Package ‘magmaR’

May 8, 2026

Type Package

Title R-Client for Interacting with the 'UCSF Data Library'

Version 1.0.4

Description A client for interacting with 'magma', the data warehouse of the 'UCSF Data Library'. 'magmaR' includes functions for querying and downloading data from 'magma', in order to enable working with such data in R, as well as for uploading local data to 'magma'.

Depends R (>= 4.0)

Imports crul, jsonlite, utils

Suggests dittoSeq, BiocStyle, vcr, webmockr, testthat, knitr, rmarkdown

License GPL-2

Encoding UTF-8

RoxygenNote 7.3.2

VignetteBuilder knitr

NeedsCompilation no

Author Daniel Bunis [aut, cre]

Maintainer Daniel Bunis <daniel.bunis@ucsf.edu>

Repository CRAN

Date/Publication 2025-05-23 15:02:02 UTC

Contents

`authentication-and-environments`
Starting out and working with different Etna environments

Description

Starting out and working with different Etna environments

Authorization via 'token's

Access to magma via magmaR is authenticated via 'token's which users can obtain from Janus. A valid token must be provided to magmaR before any calls to magma can be successfully performed. To provide this token, users should obtain their token from Janus, then provide it to magmaR functions using the `magmaRset` function. See the function's own documentation and other functions' examples for further details and usage code.

Non-production Environments

The code base relies on 2 different ecosystems of all of its components for purposes of "development" of new tools and features, and "production", the release version which most users see. We won't get into the details too much more here, but users with access to the development environment can access it magmaR.

To do so, users should provide the `url` of their alternative version of magma to magmaR functions using the `magmaRset` function. If proxy or other curl-request settings need to be adjusted, users can provide these via the `opts` input of this same `magmaRset` function. At a minimum, you will likely want to use `opts = list(ssl_verifyhost = FALSE, ssl_verifypeer = FALSE)`.

See Also

`magmaRset`

magmaRset *Set up your magma environment and authentication*

Description

Set up your magma environment and authentication

Usage

```
magmaRset(
  token = NULL,
  url = "https://magma.ucsf.edu",
  opts = list(followlocation = FALSE)
)
```

Arguments

token	Single string. Your personal token from https://janus.ucsf.edu. When not explicitly given, you will be prompted to input it via the console.
url	Single string. The url of the production, staging, or development version of magma that you you would like to target. See authentication-and-environments for more information.
opts	A named list of curl options and the values to give them (ex: list(followlocation = FALSE, othersetting = 42)). Generally, most users can ignore this input, but it can be useful for adjusting proxy settings for particular development environment setup.

Details

This function compiles a list, from the given inputs, of the information needed by other magmaR functions to properly route and authenticate a call to magma.

Value

A list with three components: token, url, and opts.

Examples

```
if (interactive()) {

    # THE DEFAULT:
    # When run in this way, it will ask you to give your token.
    # And the resulting $url will be the standard, production, magma url.
    prod <- magmaRset()
    print(prod)

    # TARGET = staging:
    # Give the proper url.
```

```
    # Again, because we are not providing our token to the call, it will ask.
    stage <- magmaRset(url = "https://magma-stage.ucsf.edu")
    print(stage)

    # We can also give additional curl options to the 'opts' input:
    prod_opts <- magmaRset(token = prod$token,
        opts = list(proxyport = 1234))
    print(prod_opts)

    # Now we can retrieve data with...
    retrieve(
        target = prod,
        projectName = "example",
        modelName = "rna_seq",
        recordNames = "all",
        attributeNames = "all",
        filter = "")
}
```

---

query *Search-like function that can obtain linked data from distinct models.*

Description

Analogous to the '/query' function of magma.

Usage

```
query(target, projectName, queryTerms = list(), format = c("list", "df"), ...)
```

Arguments

target	A list, which can be created using magmaRset, containing your authorization 'token' (a string), a 'url' of magma to target (a string), and optional 'opts' for specifying additions parameters for curl requests (a named list).
projectName	Single string. The name of the project you would like to interact with. For options, see retrieveProjects.
queryTerms	A list of strings where list elements are query predicates and verbs. See https://mountetna.github.io/magma.html#query for details.
format	Either "list" or "df" (=dataframe). This sets the desired output format. The list option is the more raw form.
...	Additional parameters passed along to the internal '.retrieve()', '.query()', or '.update()' functions, for troubleshooting or advanced-user purposes only: • request.only (Logical) & json.params.only (Logical) which 1) stop the function before its main curl request to magma and 2) returns the values that would have been sent to magma in either of two formats. • verbose (Logical) sets whether to report the status of the curl request after it is performed.

Details

This function initially mimics the activity of the magma's /query functionality, which is documented here https://mountetna.github.io/magma.html#query.

Afterwards, the json list output of magma/query is converted into an R list, and then the `format` input determines whether it should be wrangled further:

- `format = "list"`, default: R list output directly.
- `format = "df"`: R list converted into a dataframe where data comes from the list$answer and column names come from the list$format

Value

A list, default, if `format == "list"`,

OR A dataframe conversion if `format = "df"`

See Also

https://mountetna.github.io/magma.html#query for documentation of the underlying magma/query function.

