

# Package ‘geniusr’

April 13, 2020

**Title** Tools for Working with the 'Genius' API

**Version** 1.2.0

**Description** Provides tools to interact nicely with the 'Genius' API  
<<https://docs.genius.com/>>.  
Search hosted content, extract associated metadata and retrieve lyrics with ease.

**URL** <https://ewenme.github.io/geniusr/>,  
<https://github.com/ewenme/geniusr>

**BugReports** <https://github.com/ewenme/geniusr/issues>

**Depends** R (>= 3.2.0)

**License** MIT + file LICENSE

**Imports** curl, dplyr, httr, purrr, rvest, stringr, tibble, xml2

**Suggests** knitr, rmarkdown, testthat (>= 2.1.0), covr

**VignetteBuilder** knitr

**Encoding** UTF-8

**LazyData** yes

**RoxygenNote** 7.0.2

**NeedsCompilation** no

**Author** Ewen Henderson [aut, cre] (<<https://orcid.org/0000-0002-4748-4693>>)

**Maintainer** Ewen Henderson <[ewenhenderson@gmail.com](mailto:ewenhenderson@gmail.com)>

**Repository** CRAN

**Date/Publication** 2020-04-13 12:10:06 UTC

## R topics documented:

album_to_df . . . . .	2
artist_to_df . . . . .	3
browse_genius . . . . .	4
browse_genius_resource . . . . .	4
genius_token . . . . .	5

get_album . . . . .	5
get_album_df . . . . .	6
get_album_meta . . . . .	7
get_album_tracklist_id . . . . .	7
get_album_tracklist_search . . . . .	8
get_annotation . . . . .	9
get_artist . . . . .	10
get_artist_df . . . . .	10
get_artist_meta . . . . .	11
get_artist_songs . . . . .	12
get_artist_songs_df . . . . .	13
get_lyrics_id . . . . .	14
get_lyrics_search . . . . .	15
get_lyrics_url . . . . .	15
get_referent . . . . .	16
get_song . . . . .	17
get_song_df . . . . .	18
get_song_meta . . . . .	18
print.genius_album . . . . .	19
print.genius_annotation . . . . .	20
print.genius_artist . . . . .	20
print.genius_referent . . . . .	21
print.genius_resource . . . . .	21
print.genius_song . . . . .	22
scrape_lyrics_id . . . . .	22
scrape_lyrics_url . . . . .	23
scrape_tracklist . . . . .	23
search_artist . . . . .	24
search_genius . . . . .	25
search_song . . . . .	25
song_to_df . . . . .	26
tidy_album_performances . . . . .	27
tidy_song_performances . . . . .	27
tidy_song_producers . . . . .	28
tidy_song_relationships . . . . .	29
tidy_song_writers . . . . .	30

**Index****31**

album\_to\_df

*Convert genius\_album object to a data frame***Description**

Convert genius\_album object to a data frame

**Usage**

```
album_to_df(x)
```

**Arguments**

x                    a genius\_album object

**Value**

a tibble

**Examples**

```
## Not run:  
album <- get_album(album_id = 337082)  
album_to_df(album)  
  
## End(Not run)
```

---

artist\_to\_df                    *Convert genius\_artist object to a data frame*

---

**Description**

Convert genius\_artist object to a data frame

**Usage**

```
artist_to_df(x)
```

**Arguments**

x                    a genius\_artist object

**Value**

a tibble

**Examples**

```
## Not run:  
artist <- get_artist(artist_id = 16775)  
artist_to_df(artist)  
  
## End(Not run)
```

---

`browse_genius`*Open the Genius homepage in your browser*

---

**Description**

Opens a browser to `https://genius.com/`.

**Usage**

```
browse_genius()
```

**Value**

A browser is opened to the Genius website if the session is interactive. The URL is returned as a character string.

**Examples**

```
browse_genius()
```

---

`browse_genius_resource`*Open the Genius url of a resource in your browser*

---

**Description**

Opens a browser to the Genius url of a Genius "resource" (i.e. the result of a successful `get_album|artist|song()` call).

**Usage**

```
browse_genius_resource(x)
```

**Arguments**

`x` a `genius_album`, `genius_artist`, or `genius_song` object

**Value**

A browser is opened to the Genius resource's url if the session is interactive. The URL is returned as a character string.

