

Package: peekbankr (via r-universe)

September 23, 2026

Type Package

Title Accessing and Working with Peekbank Data

Version 0.4.2

Description Collection of tools for working with peekbank, an open repository for developmental eye-tracking data.

Depends R (>= 4.1.0)

License GPL-3

URL <https://github.com/peekbank/peekbankr>

BugReports <https://github.com/peekbank/peekbankr/issues>

Encoding UTF-8

Imports dplyr (>= 1.0.4), glue (>= 1.4.2), jsonlite (>= 1.7.2),
magrittr (>= 2.0.1), purrr (>= 0.3.4), rlang (>= 0.4.10), tidyr
(>= 1.1.2), stringr (>= 1.5.1), redivis (>= 0.6.0)

Suggests testthat (>= 3.0.0), knitr, rmarkdown

Additional_repositories <https://langcog.r-universe.dev>

RoxygenNote 7.3.3

VignetteBuilder knitr

Config/testthat/edition 3

Repository <https://mikabr.r-universe.dev>

Date/Publication 2026-09-23 00:58:57 UTC

RemoteUrl <https://github.com/langcog/peekbankr>

RemoteRef HEAD

RemoteSha 6a99785c491b3a596865e6175d06c36cefb44566

Contents

append_relative_cdi_scores	2
cleanup_cdi_data	3
connect_to_peekbank	4
download_stimuli	5
ds.compute_aois	6
ds.get_json_fields	6
ds.get_peekjson	7
ds.is_table_required	7
ds.list_coding_methods	8
ds.list_ds_tables	8
ds.list_language_choices	9
ds.map_columns	10
ds.normalize_times	10
ds.resample_times	11
ds.rezero_times	12
ds.validate_for_db_import	12
ds.validate_table	13
ds.validate_trial_uniqueness_constraint	14
file.exists.case.sensitive	15
get_administrations	16
get_aoi_region_sets	16
get_aoi_timepoints	17
get_datasets	18
get_db_info	18
get_readmes	19
get_sql_query	20
get_stimuli	20
get_subjects	21
get_trial_types	22
get_trials	22
get_xy_timepoints	23
list_peekbank_tables	24
populate_cdi_percentiles	24
unpack_aux_data	25
Index	26

append_relative_cdi_scores	<i>Adds a relative cdi score indicating the percentage of total achievable points the subject got on each given measure</i>
----------------------------	---

Description

Adds a relative cdi score indicating the percentage of total achievable points the subject got on each given measure

Usage

```
append_relative_cdi_scores(subjects_table)
```

Arguments

`subjects_table` a subjects table with unnested cdi data, needs columns "subject_id", "language", "instrument_type", "measure", "rawscore"

Value

the input table with an added "cdi_relative" column that contains the percentage of total points gained in the given administrations

Examples

```
## Not run:
cdi_data <- all_subjects %>%
  unnest(subject_aux_data) %>%
  filter(!is.na(cdi_responses)) %>%
  unnest(cdi_responses) %>%
  append_relative_cdi_scores()

## End(Not run)
```

cleanup_cdi_data	<i>Checks cdi data for inconsistencies, warns about them, and fixes them</i>
------------------	--

Description

Checks cdi data for inconsistencies, warns about them, and fixes them

Usage

```
cleanup_cdi_data(cdi_data)
```

Arguments

`cdi_data` a subjects table with unnested cdi data, needs columns "subject_id", "language", "instrument_type", "age", "sex", "measure", "rawscore"

Value

a cleaned up version of the cdi data

Examples

```
## Not run:
clean_cdi_data <- all_subjects %>%
  unnest(subject_aux_data) %>%
  filter(!is.na(cdi_responses)) %>%
  unnest(cdi_responses) %>%
  peekbankr::cleanup_cdi_data()

## End(Not run)
```

connect_to_peekbank	<i>Connect to Peekbank</i>
---------------------	----------------------------

Description

As of peekbankr 0.4, data are retrieved from the versioned peekbank dataset on Redivis rather than a MySQL server. ‘connect_to_peekbank()’ now returns a lightweight handle that pins a database version (Redivis dataset version); it no longer opens a DBI connection. Existing code that passes the result via the ‘connection’ argument of the ‘get_’ functions continues to work.