`retrieveProjects` for exploring options for the `projectName` input.

`retrieveModels`, `retrieveIds`, and `retrieveAttributes` and `retrieveTemplate` for exploring the project structure and determining `queryTerm` options.

Examples

```
if (interactive()) {
    # First, we use magmaRset to create an object which will tell other magmaR
    #  functions our authentication token (as well as some other optional bits).
    # When run in this way, it will ask you to give your token.
    magma <- magmaRset()

    ### To obtain the 'group' attribute, from the subject-model, that are
    #   associated with records of the rna_seq-model:

    # "Raw" output of query:
    query_list <- query(
        target = magma,
        projectName = "example",
        queryTerms =
            list('rna_seq',
                 '::all',
                 'biospecimen',
                 'subject',
                 'group'))
    print(query_list)

    # Or instead re-formatted to a dataframe, which may be easier for
    #   downstream applications in R:
    query_df <- query(
```

```
        target = magma,
        projectName = "example",
        queryTerms =
            list('rna_seq',
                 '::all',
                 'biospecimen',
                 'subject',
                 'group'),
        format = 'df')
    print(query_df)
}
```

`retrieve` *Download data from magma as a tsv, and convert to a data.frame*

Description

Analogous to the '/retrieve' function of magma, with `format = "tsv"`

Usage

```
retrieve(
  target,
  projectName,
  modelName,
  recordNames = "all",
  attributeNames = "all",
  filter = "",
  page = NULL,
  pageSize = 10,
  showDisconnected = FALSE,
  ...
)
```

Arguments

`target`	A list, which can be created using `magmaRset`, containing your authorization 'token' (a string), a 'url' of magma to target (a string), and optional 'opts' for specifying additions parameters for curl requests (a named list).
`projectName`	Single string. The name of the project you would like to interact with. For options, see `retrieveProjects`.
`modelName`	Single string. The name of the subset data structure within the project, which are referred to as 'model's in magma, to interact with. For options, see `retrieveModels` or https://timur.ucsf.edu/<projectName>/map.
`recordNames`	Single string or string vector indicating which particular sample/tube/etc. records to target. Options are "all" or any combination of individual record names. To retrieve individual options, see `retrieveIds`.

`attributeNames` Single string or string vector indicating which features of the data to target. Options are "all" or any combination of individual attribute names. To retrieve individual options, see `retrieveAttributes`.

`filter` String. Potential filter(s) of the data. Example: "<targetAttributeName>~GYN" to filter to records where <targetAttributeName> contains "GYN".

Refer to https://mountetna.github.io/magma.html#retrieve for more details about options and format.

`page` Integer. For retrieving just a portion of the data, sets which slice to get.

`pageSize` Integer. For retrieving just a portion of the data, sets slice/page size, which is equivalent to the a number of rows.

`showDisconnected`

Boolean. Set to true to access "disconnected" records, which are records that are missing an (upstream) parent linkage, and so do not connect up with the project's top-level project record. Generally, disconnected records are ones that were deemed low quality in some way, thus were purposefully disconnected from the rest of the dataset. But sometimes a record might just be disconnected because an upload went awry.

`...` Additional parameters passed along to the internal '.retrieve()', '.query()', or '.update()' functions, for troubleshooting or advanced-user purposes only:

- `request.only` (Logical) & `json.params.only` (Logical) which 1) stop the function before its main curl request to magma and 2) returns the values that would have been sent to magma in either of two formats.
- `verbose` (Logical) sets whether to report the status of the curl request after it is performed.

Details

This function makes a curl get request to magma/retrieve, with properly reformatted versions of user inputs, plus `format = "tsv"`. Then, it converts the tsv-string output into a dataframe.

Note: When `format = "tsv"`, magma/retrieve returns just an identifier for matrix-type attributes. To retrieve underlying data for such attributes, use the specialized `retrieveMatrix` function.

Value

A dataframe

See Also

`retrieveMatrix` for retrieving attributes of type matrix.

`retrieveJSON` for similar functionality to `retrieve`, but where the call to magma/retrieve is made with `format = "json"` and the output is a list. This output often contains more information, and can retrieve data for attribute types of type matrix, which are not returned by the current function. But in most cases, the data returned by `retrieve` and `retrieveMatrix` will suffice.

https://mountetna.github.io/magma.html#retrieve for documentation of the underlying magma/retrieve function.

Examples

```
if (interactive()) {
    # First, we use magmaRset to create an object which will tell other magmaR
    #  functions our authentication token (as well as some other optional bits).
    # When run in this way, it will ask you to give your token.
    magma <- magmaRset()

    # Now we can retrieve data with...
    retrieve(
        target = magma,
        projectName = "example",
        modelName = "rna_seq",
        recordNames = "all",
        attributeNames = "all",
        filter = "")
}
```

`retrieveJSON` *Download data from magma as a json, and convert to a list*

Description

Analogous to the '/retrieve' function of magma, with `format = "json"`

Usage

```
retrieveJSON(
  target,
  projectName,
  modelName,
  recordNames = "all",
  attributeNames = "all",
  filter = "",
  page = NULL,
  pageSize = 10,
  showDisconnected = FALSE,
  hideTemplate = FALSE,
  ...
)
```

Arguments

`target`	A list, which can be created using `magmaRset`, containing your authorization 'token' (a string), a 'url' of magma to target (a string), and optional 'opts' for specifying additions parameters for curl requests (a named list).
`projectName`	Single string. The name of the project you would like to interact with. For options, see `retrieveProjects`.

