" version of the lateformeeting sample data set*

Description

This data is a simulated collection of dates, days of the week, numbers of hours slept and indicators of whether or not the subject was on time for work. All observations appearing in this data set are fictitious, and any resemblance to actual arrival times for work is purely coincidental.

Usage

```
lfm_clean
```

Format

A data frame with 63 rows and 4 variables

- date date of observation in POSIXct format
- dotw day of the week in consistent format
- hours.slept number of hours slept
- on.time indicator of on-time arrival to work

Examples

```
head(lfm_clean)
```

refine_add_column *Add column to OpenRefine project*

Description

This function will add a column to an existing OpenRefine project via an API query to `/command/core/apply-operations` and the `core/column-addition` operation. The value for the new column can be specified in this function either based on value of an existing column. The value can be defined using an expression written in General Refine Expression Language (GREL) syntax.

Usage

```
refine_add_column(
  new_column,
  new_column_index = 0,
  base_column = NULL,
  value,
  mode = "row-based",
  on_error = "set-to-blank",
```

```
  project.name = NULL,
  project.id = NULL,
  verbose = FALSE,
  validate = TRUE,
  ...
)
```

Arguments

new_column	Name of the new column
new_column_index	Index at which the new column should be placed in the project; default is 0 to position the new column as the first column in the project
base_column	Name of the column on which the value will be based; default is NULL, which means that the value will not be based off of a value in an existing column
value	Definition of the value for the new column; can accept a GREL expression
mode	Mode of operation; must be one of "row-based" or "record-based"; default is "row-based
on_error	Behavior if there is an error on new column creation; must be one of "set-to-blank", "keep-original", or "store-error"; default is "set-to-blank"
project.name	Name of project
project.id	Unique identifier for project
verbose	Logical specifying whether or not query result should be printed; default is FALSE
validate	Logical as to whether or not the operation should validate parameters against existing data in project; default is TRUE
...	Additional parameters to be inherited by refine_path; allows users to specify host and port arguments if the OpenRefine instance is running at a location other than http://127.0.0.1:3333

Value

Operates as a side-effect passing operations to the OpenRefine instance. However, if verbose=TRUE then the function will return an object of the class "response".

Examples

```
## Not run:
fp <- system.file("extdata", "lateformeeting.csv", package = "rrefine")
refine_upload(fp, project.name = "lfm")

refine_add_column(new_column = "date_type",
                  value = "grel:value.type()",
                  base_column = "theDate",
                  project.name = "lfm")

refine_add_column(new_column = "example_value",
```

```
                  new_column_index = 0,
                  value = "1",
                  project.name = "lfm")

## End(Not run)
```

`refine_check` *Helper function to check if* `rrefine` *can connect to OpenRefine*

Description

This function will check that `rrefine` is able to access the running OpenRefine instance. Used internally prior to upload, delete, and export operations.

Usage

```
refine_check(...)
```

Arguments

`...`	Additional parameters to be inherited by `refine_path`; allows users to specify `host` and `port` arguments if the OpenRefine instance is running at a location other than `http://127.0.0.1:3333`

Value

Error message if `rrefine` is unable to connect to OpenRefine, otherwise is invisible

---

`refine_delete` *Delete project from OpenRefine*

Description

This function allows users to delete a project in OpenRefine by name or unique project identifier. By default users are prompted to confirm deletion. The function wraps the OpenRefine API `/command/core/delete-project` query.

Usage

```
refine_delete(project.name = NULL, project.id = NULL, force = FALSE, ...)
```

Arguments

project.name	Name of project to be deleted
project.id	Unique identifier for open refine project to be deleted
force	Boolean indicating whether or not the prompt to confirm deletion should be skipped; default is FALSE
...	Additional parameters to be inherited by refine_path; allows users to specify host and port arguments if the OpenRefine instance is running at a location other than http://127.0.0.1:3333

Value

Operates as a side-effect to delete the project. Issues a message that the project has been deleted.

References

https://docs.openrefine.org/technical-reference/openrefine-api#delete-project

Examples

```
## Not run:
fp <- system.file("extdata", "lateformeeting.csv", package = "rrefine")
refine_upload(fp, project.name = "lfm")
refine_delete("lfm", force = TRUE)

## End(Not run)
```

---

refine_export *Export data from OpenRefine*

Description

This function allows users to pull data from a running OpenRefine instance into R. Users can specify project by name or unique identifier. The function wraps the OpenRefine API query to /command/core/export-rows and currently only supports export of data in tabular format.

