Package 'WikidataR'

July 12, 2021

Type Package

Title Read-Write API Client Library for 'Wikidata'

Version 2.3.1

Date 2021-07-09

Author Thomas Shafee [aut, cre], Oliver Keyes [aut, cre], Serena Signorelli [aut, cre], Alex Lum [ctb], Christian Graul [ctb], Mikhail Popov [ctb]

Maintainer Thomas Shafee <T.Shafee@latrobe.edu.au>

Description An API client for the Wikidata <https://www.wikidata.org/wiki/Wikidata:Main_Page> store of semantic data.

BugReports https://github.com/TS404/WikidataR/issues

URL https://github.com/TS404/WikidataR/issues

License MIT + file LICENSE

Imports httr, jsonlite, WikipediR, WikidataQueryServiceR, tibble, dplyr, stringr, Hmisc, progress, pbapply, stats, readr, crayon, utils

Suggests markdown, testthat, tidyverse, knitr, pageviews

RoxygenNote 7.1.1

Encoding UTF-8

Depends R (>= 3.5.0)

NeedsCompilation no

Repository CRAN

Date/Publication 2021-07-12 07:50:02 UTC

R topics documented:

`as_pid` *Convert an input to a property PID*

Description

Convert an input string to the most likely property PID

Usage

```
as_pid(x)
```

Arguments

x	a vector, data frame, or tibble of strings representaing wikidata properties

Value

if the inputted string is a valid PID, return the string. If the inputted string matches a property label, return its PID. If the inputted string matches multiple labels of multiple properties, return the PID of the first hit.

Examples

```
# if input string is a valid PID
as_pid("P50")
# if input string matches multiple item labels
as_pid("author")
# if input string matches a single unique label
as_pid("Scopus author ID")
```

---

as_qid *Convert an input to a item QID*

Description

Convert an input string to the most likely item QID

Usage

```
as_qid(x)
```

Arguments

x	a vector, data frame, or tibble of strings representaing wikidata items

Value

if the inputted string is a valid QID, return the string. If the inputted string matches an item label, return its QID. If the inputted string matches multiple labels of multiple items, return the QID of the first hit.

Examples

```
# if input string is a valid QID
as_qid("Q42")
# if input string matches multiple item labels
as_qid("Douglas Adams")
# if input string matches a single unique label
as_qid("Douglas Adams and the question of arterial blood pressure in mammals")
```

as_quot *Add quotations marks*

Description

Add escaped quotation marks around strings that need them ready for submision to an API

Usage

```
as_quot(x, format = "tibble")
```

Arguments

x	a vector, data frame, or tibble of strings
format	either "tibble" / "csv" to use plain quotation marks (default), or "api" / "website" to use '%22'

Value

tibble of items inside of escaped quotation marks unless they are already in escaped quotation marks, is a QID, (in which chase it is returned unchanged)

Examples

```
as_quot("text")
```

as_sid *Convert an input to a source property SID*

Description

Convert an input string to the most likely source SID (equivalent to PID)

Usage

```
as_sid(x)
```

Arguments

x	a vector, data frame, or tibble of strings representaing wikidata source properties

Value

if the inputted string is a valid SID, return the string. If the inputted string matches a property label, return its SID If the inputted string matches multiple labels of multiple properties, return the SID of the first hit.

Examples

```
# if input string is a valid SID
as_pid("S854")
# if input string matches multiple item labels
as_pid("URL")
# if input string matches a single unique label
as_pid("Reference URL")
```

---

check_input *Generic input checker*

Description

Utility function to handle namespaces. Used by `get_item` and `get_property`

Usage

```
check_input(input, substitution)
```

Arguments

`input`	string to check
`substitution`	string for what's been looked for

Value

boolian indicating whether the checked string contains a match for the substitution string

---

createrows *"CREATE" rows*

Description

Add in empty lines for QuickStatements CREATE rows that mint new QIDs. This is a slightly messy quirk of the QuickStatements format that mints new QIDs via a line containing only "CREATE", so this function is a way to approximate that bevaviour in a tibble

Usage

```
createrows(items, vector)
```

Arguments

`items`	a vector, data frame or tibble of items (which may or may not contain the keyword "CREATE")
`vector`	a vector of properties or values which may be expanded based on the items vector

Value

if the vector is NULL, return NULL. Otherwise, if the "CREATE" keyword appears in the items vector, insert blank strings at those positions in the vector.