## Examples

```
song <- get_song(song_id = 3039923)
browse_genius_resource(song)
```

---

genius_token	<i>Get or set Genius access token value</i>
--------------	---

---

## Description

The API wrapper functions in this package all rely on a Genius client access token residing in the environment variable GENIUS\_API\_TOKEN. The easiest way to accomplish this is to set it in the `‘.Renviroin’` file in your home directory.

## Usage

```
genius_token(force = FALSE)
```

## Arguments

`force` force setting a new Genius API token for the current environment?

## Value

atomic character vector containing the Genius API token

---

get_album	<i>Retrieve metadata for an album</i>
-----------	---------------------------------------

---

## Description

The Genius API lets you request data for a specific album, given an album ID. `get_album()` returns this data in full.

## Usage

```
get_album(album_id, access_token = genius_token())
```

## Arguments

`album_id` ID of the album (`album_id` within an object returned by [get\\_song](#))  
`access_token` Genius' client access token, defaults to `genius_token`

**Value**

a `genius_album` object that contains the extracted content from the request, the original JSON response object and the request path.

**See Also**

See [get\\_album\\_df](#) to return a tidy data frame.

Other album: [get\\_album\\_df\(\)](#), [get\\_album\\_tracklist\\_id\(\)](#), [get\\_album\\_tracklist\\_search\(\)](#)

**Examples**

```
## Not run:  
get_album(album_id = 337082)
```

```
## End(Not run)
```

---

get\_album\_df

*Retrieve meta data for an album*

---

**Description**

The Genius API lets you return data for a specific album, given an album ID. `get_album_meta` returns this data in a tidy, but reduced, format.

**Usage**

```
get_album_df(album_id, access_token = genius_token())
```

**Arguments**

`album_id` ID of the album (`album_id` within an object returned by [get\\_song](#))  
`access_token` Genius' client access token, defaults to `genius_token`

**Value**

a tibble

**See Also**

See [get\\_album](#) to return extended data as a list.

Other album: [get\\_album\\_tracklist\\_id\(\)](#), [get\\_album\\_tracklist\\_search\(\)](#), [get\\_album\(\)](#)

**Examples**

```
## Not run:  
get_album_df(album_id = 337082)
```

```
## End(Not run)
```

---

get_album_meta	<i>Retrieve metadata for an album</i>
----------------	---------------------------------------

---

**Description**

The Genius API lets you return data for a specific album, given an album ID. `get_album_meta` returns this data in a tidy, but reduced, format.

**Usage**

```
get_album_meta(album_id, access_token = genius_token())
```

**Arguments**

album_id	ID of the album (album_id within an object returned by <a href="#">get_song</a> )
access_token	Genius' client access token, defaults to <code>genius_token</code>

**Value**

a tibble

**See Also**

[[get\\_album\(\)](#)] to return data in full as a list.

**Examples**

```
## Not run:  
get_album_meta(album_id = 337082)  
  
## End(Not run)
```

---

get_album_tracklist_id	<i>Retrieve an album's tracklisting</i>
------------------------	---

---

**Description**

Get an album's tracklisting, and song meta data, given an album ID.

**Usage**

```
get_album_tracklist_id(album_id, access_token = genius_token())
```

**Arguments**

album\_id      ID of the album (album\_id within an object returned by [get\\_song](#))  
access\_token   Genius' client access token, defaults to genius\_token

**Value**

a tibble

**See Also**

See [get\\_album\\_tracklist\\_search](#) to search for an album tracklist by searching artist/album names.

Other album: [get\\_album\\_df\(\)](#), [get\\_album\\_tracklist\\_search\(\)](#), [get\\_album\(\)](#)

**Examples**

```
## Not run:  
get_album_tracklist_id(album_id = 337082)  
  
## End(Not run)
```

---

get\_album\_tracklist\_search

*Retrieve an album's tracklisting*

---

**Description**

Attempt to get an album's tracklisting, given an artist and album name.

**Usage**

```
get_album_tracklist_search(artist_name, album_name)
```

**Arguments**

artist\_name    Name of artist  
album\_name     Name of album

**Value**

a tibble

**See Also**

See [get\\_album\\_tracklist\\_id](#) to search for an album tracklist using an album ID.

