

# Package ‘notionR’

June 24, 2025

**Title** R Wrapper for 'Notion' API

**Version** 0.0.9

**Date** 2025-06-09

**Maintainer** Eduardo Flores <eduardo@enelmargen.org>

**Description** Provides functions to query databases and notes in 'Notion', using the official REST API. To learn more about the functionality of the 'Notion' API, see <<https://developers.notion.com/>>.

**License** MIT + file LICENSE

**URL** <<https://www.enelmargen.org/notionR/>>

**Depends** R (>= 2.10)

**Imports** dplyr, httr, httr2, stringi, tibble, tidyr

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**NeedsCompilation** no

**Author** Eduardo Flores [aut, cre]

**Repository** CRAN

**Date/Publication** 2025-06-24 09:20:02 UTC

## Contents

addBlockH1 . . . . .	2
addBlockH2 . . . . .	3
addBlockParagraph . . . . .	3
archivePage . . . . .	4
createNotionPage . . . . .	4
deleteBlock . . . . .	5
filters . . . . .	5
getBlock . . . . .	7
getMultiSelectValues . . . . .	7
getNotionDatabase . . . . .	8
getNotionDatabaseMetadata . . . . .	9

getNotionPage . . . . .	10
isEmptyNotionDatabaseExport . . . . .	11
normalizeChromePageIds . . . . .	11
updateCheckbox . . . . .	12
updateDate . . . . .	12
updateEmoji . . . . .	13
updateMultiSelect . . . . .	13
updateNumber . . . . .	14
updatePageCover . . . . .	14
updateRelationship . . . . .	15
updateSelect . . . . .	15
updateText . . . . .	16
updateTitle . . . . .	16
updateURL . . . . .	17
<b>Index</b>	<b>18</b>

---

addBlockH1	<i>Adds an H1 (heading one) Block to a Page</i>
------------	---

---

## Description

Id refers to a page id, content should be only text.

## Usage

```
addBlockH1(secret, id, content, toggle = FALSE)
```

## Arguments

secret	API token
id	Page id where block will be appended
content	content to append as H1
toggle	defaults to FALSE. If TRUE, will create an H1 Toggle.

## Value

list

---

addBlockH2	<i>Adds an H2 (heading two) Block to a Page</i>
------------	---

---

**Description**

Id refers to a page id, content should be only text.

**Usage**

```
addBlockH2(secret, id, content, toggle = FALSE)
```

**Arguments**

secret	API token
id	Page id where block will be appended
content	content to append as H2
toggle	defaults to FALSE. If TRUE, will create an H1 Toggle.

**Value**

list

---

addBlockParagraph	<i>Adds a Paragraph Block to a Page</i>
-------------------	---

---

**Description**

Id refers to a page id, content should be only text. HTML will export to text.

**Usage**

```
addBlockParagraph(secret, id, content)
```

**Arguments**

secret	API token
id	Page id where block will be appended
content	content to append

**Value**

list

---

archivePage	<i>Archive a Page</i>
-------------	-----------------------

---

**Description**

Archive's a notion page.

**Usage**

```
archivePage(secret, id)
```

**Arguments**

secret	Notion API token
id	page id

**Value**

list of response  
list

**Author(s)**

Eduardo Flores

---

createNotionPage	<i>Create a Page</i>
------------------	----------------------

---

**Description**

Silent Return

**Usage**

```
createNotionPage(secret, parent_id, title_property, title = "untitled")
```

**Arguments**

secret	API token
parent_id	database id where page will be created or page under which it will nest
title_property	name of the title property
title	title of the new page

**Value**

list

**Author(s)**

Eduardo Flores

---

deleteBlock

*Delete a Block*

---

**Description**

Deletes a block

**Usage**

deleteBlock(secret, id)

**Arguments**

secret            Notion API token

id                block id

**Value**

list of response

list

**Author(s)**

Eduardo Flores

---

filters

*Filter Operators*

---

**Description**

Helps you build a database filter call with human-readable intuition. You must finish (if using pipes) with notion\_filter().

Adds a checkbox filter condition.

**Usage**

```

notion_filter(., sort = NULL)

notion_or(...)

add_checkbox_filter(property, equals = TRUE)

add_select_filter(property, equals)

add_relation_id_filter(property, equals)

```

**Arguments**

.	List of filter conditions
sort	List of sort conditions. NULL by default.
...	A combination of filters
property	name or id of property (column) in database
equals	TRUE (default) or FALSE condition to meet in checkbox. Equals (contains) for select filter.

**Details**

Still WIP, only a few operators are currently available.

