

# ENCODExplorerData

May 27, 2026

---

ENCODExplorerData	<i>ENCODExplorerData</i>
-------------------	--------------------------

---

## Description

This package aims to ease access to ENCODE file metadata by converting them into an easy-to-use `data.table`.

## Details

The main feature of ENCODExplorerData are the two ENCODE file metadata data tables exported through AnnotationHub, [encode\\_df\\_lite](#) and [encode\\_df\\_full](#)). While these can be accessed directly like any other `data.table`, we recommend using the **ENCODExplorer** companion package, which contains utility functions for querying them, using the online ENCODE search function, downloading selected files, and retrieving control-treatment experimental designs from ENCODE.

This package also exposes functions for regenerating up-to-date versions of the metadata tables. See the [fetch\\_and\\_clean\\_raw\\_ENCODE\\_tables](#), [generate\\_encode\\_df\\_lite](#) and [generate\\_encode\\_df\\_full](#) functions for more details.

## See Also

[encode\\_df\\_lite](#), [encode\\_df\\_full](#), [fetch\\_and\\_clean\\_raw\\_ENCODE\\_tables](#), [generate\\_encode\\_df\\_lite](#), [generate\\_encode\\_df\\_full](#)

---

<code>clean_table</code>	<i>Clean a data.frame that was produced by fetch_table_from_ENCODE_REST</i>
--------------------------	---

---

## Description

`data.frames` produced when converting JSON to `data.frame` with the `fromJSON` function will sometime have columns that are lists and/or columns that are `data.frames`.

## Usage

```
clean_table(table)
```

**Arguments**

`table` The table produced by the `fetch_table_from_ENCODE_REST` function.

**Details**

This function will either remove columns that are not relevant and convert columns to a vector or `data.frame`.

**Value**

a `data.frame` corresponding to the cleaned version of the input `data.frame`.

**Examples**

```
clean_table(ENCODEExplorerData:::fetch_table_from_ENCODE_REST("award"))
```

---

<code>encode_df_full</code>	<i>ENCODE file metadata, Full version</i>
-----------------------------	---

---

**Description**

Metadata for the files made available by ENCODE database as a `data.table` object. See `inst/scripts/make-data.R` for the generation process. `encode_df_full` contains all processed metadata columns, including content md5sums, cloud URLs, etc. Operations on `encode_df_full` will take longer than those on `encode_df_lite`, but may be required if some of the extra metadata columns are necessary for your needs.

**Format**

A data table

**See Also**

[generate\\_encode\\_df\\_full](#), [encode\\_df\\_lite](#)

**Examples**

```
# You can use AnnotationHub to retrieve encode_df_full.
library(AnnotationHub)
hub <- AnnotationHub()
myfiles <- subset(hub, title=="ENCODE File Metadata (Full, 2019-04-12 build)")

# You can then have a look at the metadata of the retrieved object.
myfiles

# Finally, you can access the data.table itself by indexing into the
# object returned by subset.
myfiles[[1]]
```

---

encode_df_lite	<i>ENCODE file metadata, Light version</i>
----------------	--

---

### Description

Metadata for the files made available by ENCODE database as a `data.table` object. See `inst/scripts/make-data.R` for the generation process. `encode_df_lite` contains a curated subset of the full metadata and is faster to load and easier to work with than `encode_df_full`.

### Format

A data table

### See Also

[generate\\_encode\\_df\\_lite](#), [encode\\_df\\_full](#)

### Examples

```
# You can use AnnotationHub to retrieve encode_df_lite.
library(AnnotationHub)
hub <- AnnotationHub()
myfiles <- subset(hub, title=="ENCODE File Metadata (Light, 2019-04-12 build)")

# You can then have a look at the metadata of the retrieved object.
myfiles

# Finally, you can access the data.table itself by indexing into the
# object returned by subset.
myfiles[[1]]
```

---

fetch_and_clean_raw_ENCODE_tables	<i>Fetches and preprocess the raw metadata tables from ENCODE.</i>
-----------------------------------	--

---

### Description

Fetches and preprocess the raw metadata tables from ENCODE.