Other album: [get\\_album\\_df\(\)](#), [get\\_album\\_tracklist\\_id\(\)](#), [get\\_album\(\)](#)



## Examples

```
## Not run:  
get_album_tracklist_search(artist_name = "Kendrick Lamar",  
album_name = "DAMN.")  
  
## End(Not run)
```

---

get_annotation	<i>Retrieve metadata for an annotation</i>
----------------	--

---

## Description

The Genius API lets you return data for a specific annotation, given an annotation ID. `get_annotation` returns this data in full.

## Usage

```
get_annotation(annotation_id, access_token = genius_token())
```

## Arguments

`annotation_id` ID of the annotation  
`access_token` Genius' client access token, defaults to `genius_token`

## Details

A Genius annotation is a piece of content about a part of a document. The document may be a song (hosted on Genius) or a web page (hosted anywhere). The part of a document that an annotation is attached to is called a referent.

## Value

a `genius_annotation` object that contains the extracted content from the request, the original JSON response object and the request path.

## See Also

Other annotation: [get\\_referent\(\)](#)

## Examples

```
## Not run:  
get_annotation(annotation_id = 16511101)  
  
## End(Not run)
```

---

get_artist	<i>Retrieve metadata for an artist</i>
------------	--

---

### Description

The Genius API lets you return data for a specific artist, given an artist ID. `get_artist` returns this data in full.

### Usage

```
get_artist(artist_id, access_token = genius_token())
```

### Arguments

artist_id	ID of the artist ( <code>artist_id</code> within an object returned by <a href="#">search_artist</a> )
access_token	Genius' client access token, defaults to <code>genius_token</code>

### Value

a `genius_artist` object that contains the extracted content from the request, the original JSON response object and the request path.

### See Also

See [get\\_artist\\_df](#) to return a tidy data frame.

Other artist: [get\\_artist\\_df\(\)](#), [get\\_artist\\_songs\\_df\(\)](#), [get\\_artist\\_songs\(\)](#)

### Examples

```
## Not run:
get_artist(artist_id = 16775)

## End(Not run)
```

---

get_artist_df	<i>Retrieve metadata for an artist</i>
---------------	--

---

### Description

The Genius API lets you search for meta data for an artist, given an artist ID. `get_artist_df` returns this data in a tidy, but reduced, format.

### Usage

```
get_artist_df(artist_id, access_token = genius_token())
```

**Arguments**

artist\_id ID of the artist (artist\_id within an object returned by [search\\_artist](#))  
access\_token Genius' client access token, defaults to genius\_token

**Value**

a tibble

**See Also**

See [get\\_artist](#) to return data in full as a list.

Other artist: [get\\_artist\\_songs\\_df\(\)](#), [get\\_artist\\_songs\(\)](#), [get\\_artist\(\)](#)

**Examples**

```
## Not run:  
get_artist_df(artist_id = 16751)  
  
## End(Not run)
```

---

get_artist_meta	<i>Retrieve metadata for an artist</i>
-----------------	--

---

**Description**

The Genius API lets you search for meta data for an artist, given an artist ID. `get_artist_meta` returns this data in a tidy, but reduced, format.

**Usage**

```
get_artist_meta(artist_id, access_token = genius_token())
```

**Arguments**

artist\_id ID of the artist (artist\_id within an object returned by [search\\_artist](#))  
access\_token Genius' client access token, defaults to genius\_token

**Value**

a tibble

**See Also**

[[get\\_artist\(\)](#)] to return data in full as a list.

## Examples

```
## Not run:  
get_artist_meta(artist_id = 16751)  
  
## End(Not run)
```

---

get_artist_songs	<i>Retrieve metadata for all of an artist's songs</i>
------------------	---

---

## Description

The Genius API lets you search for song metadata of an artist, given an artist ID. `get_artist_songs` returns this data in full.