---

createrows.tidy *"CREATE" rows from tidy format*

Description

Add in QuickStatements CREATE rows that mint new QIDs from tidy input data. New items are created by any item starting that starts with the text "CREATE" followed by any unique ID.

Usage

```
createrows.tidy(QS.tib)
```

Arguments

QS.tib	a tibble of items, values and properties (optionally qualifiers and sources).

Value

a tibble, with items that start with "CREATE" followed by any unique text causing the addition of a "Create" line above, being replaced with "LAST" in the Quickstatemnts format to create new QIDs.

---

disambiguate_QIDs *Disambiguate QIDs*

Description

Interactive function that presents alternative possible QID matches for a list of text strings and provides options for choosing between alternatives, rejecting all presented alternatives, or creating new items. Useful in cases where a list of text strings may have either missing wikidata items or multiple alternative potential matches that need to be manually disambuguated. Can also used on lists of lists (see examples). For long lists of items, the process can be stopped partway through and the returned vector will indicate where the process was stopped.

Usage

```
disambiguate_QIDs(
  list,
  variablename = "variables",
  variableinfo = NULL,
  filter_property = NULL,
  filter_variable = NULL,
  filter_firsthit = FALSE,
  limit = 10
)
```

Arguments

list	a list or vector of text strings to find potential QID matches to. Can also be a list of lists (see examples)
variablename	type of items in the list that are being disambiguated (used in messages)
variableinfo	additional information about items that are being disambiguated (used in messages)
filter_property	property to filter on (e.g. "P31" to filter on "instance of")
filter_variable	values of that property to use to filter out (e.g. "Q571" to filter out books)
filter_firsthit	apply filter to the first match presented or only if alternatives requested? (default = FALSE, note: true is slower if filter not needed on most matches)
limit	number of alternative possible wikidata items to present if multiple potential matches

Value

a vector of:

QID Selected QID (for when an appropriate Wikidata match exists)

CREATE Mark that a new Wikidata item should be created (for when no appropriate Wikidata match yet exists)

NA Mark that no Wikidata item is needed

STOP Mark that the process was halted at this point (so that output can be used as input to the function later)

Examples

```
## Not run:
#Disambiguating possible QID matches for these music genres
#Results should be:
# "Q22731" as the first match
# "Q147538" as the first match
# "Q3947" as the second alternative match
disambiguate_QIDs(list=c("Rock","Pop","House"),
                  variablename="music genre")

#Disambiguating possible QID matches for these three words, but not the music genres
#This will take longer as the filtering step is slower
#Results should be:
# "Q22731" (the material) as the first match
# "Q147538" (the soft drink) as the second alternative match
# "Q3947" (the building) as the first match
disambiguate_QIDs(list=c("Rock","Pop","House"),
                  filter_property="instance of",
                  filter_variable="music genre",
                  filter_firsthit=TRUE,
```

```
                variablename="concept, not the music genre")

#Disambiguating possible QID matches for the multiple expertise of
#these three people as list of lists
disambiguate_QIDs(list=list(alice=list("physics","chemistry","maths"),
                            barry=list("history"),
                            clair=list("law","genetics","ethics")),
                variablename="expertise")

## End(Not run)
```

`extract_claims` *Extract Claims from Returned Item Data*

Description

extract claim information from data returned using `get_item`.

Usage

```
extract_claims(items, claims)
```

Arguments

`items`	a list of one or more Wikidata items returned with `get_item`.
`claims`	a vector of claims (in the form "P321", "P12") to look for and extract.

Value

a list containing one sub-list for each entry in `items`, and (below that) the found data for each claim. In the event a claim cannot be found for an item, an `NA` will be returned instead.

Examples

```
# Get item data
adams_data <- get_item("42")
# Get claim data
claims <- extract_claims(adams_data, "P31")
```

extract_para *Extract a paragraph of text*

Description

Return the nth paragraph of a section of text Useful for extracting information from wikipedia or other wikimarkup text

Usage

```
extract_para(text, para = 1, templ = NULL)
```

Arguments

text	the input text as a string
para	number indicating which paragraph(s) to return (default=1)
templ	an optional string specifying a mediawikitemplate within which to restrict the search restrict search

Value

the nth paragraph of the input text.

filter_qids *Filter QIDs*

Description

For a QID or vector of QIDs, remove ones that match a particular statement (e.g. remove all that are instances of academic publications or books).