`modelName` Single string. The name of the subset data structure within the project, which are referred to as 'model's in magma, to interact with. For options, see `retrieveModels` or https://timur.ucsf.edu/<projectName>/map.

`recordNames` Single string or string vector indicating which particular sample/tube/etc. records to target. Options are "all" or any combination of individual record names. To retrieve individual options, see `retrieveIds`.

`attributeNames` Single string or string vector indicating which features of the data to target. Options are "all" or any combination of individual attribute names. To retrieve individual options, see `retrieveAttributes`.

`filter` String. Potential filter(s) of the data. Example: "<targetAttributeName>~GYN" to filter to records where <targetAttributeName> contains "GYN".

Refer to `https://mountetna.github.io/magma.html#retrieve` for more details about options and format.

`page` Integer. For retrieving just a portion of the data, sets which slice to get.

`pageSize` Integer. For retrieving just a portion of the data, sets slice/page size, which is equivalent to the a number of rows.

`showDisconnected`

Boolean. Set to true to access "disconnected" records, which are records that are missing an (upstream) parent linkage, and so do not connect up with the project's top-level project record. Generally, disconnected records are ones that were deemed low quality in some way, thus were purposefully disconnected from the rest of the dataset. But sometimes a record might just be disconnected because an upload went awry.

`hideTemplate` Logical. Allows to leave out the project template from the return. Often this does not matter much, but the template can be bulky.

`...` Additional parameters passed along to the internal '.retrieve()', '.query()', or '.update()' functions, for troubleshooting or advanced-user purposes only:

- `request.only` (Logical) & `json.params.only` (Logical) which 1) stop the function before its main curl request to magma and 2) returns the values that would have been sent to magma in either of two formats.
- `verbose` (Logical) sets whether to report the status of the curl request after it is performed.

Details

This function makes a call to magma/retrieve with `format = "json"`. Then, it converts the json output into a list which is more compatible with R.

Value

A list

See Also

`retrieve` for similar functionality, but where the call to magma/retrieve will be made with `format = "tsv"` and the output is a dataframe.

retrieveMatrix for matrix data-targeted utilization of this current retreiveJSON function, followed by automated restructuring of the return into a matrix format.

https://mountetna.github.io/magma.html#retrieve for documentation of the underlying magma/retrieve function.

Examples

```
if (interactive()) {
    # First, we use magmaRset to create an object which will tell other magmaR
    #  functions our authentication token (as well as some other optional bits).
    # When run in this way, it will ask you to give your token.
    magma <- magmaRset()

    # Now we can retrieve data as json (->list) with...
    json_out <- retrieveJSON(
        target = magma,
        projectName = "example",
        modelName = "rna_seq",
        recordNames = "all",
        attributeNames = "all",
        filter = "")
    # The return will be a nested list with data in a 'documents' element and
    #  some extra information about each attribute in a 'template' element.
    str(json_out, max.level = 4)

    # Often, the 'template' part is bulky but not needed, so its retrieval may
    #  be turned off by giving hideTemplaate = TRUE'
    json_out <- retrieveJSON(
        target = magma,
        projectName = "example",
        modelName = "rna_seq",
        recordNames = "all",
        attributeNames = "all",
        filter = "",
        hideTemplate = TRUE)
    str(json_out, max.level = 4)
}
```

---

retrieveMatrix *Download data from magma that is stored as a matrix*

Description

Download data from magma that is stored as a matrix

Usage

```
retrieveMatrix(
  target,
  projectName,
  modelName,
  recordNames = "all",
  attributeNames,
  filter = "",
  page = NULL,
  pageSize = 10,
  ...
)
```

Arguments

target	A list, which can be created using magmaRset, containing your authorization 'token' (a string), a 'url' of magma to target (a string), and optional 'opts' for specifying additions parameters for curl requests (a named list).
projectName	Single string. The name of the project you would like to interact with. For options, see retrieveProjects.
modelName	Single string. The name of the subset data structure within the project, which are referred to as 'model's in magma, to interact with. For options, see retrieveModels or https://timur.ucsf.edu/<projectName>/map.
recordNames	Single string or string vector indicating which particular sample/tube/etc. records to target. Options are "all" or any combination of individual record names. To retrieve individual options, see retrieveIds.
attributeNames	Single string or string vector indicating which features of the data to target. Options are "all" or any combination of individual attribute names. To retrieve individual options, see retrieveAttributes.
filter	String. Potential filter(s) of the data. Example: "<targetAttributeName>~GYN" to filter to records where <targetAttributeName> contains "GYN". Refer to https://mountetna.github.io/magma.html#retrieve for more details about options and format.
page	Integer. For retrieving just a portion of the data, sets which slice to get.
pageSize	Integer. For retrieving just a portion of the data, sets slice/page size, which is equivalent to the a number of rows.
...	Additional parameters passed along to the internal '.retrieve()', '.query()', or '.update()' functions, for troubleshooting or advanced-user purposes only: • request.only (Logical) & json.params.only (Logical) which 1) stop the function before its main curl request to magma and 2) returns the values that would have been sent to magma in either of two formats. • verbose (Logical) sets whether to report the status of the curl request after it is performed.