### Usage

```
fetch_and_clean_raw_ENCODE_tables(
  cache_filename = "tables.RDA",
  types = get_encode_types(),
  overwrite = FALSE,
  precache = NULL
)
```

**Arguments**

cache_filename	A file name for caching the selected tables into.
types	The names of the tables to extract using the ENCODE rest api.
overwrite	If cache_filename already exists, should it be overwritten? Default: FALSE.
precache	A path to cache the raw metadata as returned by ENCODE and parsed using jsonlite. If NULL, no caching is performed. Default: FALSE.

**Value**

A list with all selected tables from ENCODE.

**Examples**

```
fetch_and_clean_raw_ENCODE_tables(cache_filename = "platform.RDA", types = "platform")
file.remove("platform.RDA")
```

---

```
generate_encode_df_full
```

*Given the raw ENCODE tables, this generate a data.table with the full set of file metadata columns.*

---

**Description**

Given the raw ENCODE tables, this generate a data.table with the full set of file metadata columns.

**Usage**

```
generate_encode_df_full(tables)
```

**Arguments**

tables	A list of ENCODE metadata tables as loaded by fetch_and_clean_raw_ENCODE_tables.
--------	--

**Value**

a data.table containing relevant metadata for all ENCODE files.

**Examples**

```
## Not run:
  tables = fetch_and_clean_raw_ENCODE_tables()
  export_ENCODEdb_matrix(tables = tables)

## End(Not run)
```

---

`generate_encode_df_lite`*Extract file metadata from the full set of ENCODE metadata tables.*

---

**Description**

Extract file metadata from the full set of ENCODE metadata tables.

**Usage**

```
generate_encode_df_lite(tables)
```

**Arguments**

`tables` A list of ENCODE metadata tables as loaded by `fetch_and_clean_raw_ENCODE_tables`.

**Value**

a `data.table` containing relevant metadata for all ENCODE files.

**Examples**

```
## Not run:
  tables = fetch_and_clean_raw_ENCODE_tables()
  export_ENCODEdb_matrix(tables = tables)

## End(Not run)
```

---

`get_encode_types`*A list of known tables from ENCODE database.*

---

**Description**

The type (table) names are extracted from the schema list from ENCODE-DCC github repository: <https://github.com/ENCODE-DCC/encoded/tree/master/src/encoded/schemas>

**Usage**

```
get_encode_types()
```

**Details**

The data is extracted using the github api: <https://developer.github.com/guides/getting-started/>

**Value**

a vector of character with the names of the known tables in the ENCODE database.

**Examples**

```
get_encode_types()
```

---

get_schema_urls	<i>Returns the URLs for downloading the XML schemas from ENCODE's github.</i>
-----------------	---

---

**Description**

Returns the URLs for downloading the XML schemas from ENCODE's github.

**Usage**

```
get_schema_urls()
```

**Value**

a character vector of schema download URLs.

**Examples**

```
ENCODEExplorerData::get_schema_urls()
```

---

get_schemas	<i>Extract the schemas from ENCODE's github</i>
-------------	---

---

**Description**

The JSONs are fetched from: <https://github.com/ENCODE-DCC/encoded/tree/master/src/encoded/schemas>

**Usage**

```
get_schemas()
```

**Details**

The data is extracted using the github api: <https://developer.github.com/guides/getting-started/>  
The data is then downloaded using the jsonlite package.

**Value**

a list of schemas.

**Examples**

```
ENCODEExplorerData::get_schemas()
```

# Index

`clean_table`, [1](#)

`data.table`, [2](#), [3](#)

`encode_df_full`, [1](#), [2](#), [3](#)

`encode_df_lite`, [1](#), [2](#), [3](#)

`ENCODEExplorerData`, [1](#)

`fetch_and_clean_raw_ENCODE_tables`, [1](#), [3](#)

`generate_encode_df_full`, [1](#), [2](#), [4](#)

`generate_encode_df_lite`, [1](#), [3](#), [5](#)

`get_encode_types`, [5](#)

`get_schema_urls`, [6](#)

`get_schemas`, [6](#)