Usage

```
filter_qids(
  ids,
  property = "P31",
  filter = c("Q737498", "Q5633421", "Q7725634", "Q13442814", "Q18918145"),
  message = NULL
)
```

Arguments

ids	QIDs to check
property	property to check (default = P31 to filter on "instance of")
filter	values of that property to use to filter out (default = Q737498, Q5633421, Q7725634, Q13442814, and Q18918145 to remove academic publications or books)
message	message to return (useful for disambiguate_QIDs function)

Value

a vector of QIDs that do not match the property filter

Examples

```
## Not run:
# Filter three items called "Earth Science" to show only those that aren't
# books, journals or journal articles
filter_qids(ids = c("Q96695546","Q8008","Q58966429"),
            property = "P31",
            filter = c("Q737498","Q5633421","Q7725634","Q13442814","Q18918145"))

## End(Not run)
```

`find_item` *Search for Wikidata items or properties that match a search term*

Description

`find_item` and `find_property` allow you to retrieve a set of Wikidata items or properties where the aliase or descriptions match a particular search term. As with other `WikidataR` code, custom print methods are available; use `str` to manipulate and see the underlying structure of the data.

Usage

```
find_item(search_term, language = "en", limit = 10, ...)

find_property(search_term, language = "en", limit = 10)
```

Arguments

`search_term`	a term to search for.
`language`	the language to return the labels and descriptions in; this should consist of an ISO language code. Set to "en" by default.
`limit`	the number of results to return; set to 10 by default.
`\dots`	further arguments to pass to httr's GET.

See Also

`get_random` for selecting a random item or property, or `get_item` for selecting a specific item or property.

Examples

```
#Check for entries relating to Douglas Adams in some way
adams_items <- find_item("Douglas Adams")

#Check for properties involving the peerage
peerage_props <- find_property("peerage")
```

---

get_example *Get an example SPARQL query from Wikidata*

Description

Gets the specified example(s) from [SPARQL query service examples page](https://www.wikidata.org/wiki/Wikidata:SPARQ
using [Wikidata's MediaWiki API](https://www.wikidata.org/w/api.php).

Usage

```
get_example(example_name)
```

Arguments

example_name	the names of the examples as they appear on [this page](https://www.wikidata.org/wiki/Wikidata:SPARQI

Details

If you are planning on extracting multiple examples, please provide all the names as a single vector for efficiency.

Value

The SPARQL query as a character vector.

See Also

[query_wikidata]

Examples

```
## Not run:
sparql_query <- extract_example(c("Cats", "Horses"))
query_wikidata(sparql_query)
# returns a named list with two data frames
# one called "Cats" and one called "Horses"
sparql_query <- extract_example("Largest cities with female mayor")
cat(sparql_query)
query_wikidata(sparql_query)

## End(Not run)
```

get_geo_box *Get geographic entities based on a bounding box*

Description

get_geo_box retrieves all geographic entities in Wikidata that fall between a bounding box between two existing items with geographic attributes (usually cities).

Usage

```
get_geo_box(
  first_city_code,
  first_corner,
  second_city_code,
  second_corner,
  language = "en",
  ...
)
```

Arguments

first_city_code	a Wikidata item, or series of items, to use for one corner of the bounding box.
first_corner	the direction of first_city_code relative to city (eg "NorthWest", "South-East").
second_city_code	a Wikidata item, or series of items, to use for one corner of the bounding box.
second_corner	the direction of second_city_code relative to city (eg "NorthWest", "South-East").
language	the two-letter language code to use for the name of the item. "en" by default.
\dots	further arguments to pass to httr's GET.

Value

a data.frame of 5 columns:

- item the Wikidata identifier of each object associated with entity.
- name the name of the item, if available, in the requested language. If it is not available, NA will be returned instead.
- latitude the latitude of item
- longitude the longitude of item
- entity the entity the item is associated with (necessary for multi-entity queries).