## Usage

```
get_artist_songs(  
  artist_id,  
  sort = c("title", "popularity"),  
  include_features = FALSE,  
  access_token = genius_token()  
)
```

## Arguments

<code>artist_id</code>	ID of the artist ( <code>artist_id</code> within an object returned by <a href="#">search_artist</a> )
<code>sort</code>	method to order results; by "title" (default) or by "popularity"
<code>include_features</code>	Whether to return results where artist isn't the primary artist (logical, defaults to FALSE)
<code>access_token</code>	Genius' client access token, defaults to <code>genius_token</code>

## Value

a `genius_resource` object that contains the extracted content from the request, the original JSON response object and the request path.

## See Also

See [get\\_artist\\_songs\\_df](#) to return a tidy data frame.

Other artist: [get\\_artist\\_df\(\)](#), [get\\_artist\\_songs\\_df\(\)](#), [get\\_artist\(\)](#)

## Examples

```
## Not run:
get_artist_songs(artist_id = 1421)
get_artist_songs(artist_id = 1421, sort = "popularity")
get_artist_songs(artist_id = 1421, include_features = TRUE)

## End(Not run)
```

---

get\_artist\_songs\_df     *Retrieve metadata for all of an artist's songs*

---

## Description

The Genius API lets you search for song metadata of an artist, given an artist ID. `get_artist_songs_df` returns this data in a tidy, but reduced, format.

## Usage

```
get_artist_songs_df(
  artist_id,
  sort = c("title", "popularity"),
  include_features = FALSE,
  access_token = genius_token()
)
```

## Arguments

artist_id	ID of the artist ( <code>artist_id</code> within an object returned by <a href="#">search_artist</a> )
sort	method to order results; by "title" (default) or by "popularity"
include_features	Whether to return results where artist isn't the primary artist (logical, defaults to FALSE)
access_token	Genius' client access token, defaults to <code>genius_token</code>

## Value

a tibble

## See Also

See [get\\_artist\\_songs](#) to return data in full as a list.

Other artist: [get\\_artist\\_df\(\)](#), [get\\_artist\\_songs\(\)](#), [get\\_artist\(\)](#)

## Examples

```
## Not run:  
get_artist_songs_df(artist_id = 1421)  
  
## End(Not run)
```

---

get_lyrics_id	<i>Retrieve lyrics associated with a Genius song ID</i>
---------------	---

---

## Description

Get lyrics from Genius' lyric pages using an associated song ID.

## Usage

```
get_lyrics_id(song_id, access_token = genius_token())
```

## Arguments

song\_id            ID of the song (song\_id within an object returned by [search\\_song](#))  
access\_token      Genius' client access token, defaults to genius\_token

## See Also

See [get\\_lyrics\\_url](#) to search lyrics using a song URL, [get\\_lyrics\\_search](#) to search using artist name and song title

Other lyrics: [get\\_lyrics\\_search\(\)](#), [get\\_lyrics\\_url\(\)](#)

## Examples

```
## Not run:  
get_lyrics_id(song_id = 3214267)  
  
## End(Not run)
```

---

get_lyrics_search	<i>Retrieve lyrics associated with a Genius song</i>
-------------------	--

---

**Description**

Attempt to get lyrics from Genius' lyric pages using an associated artist name and song title.

**Usage**

```
get_lyrics_search(artist_name, song_title)
```

**Arguments**

artist_name	Name of artist
song_title	Title of song

**See Also**

See [get\\_lyrics\\_id](#) to search lyrics using a song ID, [get\\_lyrics\\_url](#) to search using a song URL  
Other lyrics: [get\\_lyrics\\_id\(\)](#), [get\\_lyrics\\_url\(\)](#)

**Examples**

```
## Not run:  
get_lyrics_search(artist_name = "Anderson .Paak",  
song_title = "Come Home")  
  
## End(Not run)
```

---

get_lyrics_url	<i>Retrieve lyrics associated with a Genius lyrics page URL</i>
----------------	---

---

**Description**

Scrape lyrics from a Genius' lyric page using it's associated URL. Best used with [scrape\\_tracklist](#), when song IDs aren't returned - otherwise, [scrape\\_lyrics\\_id](#) is recommended.