Value

a matrix

Examples

```
if (interactive()) {
    # First, we use magmaRset to create an object which will tell other magmaR
    #  functions our authentication token (as well as some other optional bits).
    # When run in this way, it will ask you to give your token.
    magma <- magmaRset()

    retrieveMatrix(
        target = magma,
        projectName = "example",
        modelName = "rna_seq",
        recordNames = "all",
        attributeNames = "gene_counts")
}
```

`retrieveMetadata` *Download data from magma of one model, but transformed into the shape of a different model's records.*

Description

Retrieve data from one model ("meta") transformed into the shape of linked records of a different model ("target"). For example, one could get subject-level information for an RNAseq counts matrix with this function. The output would contain columns of subject-level attributes, and rows that are the RNAseq-model records.

Usage

```
retrieveMetadata(
  target,
  projectName,
  meta_modelName,
  meta_attributeNames = "all",
  target_modelName,
  target_recordNames = "all",
  template = NULL,
  ...
)
```

Arguments

`target`	A list, which can be created using `magmaRset`, containing your authorization 'token' (a string), a 'url' of magma to target (a string), and optional 'opts' for specifying additions parameters for curl requests (a named list).
`projectName`	Single string. The name of the project you would like to interact with. For options, see `retrieveProjects`.

`meta_modelName`, `meta_attributeNames`
: Strings which indicate the "meta"data to retrieve. They work the same as inputs of other functions without the `meta_` portion.

`target_modelName`, `target_recordNames`
: Strings which indicate the "target" data that meta-data is desired to be reshaped into. They work the same as inputs of other functions without the `target_` portion, and these inputs ultimately set which records of "meta" model data to actually obtain.

`template`
: For internal use in minimizing excess http requests to magma, NULL or the return of `retrieveTemplate(target, projectName)`.

`...`
: Additional parameters passed along to the internal '.retrieve()', '.query()', or '.update()' functions, for troubleshooting or advanced-user purposes only:

  - `request.only` (Logical) & `json.params.only` (Logical) which 1) stop the function before its main curl request to magma and 2) returns the values that would have been sent to magma in either of two formats.
  - `verbose` (Logical) sets whether to report the status of the curl request after it is performed.

Details

This function retrieves data from one model (the "meta" model) transformed so that rows of the returned dataframe correspond to records of a different model (the "target" model).

Internally, it first determines the path, through child-parent model linkages, to navigate from the meta_model to the target_model.

Then, it performs calls to `query` in order to retrieve identifier linkage along that path. The identifier linkage is turned into a dataframe of identifier traces.

Next, it performs a call to `retrieve` to obtain the wanted metadata as a dataframe.

(If linkage paths would create any 1:many mappings of target data records to metadata records, data of extra records are shifted "rightwards" into columns appended with "_#" in their names. This is a reliable, though imperfect, method so we hope to implement alternatives in the future.)

Finally, the dataframe of linkage path identifiers is merged with the metadata dataframe, reshaping the metadata to properly have one row per requested `target_recordName`.

Value

A dataframe with rows = `target_recordNames` and columns = model identifiers and either `meta_attributeNames` or repeats of `meta_attributeNames_#` when there are 1:many mappings of target data records to metadata records.

See Also

`retrieve` and `retrieveMatrix` which will likely be useful for retrieving associated "target" data.

Examples

```
if (interactive()) {
    # First, we use magmaRset to create an object which will tell other magmaR
    #  functions our authentication token (as well as some other optional bits).
    # When run in this way, it will ask you to give your token.
    magma <- magmaRset()

    # Running like this will ask for input of your janus token one time.
    retrieveMetadata(
        target = magma,
        projectName = "example",
        meta_modelName = "subject",
        meta_attributeNames = "group",
        target_modelName = "rna_seq",
        target_recordNames = "all")
}
```

`retrieveProjects` *Helper function that retrieves all the projectName options which a user has access to, from janus.*

Description

Helper function that retrieves all the projectName options which a user has access to, from janus.

Usage

```
retrieveProjects(target, all_cols = FALSE, verbose = FALSE)
```

Arguments

`target`	A list, which can be created using `magmaRset`, containing your authorization 'token' (a string), a 'url' of magma to target (a string), and optional 'opts' for specifying additions parameters for curl requests (a named list).
`all_cols`	Logical. Sets whether to report back the entire table instead of the default behavior where the column 'cc_text' is removed for brevity.
`verbose`	Logical. Sets whether to report the status of the '/api/user/projects' curl request sent to janus.