See Also

get_geo_entity for using an unrestricted search or simple radius, rather than a bounding box.

Examples

```
# Simple bounding box
bruges_box <- get_geo_box("Q12988", "NorthEast", "Q184287", "SouthWest")

# Custom language
bruges_box_fr <- get_geo_box("Q12988", "NorthEast", "Q184287", "SouthWest",
                                        language = "fr")
```

`get_geo_entity` *Retrieve geographic information from Wikidata*

Description

`get_geo_entity` retrieves the item ID, latitude and longitude of any object with geographic data associated with *another* object with geographic data (example: all the locations around/near/associated with a city).

Usage

```
get_geo_entity(entity, language = "en", radius = NULL, limit = 100, ...)
```

Arguments

`entity`	a Wikidata item (`Q...`) or series of items, to check for associated geo-tagged items.
`language`	the two-letter language code to use for the name of the item. "en" by default, because we're imperialist anglocentric westerners.
`radius`	optionally, a radius (in kilometers) around `entity` to restrict the search to.
`limit`	the maximum number of results to return.
`\dots`	further arguments to pass to httr's GET.

Value

a data.frame of 5 columns:

- item the Wikidata identifier of each object associated with `entity`.
- name the name of the item, if available, in the requested language. If it is not available, `NA` will be returned instead.
- latitude the latitude of `item`
- longitude the longitude of `item`
- entity the entity the item is associated with (necessary for multi-entity queries).

See Also

`get_geo_box` for using a bounding box rather than an unrestricted search or simple radius.

Examples

```
# All entities
sf_locations <- get_geo_entity("Q62")

# Entities with French, rather than English, names
sf_locations <- get_geo_entity("Q62", language = "fr")

# Entities within 1km
sf_close_locations <- get_geo_entity("Q62", radius = 1)

# Multiple entities
multi_entity <- get_geo_entity(entity = c("Q62", "Q64"))
```

---

get_item *Retrieve specific Wikidata items or properties*

Description

get_item and get_property allow you to retrieve the data associated with individual Wikidata items and properties, respectively. As with other WikidataR code, custom print methods are available; use str to manipulate and see the underlying structure of the data.

Usage

```
get_item(id, ...)

get_property(id, ...)
```

Arguments

id	the ID number(s) of the item or property you're looking for. This can be in various formats; either a numeric value ("200"), the full name ("Q200") or even with an included namespace ("Property:P10") - the function will format it appropriately. This function is vectorised and will happily accept multiple IDs.
\dots	further arguments to pass to httr's GET.

See Also

get_random for selecting a random item or property, or find_item for using search functionality to pull out item or property IDs where the descriptions or aliases match a particular search term.

Examples

```
#Retrieve a specific item
adams_metadata <- get_item("42")
```

```
#Retrieve a specific property
object_is_child <- get_property("P40")
```

`get_names_from_properties` *Get names of properties*

Description

For a claim or set of claims, return the names of the properties

Usage

```
get_names_from_properties(properties)
```

Arguments

`properties`	a claims list from `extract_claims`

Value

tibble of labels for each property for a set of claims

`get_random_item` *Retrieve randomly-selected Wikidata items or properties*

Description

`get_random_item` and `get_random_property` allow you to retrieve the data associated with randomly-selected Wikidata items and properties, respectively. As with other `WikidataR` code, custom print methods are available; use `str` to manipulate and see the underlying structure of the data.

Usage

```
get_random_item(limit = 1, ...)

get_random_property(limit = 1, ...)
```

Arguments

`limit`	how many random items to return. 1 by default, but can be higher.
`\dots`	arguments to pass to httr's GET.

See Also

get_item for selecting a specific item or property, or find_item for using search functionality to pull out item or property IDs where the descriptions or aliases match a particular search term.