**Usage**

```
get_lyrics_url(song_lyrics_url)
```

**Arguments**

song_lyrics_url	song lyrics url (like in <code>song_lyrics_url</code> returned by <a href="#">get_song_meta</a> )
-----------------	---

**See Also**

See [get\\_lyrics\\_id](#) to search lyrics using a song ID, [get\\_lyrics\\_search](#) to search using artist name and song title

Other lyrics: [get\\_lyrics\\_id\(\)](#), [get\\_lyrics\\_search\(\)](#)

**Examples**

```
## Not run:
get_lyrics_url(song_lyrics_url = "https://genius.com/Kendrick-lamar-dna-lyrics")

## End(Not run)
```

---

get_referent	<i>Retrieve metadata for a referent</i>
--------------	---

---

**Description**

The Genius API lets you return data for a specific referent. `get_referent` returns this data in full.

**Usage**

```
get_referent(
  created_by_id,
  song_id,
  web_page_id,
  access_token = genius_token()
)
```

**Arguments**

`created_by_id` ID of a user to get referents for  
`song_id` ID of a song to get referents for (pass only one of `song_id` and `web_page_id`)  
`web_page_id` ID of a web page to get referents for (pass only one of `song_id` and `web_page_id`)  
`access_token` Genius' client access token, defaults to `genius_token`

**Details**

Referents are the sections of a piece of content to which annotations are attached. Each referent is associated with a web page or a song and may have one or more annotations. Referents can be searched by the document they are attached to or by the user that created them.

**Value**

a `genius_referent` object that contains the extracted content from the request, the original JSON response object and the request path.



**See Also**

Other annotation: [get\\_annotation\(\)](#)

**Examples**

```
## Not run:  
get_referent(song_id = 3039923)  
  
## End(Not run)
```

---

get_song	<i>Retrieve metadata for a song</i>
----------	-------------------------------------

---

**Description**

The Genius API lets you return data for a specific song, given a song ID. `get_song` returns this data in full.

**Usage**

```
get_song(song_id, access_token = genius_token())
```

**Arguments**

`song_id` ID of the song (`song_id` within an object returned by [search\\_song](#))  
`access_token` Genius' client access token, defaults to `genius_token`

**Value**

a `genius_song` object that contains the extracted content from the request, the original JSON response object and the request path.

**See Also**

See [get\\_song\\_df](#) to return a tidy data frame.

Other song: [get\\_song\\_df\(\)](#), [tidy\\_album\\_performances\(\)](#), [tidy\\_song\\_performances\(\)](#), [tidy\\_song\\_producers\(\)](#), [tidy\\_song\\_relationships\(\)](#), [tidy\\_song\\_writers\(\)](#)

**Examples**

```
## Not run:  
get_song(song_id = 3039923)  
  
## End(Not run)
```

---

get_song_df	<i>Retrieve metadata for a song</i>
-------------	-------------------------------------

---

**Description**

The Genius API lets you search for meta data for a song, given a song ID. `get_song_meta` returns this data in a tidy, but reduced, format.

**Usage**

```
get_song_df(song_id, access_token = genius_token())
```

**Arguments**

`song_id` ID of the song (`song_id` within an object returned by [search\\_song](#))  
`access_token` Genius' client access token, defaults to `genius_token`

**Value**

a tibble

**See Also**

See [get\\_song](#) to return data in full as a list.

Other song: [get\\_song\(\)](#), [tidy\\_album\\_performances\(\)](#), [tidy\\_song\\_performances\(\)](#), [tidy\\_song\\_producers\(\)](#), [tidy\\_song\\_relationships\(\)](#), [tidy\\_song\\_writers\(\)](#)

**Examples**

```
## Not run:  
get_song_df(song_id = 3039923)  
  
## End(Not run)
```

---

get_song_meta	<i>Retrieve metadata for a song</i>
---------------	-------------------------------------

---

**Description**

The Genius API lets you search for meta data for a song, given a song ID. `get_song_meta` returns this data in a tidy, but reduced, format.

**Usage**

```
get_song_meta(song_id, access_token = genius_token())
```

### Arguments

song\_id            ID of the song (song\_id within an object returned by [search\\_song](#))  
access\_token      Genius' client access token, defaults to genius\_token

### Value

a tibble

### See Also

[[get\\_song\(\)](#)] to return data in full as a list.