Usage

```
refine_export(
  project.name = NULL,
  project.id = NULL,
  format = "csv",
  col.names = TRUE,
  encoding = "UTF-8",
  col_types = NULL,
  ...
)
```

Arguments

project.name	Name of project to be exported
project.id	Unique identifier for project to be exported
format	File format of project to be exported; note that the only current supported options are 'csv' or 'tsv'
col.names	Logical indicator for whether column names should be included; default is TRUE
encoding	Character encoding for exported data; default is UTF-8
col_types	One of NULL, a cols() specification, or a string; default is NULL. Used by read_csv to specify column types.
...	Additional parameters to be inherited by refine_path; allows users to specify host and port arguments if the OpenRefine instance is running at a location other than http://127.0.0.1:3333

Value

A tibble that has been parsed and read into memory using read_csv. If col.names=TRUE then the tibble will have column headers.

References

https://docs.openrefine.org/technical-reference/openrefine-api#export-rows

Examples

```
## Not run:
fp <- system.file("extdata", "lateformeeting.csv", package = "rrefine")
refine_upload(fp, project.name = "lfm")
refine_export("lfm", format = "csv")

## End(Not run)
```

---

refine_id *Helper function to get OpenRefine project.id by project.name*

Description

For functions that allow either a project name or id to be passed, this function is used internally to resolve the project id from name if necessary. It also validates that values passed to the 'project.id" argument match an existing project id in the running OpenRefine instance.

Usage

```
refine_id(project.name, project.id, ...)
```

Arguments

project.name	Name of project
project.id	Unique identifier for project
...	Additional parameters to be inherited by refine_path; allows users to specify host and port arguments if the OpenRefine instance is running at a location other than http://127.0.0.1:3333

Value

Unique id of project

---

refine_metadata *Get all project metadata from OpenRefine*

Description

This function is included internally to help retrieve metadata from the running OpenRefine instance. The query uses the OpenRefine API /command/core/get-all-project-metadata endpoint.

Usage

```
refine_metadata(...)
```

Arguments

...	Additional parameters to be inherited by refine_path; allows users to specify host and port arguments if the OpenRefine instance is running at a location other than http://127.0.0.1:3333

Value

Parsed list object with all project metadata including identifiers, names, dates of creation and modification, tags and more.

References

https://docs.openrefine.org/technical-reference/openrefine-api#get-all-projects-metadata

Examples

```
## Not run:
refine_metadata()

## End(Not run)
```

refine_move_column *Move a column in OpenRefine project*

Description

This function allows users to move an existing column in an OpenRefine project via an API query to `/command/core/apply-operations` and the `core/column-move` operation.

Usage

```
refine_move_column(
  column,
  index = 0,
  project.name = NULL,
  project.id = NULL,
  verbose = FALSE,
  validate = TRUE,
  ...
)
```

Arguments

`column`	Name of the column to be removed
`index`	Index to which the column should be placed in the project; default is `0` to position the new column as the first column in the project
`project.name`	Name of project
`project.id`	Unique identifier for project
`verbose`	Logical specifying whether or not query result should be printed; default is `FALSE`
`validate`	Logical as to whether or not the operation should validate parameters against existing data in project; default is `TRUE`
`...`	Additional parameters to be inherited by `refine_path`; allows users to specify `host` and `port` arguments if the OpenRefine instance is running at a location other than `http://127.0.0.1:3333`

Value

Operates as a side-effect passing operations to the OpenRefine instance. However, if `verbose=TRUE` then the function will return an object of the class "response".

Examples

```
## Not run:
fp <- system.file("extdata", "lateformeeting.csv", package = "rrefine")
refine_upload(fp, project.name = "lfm")
```

```
refine_move_column("sleephours", index = 0, project.name = "lfm")

## End(Not run)
```

`refine_operations` *Apply operations to OpenRefine project*

Description

This function allows users to pass arbitrary operations to an OpenRefine project via an API query to `/command/core/apply-operations`. The operations to perform must be formatted as valid `JSON` and passed to this function as a `list` object.