Examples

```
## Not run:
#Random item
random_item <- get_random_item()

#Random property
random_property <- get_random_property()

## End(Not run)
```

`identifier_from_identifier` *identifier from identifier*

Description

convert unique identifiers to other unique identifiers

Usage

```
identifier_from_identifier(
  property = "ORCID iD",
  return = "IMDb ID",
  value = "0000-0002-7865-7235"
)
```

Arguments

property	the identifier property to search (for caveats, see as_pid)
return	the identifier property to convert to
value	the identifier value to match

Value

vector of identifiers corresponding to identifiers submitted

Examples

```
identifier_from_identifier('ORCID iD','IMDb ID',c('0000-0002-7865-7235','0000-0003-1079-5604'))
```

initials *Format short form person names*

Description

Converting names into first initial and surname, or just initials

Usage

```
initials(x, format = "FLast")
```

Arguments

x	a vector of people's names as strings
format	a vector of strings of either "FLast" or "FL" to indicate the output format

Value

the inputted name strings with first names shortened based on the selected format.

list_properties *List properties of a Wikidata item*

Description

for a downloaded wikidata item, list the properties of all statements

Usage

```
list_properties(item, names = FALSE)
```

Arguments

item	a list of one or more Wikidata items returned with get_item.
names	a boolian for whether to return property names, or just P numbers and extract.

Value

a list containing one sub-list for each entry in items, and (below that) the found data for each claim. In the event a claim cannot be found for an item, an NA will be returned instead.

Examples

```
# Get item data
adams_data <- get_item("42")
# Get claim data
claims <- extract_claims(adams_data, "P31")
```

print.find_item *Print method for find_item*

Description

print found items.

Usage

```
## S3 method for class 'find_item'
print(x, ...)
```

Arguments

x	find_item object with search results
...	Arguments to be passed to methods

print.find_property *Print method for find_property*

Description

print found properties.

Usage

```
## S3 method for class 'find_property'
print(x, ...)
```

Arguments

x	find_property object with search results
...	Arguments to be passed to methods

print.wikidata *Print method for Wikidata objects*

Description

print found objects generally.

Usage

```
## S3 method for class 'wikidata'
print(x, ...)
```

Arguments

x	wikidata object from get_item, get_random_item, get_property or get_random_property
...	Arguments to be passed to methods

See Also

get_item, get_random_item, get_property or get_random_property

qid_from_DOI *QID from DOI*

Description

simple converter from DOIs to QIDs (for items in wikidata)

Usage

```
qid_from_DOI(DOI = "10.15347/WJM/2019.001")
```

Arguments

DOI	digital object identifiers submitted as strings

Value

vector of QIDs corresponding to DOIs submitted

qid_from_identifier *QID from identifier*

Description

convert unique identifiers to QIDs (for items in wikidata).

Usage

```
qid_from_identifier(
  property = "DOI",
  value = c("10.15347/WJM/2019.001", "10.15347/WJM/2020.002")
)
```

Arguments

property	the identifier property to search (for caveats, see as_pid)
value	the identifier value to match

Value

vector of QIDs corresponding to identifiers submitted

Examples

```
qid_from_identifier('ISBN-13','978-0-262-53817-6')
```

qid_from_name *QID from label name*

Description

simple converter from label names to QIDs (for items in wikidata). Essentially a simplification of find_item

Usage

```
qid_from_name(name = "Thomas Shafee", limit = 100, format = "vector")
```

Arguments

name	name labels submitted as strings
limit	if multiple QIDs match each submitted name, how many to return
format	output format ('vector' to return a simple vector, or 'list' to return a nested list)

Value

vector of QIDs corresponding to names submitted. Note: some names may return multiple QIDs.

qid_from_ORCID *QID from ORCID*

Description

simple converter from ORCIDs to QIDs (for items in wikidata)

Usage

```
qid_from_ORCID(ORCID = "0000-0002-2298-7593")
```

Arguments

ORCID	digital object identifiers submitted as strings

Value

vector of QIDs corresponding to ORCIDs submitted

query_wikidata *Send one or more SPARQL queries to WDQS*

Description

Makes a POST request to Wikidata Query Service SPARQL endpoint.

Usage

```
query_wikidata(sparql_query, format = "tibble", ...)
```

Arguments

sparql_query	SPARQL query (can be a vector of queries)
format	'tibble' (default) returns a pure character data frame, 'simple' returns a pure character vector, while 'smart' fetches JSON-formatted data and returns a tibble with datetime columns converted to 'POSIXct'
\dots	Additional parameters to supply to [httr::POST]

Value

A 'tibble' or 'vector'. Note: QID values will be returned as QIDs, rather than URLs.