### Examples

```
## Not run:  
get_song_meta(song_id = 3039923)  
  
## End(Not run)
```

---

`print.genius_album`      *Slightly more human-readable output for genius\_album objects*

---

### Description

Slightly more human-readable output for genius\_album objects

### Usage

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

### Arguments

x                    a genius\_album object  
...                  ignored

print.genius\_annotation

*Slightly more human-readable output for genius\_annotation objects*

---

### Description

Slightly more human-readable output for genius\_annotation objects

### Usage

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

### Arguments

x	a genius_annotation object
...	ignored

---

print.genius\_artist

*Slightly more human-readable output for genius\_artist objects*

---

### Description

Slightly more human-readable output for genius\_artist objects

### Usage

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

### Arguments

x	a genius_artist object
...	ignored

---

*print.genius\_referent* *Slightly more human-readable output for genius\_referent objects*

---

**Description**

Slightly more human-readable output for `genius_referent` objects

**Usage**

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

**Arguments**

<code>x</code>	a <code>genius_referent</code> object
<code>...</code>	ignored

---

*print.genius\_resource* *Slightly more human-readable output for genius\_resource objects*

---

**Description**

Slightly more human-readable output for `genius_resource` objects

**Usage**

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

**Arguments**

<code>x</code>	a <code>genius_resource</code> object
<code>...</code>	ignored

---

<code>print.genius_song</code>	<i>Slightly more human-readable output for genius_song objects</i>
--------------------------------	--

---

### Description

Slightly more human-readable output for genius\_song objects

### Usage

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

### Arguments

<code>x</code>	a genius_song object
<code>...</code>	ignored

---

<code>scrape_lyrics_id</code>	<i>Retrieve lyrics associated with a Genius song ID</i>
-------------------------------	---

---

### Description

Get lyrics from Genius' lyric pages using an associated song ID.

### Usage

```
scrape_lyrics_id(song_id, access_token = genius_token())
```

### Arguments

<code>song_id</code>	ID of the song (song_id within an object returned by <a href="#">search_song</a> )
<code>access_token</code>	Genius' client access token, defaults to genius_token

### Examples

```
## Not run:  
scrape_lyrics_id(song_id = 3214267)  
  
## End(Not run)
```

---

scrape_lyrics_url	<i>Retrieve lyrics associated with a Genius lyrics page URL</i>
-------------------	---

---

### Description

Scrape lyrics from a Genius' lyric page using it's associated URL. Best used with [scrape\\_tracklist](#), when song IDs aren't returned - otherwise, [scrape\\_lyrics\\_id](#) is recommended.

### Usage

```
scrape_lyrics_url(song_lyrics_url)
```

### Arguments

```
song_lyrics_url  
  song lyrics url (like in song_lyrics_url returned by get\_song\_meta)
```

### Examples

```
## Not run:  
scrape_lyrics_url(song_lyrics_url = "https://genius.com/Kendrick-lamar-dna-lyrics")  
  
## End(Not run)
```

---

scrape_tracklist	<i>Retrieve an album's tracklisting</i>
------------------	---

---

### Description

Get an album's tracklisting, and song meta data, given an album ID.

### Usage

```
scrape_tracklist(album_id, access_token = genius_token())
```

### Arguments

```
album_id      ID of the album (album_id within an object returned by get\_song)  
access_token  Genius' client access token, defaults to genius\_token
```

### Value

a tibble

## Examples

```
## Not run:  
scrape_tracklist(album_id = 337082)  
  
## End(Not run)
```

---

search_artist	<i>Search artists on Genius</i>
---------------	---------------------------------

---

## Description

The Genius API lets you search hosted content (all songs). Use `search_artist()` to return `artist_id`, `artist_name` and `artist_url` for all unique artist matches found using a search term.

## Usage

```
search_artist(search_term, n_results = 10, access_token = genius_token())
```

## Arguments

<code>search_term</code>	A character string to search for
<code>n_results</code>	Maximum no. of search results to return
<code>access_token</code>	Genius' client access token, defaults to <code>genius_token</code>

## Value

a tibble

## See Also

Other search: [search\\_genius\(\)](#), [search\\_song\(\)](#)

## Examples

```
## Not run:  
search_artist(search_term = "Lil", n_results = 20)  
  
## End(Not run)
```



---

search_genius	<i>Search documents hosted on Genius</i>
---------------	--

---

**Description**

The Genius API lets you search hosted content (all songs). Use `search_genius()` to return hits on for a given search term, in full.