Usage

```
connect_to_peekbank(
  db_version = "current",
  db_args = NULL,
  compress = TRUE,
  host = NULL,
  port = NULL,
  ssl = "auto"
)
```

Arguments

db_version	String of the name of database version to use: "current" (the default; resolves to the latest Redivis version), a Peekbank release name (e.g. "2025.1"), or a Redivis version tag (e.g. "v1.2")
db_args	Deprecated, ignored (retained for backwards compatibility)
compress	Deprecated, ignored
host	Deprecated, ignored
port	Deprecated, ignored
ssl	Deprecated, ignored

Value

A ‘peekbank_connection’ handle pinning the resolved version

Examples

```
## Not run:
con <- connect_to_peekbank(db_version = "current")

## End(Not run)
```

download_stimuli	<i>Download stimulus images for the Peekbank repository</i>
------------------	---

Description

This function downloads stimulus images for selected Peekbank datasets from the peekbank_files dataset on Redivis. It retrieves stimulus metadata from a Peekbank database connection, constructs the full stimulus file paths, and downloads them to a local directory.

Usage

```
download_stimuli(
  con,
  local_base_dir = "stimulus_data",
  datasets = c(),
  skip_existing = TRUE
)
```

Arguments

con	A version handle created by connect_to_peekbank()
local_base_dir	Local directory path where stimulus images will be saved (default: "stimulus_data")
datasets	Character vector of dataset names to download stimuli for. If empty (default), downloads stimuli for all datasets.
skip_existing	skip downloading a file if a file with that name already exists in that path locally

Value

Returns the stimulus df with an additional column for the paths of the downloaded stimuli

Examples

```
## Not run:
con <- connect_to_peekbank("2025.1")

# Download stimuli for all datasets
download_stimuli(con, local_base_dir = "stimulus_data")

# Download stimuli for specific datasets
download_stimuli(con, local_base_dir = "stimulus_data",
```

```
datasets = c("reflook_v4", "reflook_socword"))

## End(Not run)
```

ds.compute_aois	<i>Classify gaze samples into AOIs based x/y coordinates and provided region sets</i>
-----------------	---

Description

Classify gaze samples into AOIs based x/y coordinates and provided region sets

Usage

```
ds.compute_aois(data)
```

Arguments

data a data frame carrying x, y, target_side, the eight region bounds (l_x_max, l_x_min, l_y_max, l_y_min, r_x_max, r_x_min, r_y_max, r_y_min) and monitor_size_x / monitor_size_y

Value

data with the ao_i column set, overwriting any existing value

ds.get_json_fields	<i>Fetching the list of field names and requirements in each table according to the schema json file</i>
--------------------	--

Description

Fetching the list of field names and requirements in each table according to the schema json file

Usage

```
ds.get_json_fields(table_type)
```

Arguments

table_type the type of dataframe, for the most updated table types specified by schema, please use ds.list_ds_tables()

Value

the list of field names

Examples

```
## Not run:
fields_json <- ds.get_json_fields(table_type = "aoi_timepoints")

## End(Not run)
```

ds.get_peekjson	<i>parse the bundled peekbank schema json into a dataframe</i>
-----------------	--

Description

parse the bundled peekbank schema json into a dataframe

Usage

```
ds.get_peekjson()
```

Value

the organized dataframe from schema json file

Examples

```
## Not run:
peekjson <- ds.get_peekjson()

## End(Not run)
```

ds.is_table_required	<i>Check if a certain table is required according to schema</i>
----------------------	---

Description

Check if a certain table is required according to schema

Usage

```
ds.is_table_required(table_type, coding_methods)
```

Arguments

table_type	the type of dataframe, for the most updated table types specified by schema, please use ds.list_ds_tables()
coding_methods	methods used in the experiment for coding gaze data, to get the list of current coding methods, please use function ds.list_coding_methods()

Value

A boolean value

Examples

```
## Not run:
is_required <- ds.is_table_required(table_type = "xy_timepoints",
                                   coding_method = "manual gaze coding")

## End(Not run)
```

ds.list_coding_methods

Get the coding method list from json schema file

Description

Get the coding method list from json schema file

Usage

```
ds.list_coding_methods()
```

Value

a list of strings indicating allowed coding methods

Examples

```
## Not run:
coding_methods <- ds.list_coding_methods()

## End(Not run)
```

ds.list_ds_tables

List the tables required based on coding method

Description

List the tables required based on coding method

Usage

```
ds.list_ds_tables(coding_methods = c("eyetracking"))
```

Arguments

`coding_methods` a list of strings indicating the methods used in the experiment for coding gaze data, to get the list of current coding methods, please use `ds.list_coding_methods()`

Value

a list of table types that are required based on input coding method

Examples

```
## Not run:  
table_list <- ds.list_ds_tables(coding_methods = "manual gaze coding")  
  
## End(Not run)
```