Query limits

There is a hard query deadline configured which is set to 60 seconds. There are also following limits:
- One client (user agent + IP) is allowed 60 seconds of processing time each 60 seconds - One client
is allowed 30 error queries per minute See [query limits section](https://www.mediawiki.org/wiki/Wikidata_Query_Service/U
in the WDQS user manual for more information.

Examples

```
# R's versions and release dates:
sparql_query <- 'SELECT DISTINCT
  ?softwareVersion ?publicationDate
  WHERE {
    BIND(wd:Q206904 AS ?R)
    ?R p:P348 [
      ps:P348 ?softwareVersion;
      pq:P577 ?publicationDate
    ] .
}'
query_wikidata(sparql_query)

## Not run:
# "smart" format converts all datetime columns to POSIXct
query_wikidata(sparql_query, format = "smart")

## End(Not run)
```

---

searcher *Convert an input to a item QID*

Description

Convert an input string to the most likely item QID

Usage

```
searcher(search_term, language, limit, type, ...)
```

Arguments

search_term	a term to search for.
language	the language to return the labels and descriptions in; this should consist of an ISO language code. Set to "en" by default.
limit	the number of results to return; set to 10 by default.
type	type of wikidata object to return (default = "item")
\dots	Additional parameters to supply to [httr::POST]

Value

If the inputted string matches an item label, return its QID. If the inputted string matches multiple labels of multiple items, return the QID of the first hit. If the inputted string is already a QID, return the string.

Examples

```
# if input string is a valid QID
as_qid("Q42")
# if input string matches multiple item labels
as_qid("Douglas Adams")
# if input string matches a single unique label
as_qid("Douglas Adams and the question of arterial blood pressure in mammals")
```

---

sparql_query *Download full Wikidata items matching a sparql query*

Description

Utility wrapper for wikidata spargle endpoint to download items. Used by `get_geo_entity` and `get_geo_box`

Usage

```
sparql_query(query, ...)
```

Arguments

`query`	the sparql query as a string
`\dots`	Additional parameters to supply to [httr::POST]

Value

a download of the full wikidata objects formatted as a nested json list

---

unspecial *Remove special characters*

Description

Special characters can otherwise mess up wikidata read-writes

Usage

```
unspecial(x)
```

Arguments

x	a vector of strings to check for special characters

Value

the inputted strings with special characters replaced with closest match plan characters.

---

url_to_id *Extract an identifier from a wikidata URL*

Description

Convert a URL ending in an identifier (returned by SPARQL queries) to just the plain identifier (QID or PID).

Convert a URL ending in an identifier (returned by SPARQL queries) to just the plan identifier (QID or PID).

Usage

```
url_to_id(x)

url_to_id(x)
```

Arguments

x	a vector of strings representing wikidata URLs

Value

if the URL ends in a QID or PID, return that PID or QID, else return the original string

QID or PID

Examples

```
url_to_id("http://www.wikidata.org/entity/42")
url_to_id("http://www.wikidata.org/Q42")
```

WD.globalvar *Global variables for Wikidata properties*

Description

A dataset of Wikidata global variables.

Format

A list of tibbles documenting key property constraints from wikidata

SID.valid valid reference source properties

PID.datatype required data type for each property

PID.constraint expected regex match for each property

lang.abbrev language abbreviations

lang.abbrev.wiki language abbreviations for current wikis

abbrev.wiki Wikimedia abbreviations for current wikis ...

wd_query *Download a Wikidata item*

Description

Utility wrapper for wikidata API to download item. Used by `get_item` and `get_property`

Usage

```
wd_query(title, ...)
```

Arguments

`title`	the wikidata item or property as a string
`\dots`	Additional parameters to supply to [httr::POST]

Value

a download of the full wikidata object (item or property) formatted as a nested json list

wd_rand_query *Download random Wikidata items*

Description

Utility wrapper for wikidata API to download random items. Used by `random_item`

Usage

```
wd_rand_query(ns, limit, ...)
```

Arguments

`ns`	string indicating namespace, most commonly "Main" for QID items, "Property" for PID properties
`limit`	how many random objesct to return
`\dots`	Additional parameters to supply to [httr::POST]

Value

a download of the full wikidata objects (items or properties) formatted as nested json lists

WikidataR *API client library for Wikidata*

Description

This package serves as an API client for reading and writing to and from Wikidata, (including via the QuickStatements format), as well as for reading from Wikipedia.