**Usage**

```
search_genius(search_term, n_results = 10, access_token = genius_token())
```

**Arguments**

search_term	A character string to search for
n_results	Maximum no. of search results to return
access_token	Genius' client access token, defaults to <code>genius_token</code>

**Value**

a `genius_resource` object that contains the extracted content from the request, the original JSON response object and the request path.

**See Also**

Other search: [search\\_artist\(\)](#), [search\\_song\(\)](#)

**Examples**

```
## Not run:  
search_genius(search_term = "Lil", n_results = 100)  
  
## End(Not run)
```

---

search_song	<i>Search songs on Genius</i>
-------------	-------------------------------

---

**Description**

The Genius API lets you search hosted content (all songs). Use `search_song()` to return `song_id`, `song_name`, `lyrics_url` and `artist_id` for all unique song matches found using a search term.

**Usage**

```
search_song(search_term, n_results = 10, access_token = genius_token())
```

**Arguments**

search\_term     A character string to search for  
n\_results        Maximum no. of search results to return  
access\_token    Genius' client access token, defaults to genius\_token

**Value**

a tibble

**See Also**

Other search: [search\\_artist\(\)](#), [search\\_genius\(\)](#)

**Examples**

```
## Not run:  
search_song(search_term = "Gucci", n_results = 50)  
  
## End(Not run)
```

---

song\_to\_df

*Convert genius\_song object to a data frame*

---

**Description**

Convert genius\_song object to a data frame

**Usage**

```
song_to_df(x)
```

**Arguments**

x                    a genius\_song object

**Value**

a tibble

**Examples**

```
## Not run:  
song <- get_song(song_id = 3039923)  
song_to_df(song)  
  
## End(Not run)
```

---

`tidy_album_performances`*Extract album performances from a Genius album*

---

**Description**

Extract "album performances" (i.e. album credits) info from a Genius album object, as a tidy tibble.

**Usage**

```
tidy_album_performances(x)
```

**Arguments**

`x` A genius\_album object

**Value**

a tibble

**See Also**

See [get\\_album](#) to generate a Genius album object.

Other song: [get\\_song\\_df\(\)](#), [get\\_song\(\)](#), [tidy\\_song\\_performances\(\)](#), [tidy\\_song\\_producers\(\)](#), [tidy\\_song\\_relationships\(\)](#), [tidy\\_song\\_writers\(\)](#)

**Examples**

```
## Not run:  
album <- get_album(album_id = 337082)  
  
tidy_album_performances(album)  
  
## End(Not run)
```

---

`tidy_song_performances`*Extract custom performances from a Genius song*

---

**Description**

Extract "custom performances" (i.e. other song credits) info from a Genius song object, as a tidy tibble.

**Usage**

```
tidy_song_performances(x)
```

**Arguments**

x                    A genius\_song object

**Value**

a tibble

**See Also**

See [get\\_song](#) to generate a Genius song object.

Other song: [get\\_song\\_df\(\)](#), [get\\_song\(\)](#), [tidy\\_album\\_performances\(\)](#), [tidy\\_song\\_producers\(\)](#), [tidy\\_song\\_relationships\(\)](#), [tidy\\_song\\_writers\(\)](#)

**Examples**

```
## Not run:  
song <- get_song(song_id = 3039923)  
  
tidy_song_performances(song)  
  
## End(Not run)
```

---

tidy\_song\_producers    *Extract producer credits from a Genius song*

---

**Description**

Extract "producer artists" (i.e. producer credits) info from a Genius song object, as a tidy tibble.

**Usage**

```
tidy_song_producers(x)
```

**Arguments**

x                    A genius\_song object

**Value**

a tibble

**See Also**

See [get\\_song](#) to generate a Genius song object.

Other song: [get\\_song\\_df\(\)](#), [get\\_song\(\)](#), [tidy\\_album\\_performances\(\)](#), [tidy\\_song\\_performances\(\)](#), [tidy\\_song\\_relationships\(\)](#), [tidy\\_song\\_writers\(\)](#)

**Examples**

```
## Not run:  
song <- get_song(song_id = 3039923)  
  
tidy_song_producers(song)  
  
## End(Not run)
```

---

tidy\_song\_relationships

*Extract song relationships from a Genius song*

---

**Description**

Extract "song relationships" info from a Genius song object, as a tidy tibble.