**Value**

```

list
list

```

**Author(s)**

Eduardo Flores

**Examples**

```

## Not run:
# to create an OR filter on two checkbox columns with id's "tus" and "YiIx"...
my_query <- notion_or(add_checkbox_filter("tus", TRUE),
                      add_checkbox_filter("YiIx", FALSE)) %>% notion_filter()

## End(Not run)
# add a condition where checkbox should be checked
add_checkbox_filter("id_column")
# add a condition where checkbox should NOT be checked
add_checkbox_filter("id_column", FALSE)

```

---

getBlock	<i>Returns a block</i>
----------	------------------------

---

**Description**

Query API with a block ID and retrieve JSON format (for now)

**Usage**

```
getBlock(secret, id)
```

**Arguments**

secret	Notion API token
id	Notion block ID.

**Value**

JSON

**Author(s)**

Eduardo Flores

---

getMultiSelectValues	<i>Returns a vector of the available multi-select options in a field</i>
----------------------	--

---

**Description**

For a given column in a database field, creates a vector of all entries. See parameters for options.

**Usage**

```
getMultiSelectValues(  
  column,  
  strip_string = "\\|",  
  no_na = TRUE,  
  only_unique = TRUE,  
  show_progress = TRUE  
)
```

**Arguments**

column	vector of column in data.frame (usually, after using getNotionDatabase, in form db\$column_name )
strip_string	String by which we should strip >1 selects in a single database row. Defaults to " ", as this is the default behaviour in getNotionDatabase().
no_na	Strip all NA's? Defaults to TRUE.
only_unique	Export only unique values in vector? Defaults to TRUE.
show_progress	Print the count of values? Defaults to TRUE.

**Details**

does NOT call API.

**Value**

vector

**Author(s)**

Eduardo Flores

---

getNotionDatabase      *Returns a database as a data.frame*

---

**Description**

Query a database in Notion with desired filters and get a database as a data.frame in R or download the entire database (all pages).

**Usage**

```
getNotionDatabase(  
  secret,  
  database,  
  filters = NULL,  
  show_progress = FALSE,  
  all_pages = TRUE,  
  cover_icon = FALSE  
)
```



**Arguments**

secret	Notion API token
database	Notion database ID. Use normalizeChromaPageIds if using directly from browser.
filters	A list built with filter operators (see filters) to query database. If NULL will query everything.
show_progress	show prints of progress?
all_pages	download all pages (loop thru paginations)?
cover_icon	also include cover and icon metadata?

**Details**

This is actually a POST request as per Notions API: <https://developers.notion.com/reference/post-database-query>

**Value**

data.frame

**Author(s)**

Eduardo Flores

---

getNotionDatabaseMetadata

*Returns the database metadata as a data.frame*

---

**Description**

Retrieve a database's metadata as referenced in Notion API: <https://developers.notion.com/reference/get-database>

**Usage**

```
getNotionDatabaseMetadata(secret, database, raw = FALSE)
```

**Arguments**

secret	Notion API token
database	Notion database ID
raw	if TRUE will not flatten into a data.frame

**Value**

data.frame

**Author(s)**

Eduardo Flores

**Examples**

```
## Not run:  
my_db <- "database_id"  
my_secret <- "NOTION API Secret"  
  
my_db_data <- get_database_metadata(secret = my_secret, database = my_db)  
  
## End(Not run)
```

---

getNotionPage

*Get a Page*

---

**Description**

Gets a Notion Page

**Usage**

```
getNotionPage(secret, id)
```

**Arguments**

secret	Notion API token
id	page id

**Value**

list

**Author(s)**

Eduardo Flores

---

isEmptyNotionDatabaseExport  
*Is the database output empty ?*

---

**Description**

Is the database output empty ?

**Usage**

isEmptyNotionDatabaseExport(dataframe)

**Arguments**

dataframe      The database output

**Value**

vector

---

normalizeChromePageIds  
*Returns a page id from a copy-paste page id in browser*

---

**Description**

Returns a page id from a copy-paste page id in browser

**Usage**

normalizeChromePageIds(x)

**Arguments**

x                      page id in format: 1f4c70197b0e4589902d371adc1dbd9a

**Value**

character

**Author(s)**

Eduardo Flores

---

updateCheckbox	<i>Updates a Checkbox Property</i>
----------------	------------------------------------

---

**Description**

Id refers to a page in a database, and should be normalized using `normalizeChromeId()`.

**Usage**

```
updateCheckbox(secret, id, property_name, value = TRUE)
```

**Arguments**

secret	API token
id	Page id to be updated
property_name	name of property to update (should be a checkbox type property)
value	value to update. Use R bollean object. Defaults to TRUE.

**Value**

list

---

updateDate	<i>Updates a Date property</i>
------------	--------------------------------

---

**Description**

Id refers to a page in a database, and should be normalized using `normalizeChromeId()`.