See Also

`get_random` for selecting a random item or property, `get_item` for a /specific/ item or property, or `find_item` for using search functionality to pull out item or property IDs where the descriptions or aliases match a particular search term.

write_wikidata *Write statements to Wikidata*

Description

Upload data to wikidata, including creating items, adding statements to existing items (via the quickstatements format and API).

Usage

```
write_wikidata(
  items,
  properties = NULL,
  values = NULL,
  qual.properties = NULL,
  qual.values = NULL,
  src.properties = NULL,
  src.values = NULL,
  remove = FALSE,
  format = "tibble",
  api.username = NULL,
  api.token = NULL,
  api.format = "v1",
  api.batchname = NULL,
  api.submit = TRUE
)
```

Arguments

items	a vector of strings indicating the items to which to add statements (as QIDs or labels). Note: if labels are provided, and multiple items match, the first matching item will be used (see as_qid function), so use with caution. New QIDs can be created by using the "CREATE_xyz", where "_xyz" is any unique string. Using the same id will add additional statemnts to those new items
properties	a vector of strings indicating the properties to add as statements (as PIDs or labels). Note: if labels are provided, and multiple items match, the first matching item will be used (see as_pid function), so use with caution. Four special properties can also be used: labels, aliases, descriptions and sitelinks. See [this link](https://www.wikidata.org/wiki/Help:QuickStatements#Adding_labels,_aliases,_descriptions_and_s for the syntax.
values	a vector of strings indicating the values to add as statements (as QIDs or strings). Note: if strings are provided, they will be treated as plain text.
qual.properties	a vector, data frame, or tibble of strings indicating the properties to add as qualifiers to statements (as PIDs or labels). Note: if labels are provided, and multiple items match, the first matching item will be used (see as_pid function), so use with caution.

`qual.values` a vector, data frame, or tibble of strings indicating the values to add as statements (as QIDs or strings). Note: if strings are provided, they will be treated as plain text.

`src.properties` a vector, data frame, or tibble of strings indicating the properties to add as reference sources to statements (as SIDs or labels). Note: if labels are provided, and multiple items match, the first matching item will be used (see `as_sid` function), so use with caution.

`src.values` a vector, data frame, or tibble of strings indicating the values to add reference sources to statements (as QIDs or strings). Note: if strings are provided, they will be treated as plain text.

`remove` a vector of boolians for each statemnt indicating whether it should be removed from the item rather than added (default = FALSE)

`format` output format as a string. Options include:

tibble easiest format to further manuipulation in R

csv can be copy-pasted to [the QuickStatements website](https://quickstatements.toolforge.org/) (or manipulated in a spreadsheet programs)

api a url that can be copy-pasted into a web browser, or automatically submitted (see `api.submit` parameter)

website open a [QuickStatements](https://quickstatements.toolforge.org/) web browser window summarising the edits to be made to Wikidata)

`api.username` a string indicating your wikimedia username

`api.token` a string indicating your api token (the unique identifier that you can find listed at [your user page](https://quickstatements.toolforge.org/#/user))

`api.format` a string indicateing which version of the quickstatement format used to submit the api (default = "v1")

`api.batchname` a string create a named batch (listed at [your batch history page](https://quickstatements.toolforge.org/#/ba and tag in the edit summaries

`api.submit` boolian indicating whether to submit instruction directly to wikidata (else returns the URL that can be copy-pasted into a web browser)

Value

data formatted to upload to wikidata (via quickstatemsnts), optionally also directly uploded to wikidata (see `format` parameter).

Examples

```
# Add a statement to the "Wikidata sandbox" item (Q4115189)
# to say that it is an "instance of" (P31) of Q1 (the universe).
# The instruction will submit directly to wikidata via the API
# (if you include your wikimedia username and token)

write_wikidata(items       = "Wikidata Sandbox",
               properties  = "instance of",
               values      = "Q1",
               format      = "api",
```

```
                api.username = "myusername",
                api.token    = , #REDACTED#
                )
  #note:
```

Index