Usage

```
refine_operations(
  project.name = NULL,
  project.id = NULL,
  verbose = FALSE,
  operations,
  ...
)
```

Arguments

`project.name`	Name of project
`project.id`	Unique identifier for project
`verbose`	Logical specifying whether or not query result should be printed; default is `FALSE`
`operations`	List of operations to perform
`...`	Additional parameters to be inherited by `refine_path`; allows users to specify `host` and `port` arguments if the OpenRefine instance is running at a location other than `http://127.0.0.1:3333`

Value

Operates as a side-effect passing operations to the OpenRefine instance. However, if `verbose=TRUE` then the function will return an object of the class "response".

References

https://docs.openrefine.org/technical-reference/openrefine-api#apply-operations

Examples

```
## Not run:
fp <- system.file("extdata", "lateformeeting.csv", package = "rrefine")
refine_upload(fp, project.name = "lfm")

ops <-
   list(
       op = "core/text-transform",
       engineConfig = list(mode = "row-based", facets = list()),
       columnName = "was i on time for work",
       expression = "value.toUppercase()",
       onError = "set-to-blank")

refine_operations(project.name = "lfm", operations = list(ops), verbose = TRUE)

## End(Not run)
```

`refine_path` *Helper function to configure and call path to OpenRefine*

Description

This function is a helper that is used throughout `rrefine` to construct the path to the OpenRefine instance. By default this points to the localhost (`http://127.0.0.1:3333`).

Usage

```
refine_path(host = "http://127.0.0.1", port = "3333")
```

Arguments

`host`	Host for running OpenRefine instance; default is `http://127.0.0.1`
`port`	Port number for running OpenRefine instance; default is `3333`

Value

Character vector with path to running OpenRefine instance

`refine_project_summary` *Get project summary data*

Description

This function retrieves high-level project summary data (such as id, name, date created, date modified, description, and row count) from all projects in the OpenRefine instance. Internally this function uses `refine_metadata` to pull information from project metadata.

Usage

```
refine_project_summary(...)
```

Arguments

`...`	Additional parameters to be inherited by `refine_path`; allows users to specify `host` and `port` arguments if the OpenRefine instance is running at a location other than `http://127.0.0.1:3333`

Value

A `data.frame` with observations containting high-level summary metadata for all projects in the OpenRefine instance. Columns include: project id ("id"), project name ("name"), project description ("description"), count of number of project rows ("rowCount"), date created ("created"), and date modified ("modified").

References

https://docs.openrefine.org/technical-reference/openrefine-api#get-all-projects-metadata

Examples

```
## Not run:
refine_project_summary()

## End(Not run)
```

refine_query *Helper function to build OpenRefine API query*

Description

Starting with the path to the running instance, this function will add a query command and (optionally) a CSFR token with `refine_token`

Usage

```
refine_query(query, use_token = TRUE, ...)
```

Arguments

`query`	Character vector specifying the API endpoint to query
`use_token`	Boolean indicating whether or not the query string should include a CSRF Token (see `refine_token`; default is `TRUE`
`...`	Additional parameters to be inherited by `refine_path`; allows users to specify `host` and `port` arguments if the OpenRefine instance is running at a location other than `http://127.0.0.1:3333`

Value

Character vector with query based on parameter entered

refine_remove_column *Remove column from OpenRefine project*

Description

This function will remove a column from an existing OpenRefine project via an API query to `/command/core/apply-operations` and the `core/column-removal` operation.

Usage

```
refine_remove_column(
  column,
  project.name = NULL,
  project.id = NULL,
  verbose = FALSE,
  validate = TRUE,
  ...
)
```

Arguments

column	Name of the column to be removed
project.name	Name of project
project.id	Unique identifier for project
verbose	Logical specifying whether or not query result should be printed; default is FALSE
validate	Logical as to whether or not the operation should validate parameters against existing data in project; default is TRUE
...	Additional parameters to be inherited by refine_path; allows users to specify host and port arguments if the OpenRefine instance is running at a location other than http://127.0.0.1:3333

Value

Operates as a side-effect passing operations to the OpenRefine instance. However, if verbose=TRUE then the function will return an object of the class "response".

Examples

```
## Not run:
fp <- system.file("extdata", "lateformeeting.csv", package = "rrefine")
refine_upload(fp, project.name = "lfm")

refine_remove_column(column = "theDate", project.name = "lfm")

## End(Not run)
```

---

refine_rename_column *Rename a column in OpenRefine project*

Description

This function allows users to rename an existing column in an OpenRefine project via an API query to /command/core/apply-operations and the core/column-rename operation.