Details

This function takes in the user's `target` containing their authorization token, and a url targeting either magma or janus. It then converts the given url to target janus, and makes a curl request to <janus-url>/api/user/projects in order to return which projects a user can access.

Value

A data.frame where elements of the 'project_name' column reflect what can be given to `projectName` inputs of other magmaR functions.

Examples

```
if (interactive()) {
    retrieveProjects(target = magmaRset())
}
```

`retrieve_SpecialCases` *Helper functions that utilize special cases of magma /retrieve*

Description

Helper functions that utilize special cases of magma /retrieve

Usage

```
retrieveTemplate(target, projectName, ...)

retrieveModels(target, projectName, template = NULL, ...)

retrieveIds(target, projectName, modelName, ...)

retrieveAttributes(target, projectName, modelName, ...)

retrieveParentName(target, projectName, modelName, template = NULL, ...)
```

Arguments

`target`	A list, which can be created using `magmaRset`, containing your authorization 'token' (a string), a 'url' of magma to target (a string), and optional 'opts' for specifying additions parameters for curl requests (a named list).
`projectName`	Single string. The name of the project you would like to interact with. For options, see `retrieveProjects`.
`...`	Additional parameters passed along to the internal '.retrieve()', '.query()', or '.update()' functions, for troubleshooting or advanced-user purposes only: • `request.only` (Logical) & `json.params.only` (Logical) which 1) stop the function before its main curl request to magma and 2) returns the values that would have been sent to magma in either of two formats. • `verbose` (Logical) sets whether to report the status of the curl request after it is performed.
`template`	For internal use in minimizing excess http requests to magma, NULL or the return of `retrieveTemplate(target, projectName)`.

`modelName` Single string. The name of the subset data structure within the project, which are referred to as 'model's in magma, to interact with. For options, see `retrieveModels` or https://timur.ucsf.edu/<projectName>/map.

Details

These functions aim to help users determine acceptable inputs to other magmaR functions without needing to leave R.

They make properly crafted calls to `retrieve` which target either the "template" or "identifier" special cases outlined in https://mountetna.github.io/magma.html#retrieve, followed by directly returning the output (`retrieveTemplate` and `retrieveIds`), by returning just a targeted portion of that output (`retrieveModels`), or by returning a targeted portion of a subsequent single-record call to `retrieve` (`retrieveAttributes`).

Value

retrieveTemplate = a list conversion of the project's template json.

retrieveModels = a string vector of model names

retrieveIds = a string vector of record names/identifiers.

retrieveAttributes = a string vector of attribute names.

Functions

- `retrieveTemplate()`: Retrieve the template for a given project
- `retrieveModels()`: Retrieve the modelNames for a given project
- `retrieveIds()`: Retrieve all the identifiers/recordNames for a given project-model pair.
- `retrieveAttributes()`: Retrieve all the attribute options for a given project-model pair.
- `retrieveParentName()`: Retrieve the parent modelName / attributeName for a given project-model pair.

Examples

```
if (interactive()) {
    # First, we use magmaRset to create an object which will tell other magmaR
    #  functions our authentication token (as well as some other optional bits).
    # When run in this way, it will ask you to give your token.
    magma <- magmaRset()

    template <- retrieveTemplate(
        target = magma,
        projectName = "example")
    str(template, max.level = 4)

    models <- retrieveModels(
        target = magma,
        projectName = "example")
    print(models)
```

```
    ids <- retrieveIds(
        target = magma,
        projectName = "example",
        modelName = "rna_seq")
    print(ids)

    atts <- retrieveAttributes(
        target = magma,
        projectName = "example",
        modelName = "subject")
    print(atts)
}
```

---

updateFromDF *Easier to use wrapper of* updateValues

Description

A wrapper of updateValues which takes in updates in the form of a dataframe, csv, tsv, with rows = records and columns = attributes.

Usage

```
updateFromDF(
  target,
  projectName,
  modelName,
  df,
  table.method = NULL,
  autolink = FALSE,
  dryRun = FALSE,
  separator = ",",
  show.df = TRUE,
  auto.proceed = FALSE,
  revisions.only = FALSE,
  template = NULL,
  ...
)
```

Arguments

target	A list, which can be created using magmaRset, containing your authorization 'token' (a string), a 'url' of magma to target (a string), and optional 'opts' for specifying additions parameters for curl requests (a named list).
projectName	Single string. The name of the project you would like to interact with. For options, see retrieveProjects.

`modelName` Single string. The name of the subset data structure within the project, which are referred to as 'model's in magma, to interact with. For options, see `retrieveModels` or https://timur.ucsf.edu/<projectName>/map.

`df` A dataframe, containing the data to upload to magma.

Alternatively, a String specifying the file path of a file containing such data.

See below for additional formatting details.

`table.method` "replace" or "append". Sets the methodology used for building the `revisions` to request for table-type model updates:

- "append": Add the currently defined records, attaching them to parents defined in the `df`.
- "replace": Add the currently defined records, attaching them to parents defined in the `df`, AND unlink / remove all current records, of `modelName`, attached to parent records defined in the `df`.