**Usage**

```
tidy_song_relationships(x)
```

**Arguments**

x                    A genius\_song object

**Value**

a tibble

**See Also**

See [get\\_song](#) to generate a Genius song object.

Other song: [get\\_song\\_df\(\)](#), [get\\_song\(\)](#), [tidy\\_album\\_performances\(\)](#), [tidy\\_song\\_performances\(\)](#), [tidy\\_song\\_producers\(\)](#), [tidy\\_song\\_writers\(\)](#)

**Examples**

```
## Not run:  
song <- get_song(song_id = 3039923)  
  
tidy_song_relationships(song)  
  
## End(Not run)
```

---

tidy\_song\_writers      *Extract writer credits from a Genius song*

---

**Description**

Extract "writer artists" (i.e. writer credits) info from a Genius song object, as a tidy tibble.

**Usage**

```
tidy_song_writers(x)
```

**Arguments**

x                      A genius\_song object

**Value**

a tibble

**See Also**

See [get\\_song](#) to generate a Genius song object.

Other song: [get\\_song\\_df\(\)](#), [get\\_song\(\)](#), [tidy\\_album\\_performances\(\)](#), [tidy\\_song\\_performances\(\)](#), [tidy\\_song\\_producers\(\)](#), [tidy\\_song\\_relationships\(\)](#)

**Examples**

```
## Not run:  
song <- get_song(song_id = 3039923)  
  
tidy_song_writers(song)  
  
## End(Not run)
```

# Index

album\_to\_df, [2](#)  
artist\_to\_df, [3](#)

browse\_genius, [4](#)  
browse\_genius\_resource, [4](#)

genius\_token, [5](#)  
get\_album, [5](#), [6](#), [8](#), [27](#)  
get\_album\_df, [6](#), [6](#), [8](#)  
get\_album\_meta, [7](#)  
get\_album\_tracklist\_id, [6](#), [7](#), [8](#)  
get\_album\_tracklist\_search, [6](#), [8](#), [8](#)  
get\_annotation, [9](#), [17](#)  
get\_artist, [10](#), [11–13](#)  
get\_artist\_df, [10](#), [10](#), [12](#), [13](#)  
get\_artist\_meta, [11](#)  
get\_artist\_songs, [10](#), [11](#), [12](#), [13](#)  
get\_artist\_songs\_df, [10–12](#), [13](#)  
get\_lyrics\_id, [14](#), [15](#), [16](#)  
get\_lyrics\_search, [14](#), [15](#), [16](#)  
get\_lyrics\_url, [14](#), [15](#), [15](#)  
get\_referent, [9](#), [16](#)  
get\_song, [5–8](#), [17](#), [18](#), [23](#), [27–30](#)  
get\_song\_df, [17](#), [18](#), [27–30](#)  
get\_song\_meta, [15](#), [18](#), [23](#)

print.genius\_album, [19](#)  
print.genius\_annotation, [20](#)  
print.genius\_artist, [20](#)  
print.genius\_referent, [21](#)  
print.genius\_resource, [21](#)  
print.genius\_song, [22](#)

scrape\_lyrics\_id, [15](#), [22](#), [23](#)  
scrape\_lyrics\_url, [23](#)  
scrape\_tracklist, [15](#), [23](#), [23](#)  
search\_artist, [10–13](#), [24](#), [25](#), [26](#)  
search\_genius, [24](#), [25](#), [26](#)  
search\_song, [14](#), [17–19](#), [22](#), [24](#), [25](#), [25](#)  
song\_to\_df, [26](#)

tidy\_album\_performances, [17](#), [18](#), [27](#),  
[28–30](#)  
tidy\_song\_performances, [17](#), [18](#), [27](#), [27](#), [29](#),  
[30](#)  
tidy\_song\_producers, [17](#), [18](#), [27](#), [28](#), [28](#), [29](#),  
[30](#)  
tidy\_song\_relationships, [17](#), [18](#), [27–29](#),  
[29](#), [30](#)  
tidy\_song\_writers, [17](#), [18](#), [27–29](#), [30](#)