Usage

```
refine_rename_column(
  original_name,
  new_name,
  project.name = NULL,
  project.id = NULL,
  verbose = FALSE,
  validate = TRUE,
  ...
)
```

Arguments

original_name	Original name for the column
new_name	New name for the column
project.name	Name of project
project.id	Unique identifier for project
verbose	Logical specifying whether or not query result should be printed; default is FALSE
validate	Logical as to whether or not the operation should validate parameters against existing data in project; default is TRUE
...	Additional parameters to be inherited by refine_path; allows users to specify host and port arguments if the OpenRefine instance is running at a location other than http://127.0.0.1:3333

Value

Operates as a side-effect passing operations to the OpenRefine instance. However, if verbose=TRUE then the function will return an object of the class "response".

Examples

```
## Not run:
fp <- system.file("extdata", "lateformeeting.csv", package = "rrefine")
refine_upload(fp, project.name = "lfm")
refine_rename_column("what day whas it", "what_day_was_it", project.name = "lfm")

## End(Not run)
```

---

refine_token *Helper function to retrieve CSFR token*

Description

Helper function to retrieve CSFR token

Usage

```
refine_token(...)
```

Arguments

...	Additional parameters to be inherited by refine_path; allows users to specify host and port arguments if the OpenRefine instance is running at a location other than http://127.0.0.1:3333

Value

Character vector with OpenRefine CSFR token

`refine_upload` *Upload a file to OpenRefine*

Description

This function attempts to upload contents of a file and create a new project in OpenRefine. Users can optionally navigate directly to the running instance to interact with the project. The function wraps the OpenRefine API `/command/core/create-project-from-upload` query.

Usage

```
refine_upload(file, project.name = NULL, open.browser = FALSE, ...)
```

Arguments

`file`	Path to file to upload; upload format is inferred from the file extension, and currently only ".csv" and ".tsv" files are allowed.
`project.name`	Optional parameter to specify name of the project to be created upon upload; default is `NULL` and project will be named 'Untitled' in OpenRefine
`open.browser`	Boolean for whether or not the browser should open on successful upload; default is `FALSE`
`...`	Additional parameters to be inherited by `refine_path`; allows users to specify `host` and `port` arguments if the OpenRefine instance is running at a location other than `http://127.0.0.1:3333`

Value

Operates as a side-effect, either opening a browser and pointing to the OpenRefine instance (if `open.browser=TRUE`) or issuing a message.

References

https://docs.openrefine.org/technical-reference/openrefine-api#create-project

Examples

```
## Not run:
fp <- system.file("extdata", "lateformeeting.csv", package = "rrefine")
refine_upload(fp, project.name = "lfm")
write.table(x = mtcars, file = "mtcars.tsv", sep = "\t")
refine_upload(file = "mtcars.tsv", project.name = "mtcars")

## End(Not run)
```

transform *Text transformation for OpenRefine project*

Description

The text transform functions allow users to pass arbitrary text transformations to a column in an existing OpenRefine project via an API query to `/command/core/apply-operations` and the `core/text-transform` operation. Besides the generic `refine_transform()`, the package includes a series of transform functions that apply commonly used text operations. For more information on these functions see 'Details'.