`autolink` Logical. FALSE by default for safety, but often you will want to set it to TRUE. Passed through to magma, this parameter controls whether the system will attempt to connect all targeted records up with the project's root. Specifically, this means the system will 1) determine parent records of all targeted records if it can, based on the project's gnomon grammar, 2) continue parent determination for those parent records, repeating this process until reaching the project's root (the project record), then 3) creates any of these records that don't currently exist, and finally 4) creates all the assumed parent-child linkages

`dryRun` Logical. FALSE by default. Passed through to magma, this parameter controls whether the system will only test whether the update is valid without making changes to the database.

`separator` String indicating the field separator to use if providing `df` as a file path. Default = `","`. Use `"\t"` for tsvs.

`show.df` Logical which sets whether the `df`-data should be printed out.

`auto.proceed` Logical. When set to TRUE, the function does not ask before proceeding forward with the 'magma/update'.

`revisions.only` Logical. For troubleshooting purposes, when set to `TRUE`, no data will be sent to magma. Instead, the list structure that would have been passed to the `revisions` input of `updateValues` is returned as output.

`template` For internal use in minimizing excess http requests to magma, NULL or the return of `retrieveTemplate(target, projectName)`.

`...` Additional parameters passed along to the internal '.retrieve()', '.query()', or '.update()' functions, for troubleshooting or advanced-user purposes only:

- `request.only` (Logical) & `json.params.only` (Logical) which 1) stop the function before its main curl request to magma and 2) returns the values that would have been sent to magma in either of two formats.
- `verbose` (Logical) sets whether to report the status of the curl request after it is performed.

Details

This function provides a simple method for updating multiple attributes of multiple magma records provided as a rectangular dataframe, or equivalent file structure. It utilizes the magma/query function, documented here https://mountetna.github.io/magma.html#update, to upload data after converting to the format required by that function.

The user-indicated `df` is read in, presented to the user for inspection, then transformed to the necessary format and passed along to `updateValues`.

The `updateValues()` function will then summarize records to be updated and allow the user to double-check this information before proceeding.

This user-prompt step can be bypassed (useful when running in a non-interactive way) by setting `auto.proceed = TRUE`, but NOTE: It is a good idea to always check carefully before proceeding, if possible. Data can be overwritten with NAs or zeros or the like, or disconnected from parent records, but improperly named records cannot be easily removed.

For "standard" models with explicit identifiers, the function targets the `df`'s row-indicated records and column-indicated attributes of the `modelName` model of `projectName` project. In such cases, the first column of `df` must contain the identifiers of the records your wish to update.

For table-type models which do not have explicit identifiers, the function creates records per each row of the given `df` with the requested values filled in for column-indicated attributes of the `modelName` model of `projectName` project, and attaches these records to the indicated parent records. In such cases, a column named as the parent model must exist in `df` to provide the parent identifiers of all requested new data. In such cases, `table.method` must also be given as either `"append"` or `"replace"`.

`df` can be provided either as a data.frame directly, or as a file path pointing to a file containing such data. If given as a file path, the `separator` input can be used to adjust for whether the file is a csv (the default, `separator = ","`), or tsv, `separator = "\t"`, or other format.

The `df` data structure when targeting 'standard' models:

- Rows = records, with the first column indicating the record identifiers.
- Columns = represent the data desired to be given for each attribute.
- Column Names (or the top row when providing a file) = attribute names. Except for the first column (ignored as this column's data are used as identifiers), all column names must be valid attribute names of the target `modelName`.

The `df` data structure when targeting table-type models:

- Rows = records, but no identifiers are needed.
- Columns = represent the data desired to be given for each attribute.
- Column Names (or the top row when providing a file) = attribute names. At least one column must be named after the parent model and must represent parent model identifiers.

Value

None directly.

The function sends data to magma, and the only outputs are information reported via the console.

Use Case. Using this function to change records' identifiers

To do so, provide a file or dataframe where 1) The first column, named something random Iits name will be ignored.), contains current identifiers; 2) Some other column, named as the attribute which is treated as the identifier for the model, contains the new identifiers

To determine the identifier attribute's name, you can use `retrieveTemplate`:

`retrieveTemplate(<target>, <projectName>)$models$<modelName>$template$identifier`.

See Also

`updateMatrix` for uploading matrix data

`updateValues` for a more direct replica of `magma/update` which is more flexible, though a bit more complicated to use.

https://mountetna.github.io/magma.html#update for documentation of the underlying `magma/update` function.