**Usage**

```
updateDate(secret, id, property_name, value)
```

**Arguments**

secret	API token
id	Page id to be updated
property_name	name of property to update (should be a date type property)
value	value to update

**Value**

list

---

updateEmoji	<i>Updates an Emoji Property of a page</i>
-------------	--

---

**Description**

Id refers to a page in a database, and should be normalized using normalizeChromeId().

**Usage**

```
updateEmoji(secret, id, emoji)
```

**Arguments**

secret	API token
id	Page id to be updated
emoji	emoji to update to

**Value**

list

---

updateMultiSelect	<i>Updates a Select Property</i>
-------------------	----------------------------------

---

**Description**

Id refers to a page in a database, and should be normalized using normalizeChromeId().

**Usage**

```
updateMultiSelect(secret, id, property_name, value)
```

**Arguments**

secret	API token
id	Page id to be updated
property_name	name of property to update (should be a multiselect type property)
value	value(s) to update. Could be 1 value or multiple, created with c().

**Details**

This will rewrite whatever is already in the property. It will NOT append another select.

**Value**

list

---

updateNumber      *Updates a Number Property*

---

**Description**

Id refers to a page in a database, and should be normalized using normalizeChromeId().

**Usage**

```
updateNumber(secret, id, property_name, value)
```

**Arguments**

secret	API token
id	Page id to be updated
property_name	name of property to update (should be a number type property)
value	value to update

**Value**

list

---

updatePageCover      *Updates a Page Cover*

---

**Description**

Updates a page cover to the url specified

**Usage**

```
updatePageCover(secret, id, cover_url)
```

**Arguments**

secret	Notion API token
id	page id to be updated
cover_url	url of cover to be update

**Value**

list of response  
list

**Author(s)**

Eduardo Flores

---

updateRelationship	<i>Updates (adds) a relationship to a page id</i>
--------------------	---

---

**Description**

Id refers to a page in a database, and should be normalized using `normalizeChromeId()`.

**Usage**

```
updateRelationship(secret, id, property_name, value)
```

**Arguments**

secret	API token
id	Page id to be updated
property_name	name of property to update (should be a relationship type property)
value	value to update (should be a unique page id)

**Value**

list

---

updateSelect	<i>Updates a Select Property</i>
--------------	----------------------------------

---

**Description**

Id refers to a page in a database, and should be normalized using `normalizeChromeId()`.

**Usage**

```
updateSelect(secret, id, property_name, value)
```

**Arguments**

secret	API token
id	Page id to be updated
property_name	name of property to update (should be a select type property)
value	value to update

**Value**

list

---

updateText	<i>Updates a Text Property</i>
------------	--------------------------------

---

**Description**

Id refers to a page in a database, and should be normalized using `normalizeChromeId()`.

**Usage**

```
updateText(secret, id, property_name, value)
```

**Arguments**

secret	API token
id	Page id to be updated
property_name	name of property to update (should be a text type property)
value	value to update

**Value**

list

---

updateTitle	<i>Updates a Title Property</i>
-------------	---------------------------------

---

**Description**

Id refers to a page in a database, and should be normalized using `normalizeChromeId()`.

**Usage**

```
updateTitle(secret, id, property_name, value)
```

**Arguments**

secret	API token
id	Page id to be updated
property_name	name of property to update (should be the title property of database)
value	value to update

**Value**

list



---

updateURL	<i>Updates a URL Property</i>
-----------	-------------------------------

---

**Description**

Id refers to a page in a database, and should be normalized using `normalizeChromeId()`.

**Usage**

```
updateURL(secret, id, property_name, value)
```

**Arguments**

secret	API token
id	Page id to be updated
property_name	name of property to update (should be a URL type property)
value	value to update

**Value**

list

# Index

`add_checkbox_filter (filters)`, 5  
`add_relation_id_filter (filters)`, 5  
`add_select_filter (filters)`, 5  
`addBlockH1`, 2  
`addBlockH2`, 3  
`addBlockParagraph`, 3  
`archivePage`, 4

`createNotionPage`, 4

`deleteBlock`, 5

`filters`, 5

`getBlock`, 7  
`getMultiSelectValues`, 7  
`getNotionDatabase`, 8  
`getNotionDatabaseMetadata`, 9  
`getNotionPage`, 10

`isEmptyNotionDatabaseExport`, 11

`normalizeChromePageIds`, 11  
`notion_filter (filters)`, 5  
`notion_or (filters)`, 5

`updateCheckbox`, 12  
`updateDate`, 12  
`updateEmoji`, 13  
`updateMultiSelect`, 13  
`updateNumber`, 14  
`updatePageCover`, 14  
`updateRelationship`, 15  
`updateSelect`, 15  
`updateText`, 16  
`updateTitle`, 16  
`updateURL`, 17