Usage

```
refine_transform(
  column_name,
  expression,
  mode = "row-based",
  on_error = "set-to-blank",
  project.name = NULL,
  project.id = NULL,
  verbose = FALSE,
  validate = TRUE,
  ...
)

refine_to_lower(
  column_name,
  mode = "row-based",
  on_error = "set-to-blank",
  project.name = NULL,
  project.id = NULL,
  verbose = FALSE,
  validate = TRUE,
  ...
)

refine_to_upper(
  column_name,
  mode = "row-based",
  on_error = "set-to-blank",
  project.name = NULL,
  project.id = NULL,
  verbose = FALSE,
  validate = TRUE,
  ...
)
```

```
  refine_to_title(
    column_name,
    mode = "row-based",
    on_error = "set-to-blank",
    project.name = NULL,
    project.id = NULL,
    verbose = FALSE,
    validate = TRUE,
    ...
  )

  refine_to_null(
    column_name,
    mode = "row-based",
    on_error = "set-to-blank",
    project.name = NULL,
    project.id = NULL,
    verbose = FALSE,
    validate = TRUE,
    ...
  )

  refine_to_empty(
    column_name,
    mode = "row-based",
    on_error = "set-to-blank",
    project.name = NULL,
    project.id = NULL,
    verbose = FALSE,
    validate = TRUE,
    ...
  )

  refine_to_text(
    column_name,
    mode = "row-based",
    on_error = "set-to-blank",
    project.name = NULL,
    project.id = NULL,
    verbose = FALSE,
    validate = TRUE,
    ...
  )

  refine_to_number(
    column_name,
    mode = "row-based",
    on_error = "set-to-blank",
```

```
  project.name = NULL,
  project.id = NULL,
  verbose = FALSE,
  validate = TRUE,
  ...
)

refine_to_date(
  column_name,
  mode = "row-based",
  on_error = "set-to-blank",
  project.name = NULL,
  project.id = NULL,
  verbose = FALSE,
  validate = TRUE,
  ...
)

refine_trim_whitespace(
  column_name,
  mode = "row-based",
  on_error = "set-to-blank",
  project.name = NULL,
  project.id = NULL,
  verbose = FALSE,
  validate = TRUE,
  ...
)

refine_collapse_whitespace(
  column_name,
  mode = "row-based",
  on_error = "set-to-blank",
  project.name = NULL,
  project.id = NULL,
  verbose = FALSE,
  validate = TRUE,
  ...
)

refine_unescape_html(
  column_name,
  mode = "row-based",
  on_error = "set-to-blank",
  project.name = NULL,
  project.id = NULL,
  verbose = FALSE,
  validate = TRUE,
```

```
  ...
)
```

Arguments

column_name	Name of the column on which text transformation should be performed
expression	Expression defining the text transformation to be performed
mode	Mode of operation; must be one of "row-based" or "record-based"; default is "row-based"
on_error	Behavior if there is an error on new column creation; must be one of "set-to-blank", "keep-original", or "store-error"; default is "set-to-blank"
project.name	Name of project
project.id	Unique identifier for project
verbose	Logical specifying whether or not query result should be printed; default is FALSE
validate	Logical as to whether or not the operation should validate parameters against existing data in project; default is TRUE
...	Additional parameters to be inherited by refine_path; allows users to specify host and port arguments if the OpenRefine instance is running at a location other than http://127.0.0.1:3333

Details

The refine_transform() function allows the user to pass arbitrary text transformations to a given column in an OpenRefine project. The package includes a set of functions that wrap refine_transform() to execute common transformations:

- refine_to_lower(): Coerce text to lowercase
- refine_to_upper(): Coerce text to uppercase
- refine_to_title(): Coerce text to title case
- refine_to_null(): Set values to NULL
- refine_to_empty(): Set text values to empty string ("")
- refine_to_text(): Coerce value to string
- refine_to_number(): Coerce value to numeric
- refine_to_date(): Coerce value to date
- refine_trim_whitespace(): Remove leading and trailing whitespaces
- refine_collapse_whitespace(): Collapse consecutive whitespaces to single whitespace
- refine_unescape_html(): Unescape HTML in string

Value

Operates as a side-effect passing operations to the OpenRefine instance. However, if verbose=TRUE then the function will return an object of the class "response".

Examples

```
## Not run:
fp <- system.file("extdata", "lateformeeting.csv", package = "rrefine")
refine_upload(fp, project.name = "lfm")

refine_add_column(new_column = "dotw",
                  base_column = "what day whas it",
                  value = "grel:value",
                  project.name = "lfm")

refine_export("lfm")$dotw
refine_to_lower("dotw", project.name = "lfm")
refine_export("lfm")$dotw
refine_to_upper("dotw", project.name = "lfm")
refine_export("lfm")$dotw
refine_to_title("dotw", project.name = "lfm")
refine_export("lfm")$dotw
refine_to_null("dotw", project.name = "lfm")
refine_export("lfm")$dotw
refine_remove_column("dotw", project.name = "lfm")

refine_add_column(new_column = "date",
                  base_column = "theDate",
                  value = "grel:value",
                  project.name = "lfm")

refine_export("lfm")$date
refine_to_date("date", project.name = "lfm")
refine_export("lfm")$date
refine_remove_column("date", project.name = "lfm")


## End(Not run)
```

Index