Examples

```
if (interactive()) {
    # First, we use magmaRset to create an object which will tell other magmaR
    #  functions our authentication token (as well as some other optional bits).
    # When run in this way, it will ask you to give your token.
    magma <- magmaRset()

    ### Note that you likely do not have write-permissions for the 'example'
    # project, so this code can be expected to give an authorization error,
    # yet still provides a good example of how to structure your code.

    ### Case 1: A 'standard' model which has an identifier:
    # Retrieving some example data from magma to use as our update
    df <- retrieve(
        magma, projectName = "example", modelName = "rna_seq",
        recordNames = "all",
        attributeNames = c("tube_name", "biospecimen", "cell_number")
        )
    df
    # Keys to note:
    # - the first column of this df holds the identifiers of all records we
    #     wish to target.  (The fact that this column is properly named as
    #     'tube_name' des not matter.)
    # - all subsequent columns hold the new values and are named as the
    #     attributes we wish to update.

    # To update values of the 'standard'-type "rna_seq" model
    updateFromDF(
        target = magma,
        projectName = "example",
        modelName = "rna_seq",
        df = df)

    ### Case 2: A 'table' model which has no identifiers:
```

```
    # Retrieving some example data from magma to use as our update
    table <- retrieve(
        magma, projectName = "example", modelName = "demographic",
        recordNames = "all",
        attributeNames = c("subject", "name", "value")
        )
    table
    # Keys to note:
    # - the first column of this df holds the parent record identifiers we
    #     wish to target. The fact that this column is properly named as
    #     'subject' does matter, but this column does not need to have
    #     been the first column.
    # - all subsequent columns hold the new values and are named as the
    #     attributes we wish to update.

    ## Key decision:
    # For table models, you must additionally decide whether to 'append' to
    #  (meaning to add them in addition to all previous records which
    #  attach to the same parents), or 'replace' (meaning clear all previous
    #  records attaching to the same parents this update hits, so that only
    #  the values within this update will exist) current values of the
    #  target model with your update's values.
    # This choice is given to the 'table.method' input.

    # To update values of the 'table'-type "demographics" model
    updateFromDF(
        target = magma,
        projectName = "example",
        modelName = "demographic",
        table.method = "replace",
        df = table)

}
```

---

updateMatrix *A matrix-specific wrapper of* updateValues

Description

A matrix-specific wrapper of updateValues which can take in a matrix, data.frame, or file path, directly.

Usage

```
updateMatrix(
  target,
  projectName,
  modelName,
  attributeName,
```

```
  matrix,
  autolink = FALSE,
  dryRun = FALSE,
  separator = ",",
  auto.proceed = FALSE,
  revisions.only = FALSE,
  template = NULL,
  ...
)
```

Arguments

target	A list, which can be created using magmaRset, containing your authorization 'token' (a string), a 'url' of magma to target (a string), and optional 'opts' for specifying additions parameters for curl requests (a named list).
projectName	Single string. The name of the project you would like to interact with. For options, see retrieveProjects.
modelName	Single string. The name of the subset data structure within the project, which are referred to as 'model's in magma, to interact with. For options, see retrieveModels or https://timur.ucsf.edu/<projectName>/map.
attributeName	String naming the matrix attribute for which to upload data.
matrix	A matrix or dataframe containing the data to upload to magma. Alternatively, a String specifying the file path of a file containing such data. No matter the provision method, colnames must be record identifiers, and rownames should match the values of 'options' associated with the target 'attribute'. Check the 'See Also' section below for how to determine the needed 'options'.
autolink	Logical. FALSE by default for safety, but often you will want to set it to TRUE. Passed through to magma, this parameter controls whether the system will attempt to connect all targeted records up with the project's root. Specifically, this means the system will 1) determine parent records of all targeted records if it can, based on the project's gnomon grammar, 2) continue parent determination for those parent records, repeating this process until reaching the project's root (the project record), then 3) creates any of these records that don't currently exist, and finally 4) creates all the assumed parent-child linkages
dryRun	Logical. FALSE by default. Passed through to magma, this parameter controls whether the system will only test whether the update is valid without making changes to the database.
separator	String indicating the field separator to use if providing matrix as a file path. Default = ",".
auto.proceed	Logical. When set to TRUE, the function does not ask before proceeding forward with the 'magma/update'.
revisions.only	Logical. For troubleshooting purposes, when set to TRUE, no data will be sent to magma. Instead, the list structure that would have been passed to the revisions input of updateValues is returned as output.
template	For internal use in minimizing excess http requests to magma, NULL or the return of retrieveTemplate(target, projectName).

... Additional parameters passed along to the internal '.retrieve()', '.query()', or '.update()' functions, for troubleshooting or advanced-user purposes only:

- `request.only` (Logical) & `json.params.only` (Logical) which 1) stop the function before its main curl request to magma and 2) returns the values that would have been sent to magma in either of two formats.
- `verbose` (Logical) sets whether to report the status of the curl request after it is performed.

Details

This function utilizes the magma/query function, documented here https://mountetna.github.io/magma.html#update, to upload data to a matrix attribute (named `attributeName`) of the `modelName` model of `projectName` project.

`matrix` data are provided either as a matrix, dataframe, or file path which points toward such data. If given as a file path, the `separator` input can be used to adjust for whether the file is a csv (the default, `separator = ","`), or tsv, `separator = "\t"`, or other format.

Data is then validated by ensuring that all row names are among the valid 'options' of the target attribute (See the See Also section below for a note on how to explore these options yourself.). Rows are reordered to be in the same order as these 'options'.

For any missing 'options', rows of NAs are added.

The data is then transformed and passed along to `updateValues`.

The `updateValues()` function will summarize records to be updated and allow the user to double-check this information before proceeding.

This user-prompt step can be bypassed (useful when running in a non-interactive way) by setting `auto.proceed = TRUE`, but NOTE: It is a good idea to always check carefully before proceeding, if possible. Data can be overwritten with NAs or zeros or the like, but improperly named records cannot be easily removed.

Value

None directly.

The function sends data to magma, and the only outputs are information reported via the console.

See Also

`updateFromDF` for a more flexible function for uploading multiple attributes-worth of (non-matrix) data at a time.

`updateValues` for the more direct replica of `magma/update` which is more even more flexible that `updateFromDF`, though a bit more complicated to use.

`retrieveTemplate`, then check the `<output>$models$<modelName>$template$attributes$<attributeName>$options` to explore the rownames that your matrix should have.

https://mountetna.github.io/magma.html#update for documentation of the underlying `magma/update` function.

Examples

```
if (interactive()) {
    # First, we use magmaRset to create an object which will tell other magmaR
    #  functions our authentication token (as well as some other optional bits).
    # When run in this way, it will ask you to give your token.
    magma <- magmaRset()

    ### Note that you likely do not have write-permissions for the 'example'
    # project, so this code can be expected to give an authorization error.

    ### Retrieve some data from magma, then update that same data.
    mat <- retrieveMatrix(magma, "example", "rna_seq", "all", "gene_tpm")

    updateMatrix(
        target = magma,
        projectName = "example",
        modelName = "rna_seq",
        attributeName = "gene_tpm",
        matrix = mat)
}
```

---

updateValues *Analogous to the '/update' function of magma*

Description

Analogous to the '/update' function of magma, allows data to be sent to magma (by users with at least "editor" authorization).

Usage

```
updateValues(
  target,
  projectName,
  revisions = list(),
  auto.proceed = FALSE,
  autolink = FALSE,
  dryRun = FALSE,
  template = NULL,
  ...
)
```

Arguments

target	A list, which can be created using magmaRset, containing your authorization 'token' (a string), a 'url' of magma to target (a string), and optional 'opts' for specifying additions parameters for curl requests (a named list).

`projectName` Single string. The name of the project you would like to interact with. For options, see `retrieveProjects`.

`revisions` A list of named lists containing the data to be updated.

List structure:

- top level name(s): modelNames, can be 1 or more.
- 2nd level name(s): recordNames, can be 1 or more.
- 3rd level name(s) & contents: the attributes to update & the values to use.

See https://mountetna.github.io/magma.html#update for additional formatting details.

`auto.proceed` Logical. When set to TRUE, the function does not ask before proceeding forward with the 'magma/update'.

`autolink` Logical. FALSE by default for safety, but often you will want to set it to TRUE. Passed through to magma, this parameter controls whether the system will attempt to connect all targeted records up with the project's root. Specifically, this means the system will 1) determine parent records of all targeted records if it can, based on the project's gnomon grammar, 2) continue parent determination for those parent records, repeating this process until reaching the project's root (the project record), then 3) creates any of these records that don't currently exist, and finally 4) creates all the assumed parent-child linkages

`dryRun` Logical. FALSE by default. Passed through to magma, this parameter controls whether the system will only test whether the update is valid without making changes to the database.

`template` For internal use in minimizing excess http requests to magma, NULL or the return of `retrieveTemplate(target, projectName)`.

`...` Additional parameters passed along to the internal '.retrieve()', '.query()', or '.update()' functions, for troubleshooting or advanced-user purposes only:

- `request.only` (Logical) & `json.params.only` (Logical) which 1) stop the function before its main curl request to magma and 2) returns the values that would have been sent to magma in either of two formats.
- `verbose` (Logical) sets whether to report the status of the curl request after it is performed.

Details

This function mimics the activity of the magma/update function, documented here https://mountetna.github.io/magma.html#update, with the main difference being that the `revisions` input should be in nested list format rather than nested hash (because R does not support hash structures).

Internally, the function:

1. Summarizes records of each model that will be targeted for updating.
2. Prompts the user before proceeding (unless `auto.proceed` is set to `TRUE`)
3. Directly passes its inputs along to magma/update via a curl request.

Value

None directly.

The function sends data to magma, and the only outputs are information reported via the console.

See Also

https://mountetna.github.io/magma.html#update for documentation of the underlying magma/update function.

updateMatrix for a matrix-dedicated version of this function which can be provided a matrix, or matrix's file location, directly.

Examples

```
if (interactive()) {
    # First, we use magmaRset to create an object which will tell other magmaR
    #  functions our authentication token (as well as some other optional bits).
    # When run in this way, it will ask you to give your token.
    magma <- magmaRset()

    # Note that you likely do not have write-permissions for the 'example'
    # project, so this code can be expected to give an authorization error.

    updateValues(
        target = magma,
        projectName = "example",
        autolink = TRUE,
        dryRun = FALSE,
        revisions = list(
            # model
            'rna_seq' = list(
                # record
                'EXAMPLE-HS1-WB1-RSQ1' = list(
                    # attribute
                    'fraction' = list(
                        # value(s)
                        "Tcells"
                        )
                    )
                )
            )
        )
}
```

Index