`ds.list_language_choices`

List current allowed language choices for db import

Description

List current allowed language choices for db import

Usage

```
ds.list_language_choices()
```

Value

a list of strings containing all the allowed language codes based on json schema file

Examples

```
## Not run:  
language_list <- ds.list_language_choices()  
  
## End(Not run)
```

ds.map_columns	<i>Function for mapping raw data columns to processed table columns</i>
----------------	---

Description

Function for mapping raw data columns to processed table columns

Usage

```
ds.map_columns(raw_data, raw_format, table_type)
```

Arguments

raw_data	raw data frame
raw_format	source of the eye-tracking data, e.g. "tobii"
table_type	type of processed table, e.g. "xy_data" "aoi_table"

Value

processed data frame with specified column names

Examples

```
## Not run:
df_xy_data <- ds.map_columns(raw_data = raw_data, raw_format = "tobii",
                             table_type = "xy_data")
df_aoi_data <- ds.map_columns(raw_data = raw_data, raw_format = "tobii",
                              table_type = "aoi_data")

## End(Not run)
```

ds.normalize_times	<i>sets the starting point of a given trial to be zero</i>
--------------------	--

Description

sets the starting point of a given trial to be zero

Usage

```
ds.normalize_times(df_table)
```

Arguments

df_table	to-be-resampled dataframe with t, aoi/xy values, trial_id and administration_id
----------	---

Value

df_out with resampled time, xy or aoi value rows

ds.resample_times	<i>This function resample times to be consistent across labs.</i>
-------------------	---

Description

Resampling is done by the following steps:

Usage

```
ds.resample_times(df_table, table_type)
```

Arguments

df_table	to-be-resampled dataframe with t, aoi/xy values, trial_id and administration_id
table_type	table name, can only be "aoi_timepoints" or "xy_timepoints"

Details

1. iterate through every trial for every administration
2. create desired timepoint sequence with equal spacing according to pre-specified SAMPLE_RATE parameter
3. use approxfun to interpolate given data points to align with desired timepoint sequence "constant" interpolation method is used for AOI timepoints; "linear" interpolation method is used for xy timepoints; for more details on approxfun, please see: <https://stat.ethz.ch/R-manual/R-devel/library/stats/html/approxfun.html>
4. after resampling, bind resampled dataframes back together and re-assign aoi_timepoint_id

Value

df_out with resampled time, xy or aoi value rows

Examples

```
## Not run:
dir_datasets <- "testdataset" # local datasets dir
lab_dataset_id <- "pomper_saffran_2016"
dir_csv <- file.path(dir_datasets, lab_dataset_id, "processed_data")
table_type <- "aoi_timepoints"
file_csv <- file.path(dir_csv, paste0(table_type, '.csv'))
df_table <- utils::read.csv(file_csv)
df_resampled <- ds.resample_times(df_table, table_type = "aoi_timepoints")

## End(Not run)
```

ds.rezero_times	<i>sets the starting point of a given trial to be zero</i>
-----------------	--

Description

sets the starting point of a given trial to be zero

Usage

```
ds.rezero_times(df_table)
```

Arguments

df_table	to-be-resampled dataframe with t, aoi/xy values, trial_id and administration_id
----------	---

Value

df_out with resampled time, xy or aoi value rows

ds.validate_for_db_import	<i>check all csv files against database schema for database import</i>
---------------------------	--

Description

check all csv files against database schema for database import

Usage

```
ds.validate_for_db_import(
  dir_csv,
  cdi_expected,
  file_ext = ".csv",
  is_null_field_required = TRUE,
  suppress_warnings = c()
)
```

Arguments

dir_csv	the folder directory containing all the csv files, the path should end in "processed_data"
cdi_expected	specifies whether cdi_data is to be expected to be present in the imported data
file_ext	the default is ".csv"

is_null_field_required
by default is set to TRUE which means that all the columns in the json file are required; when set to FALSE, fields that are allowed null values are not required

suppress_warnings
character vector of warning IDs to silence. Currently supported: "cdi_collision" (duplicate CDI entries for the same subject/instrument/measure/age/language) and "stimulus_image_filetype" (stimulus images in a format other than jpg/jpeg/png).

Value

A list with two elements:

errors Character vector of validation errors (blocking), or NULL if none.

warnings Character vector of validation warnings (suppressible), or NULL if none. Warnings matching suppress_warnings are excluded.

Examples

```
## Not run:
result <- ds.validate_for_db_import(dir_csv = "./processed_data", cdi_expected = TRUE)
result$errors    # blocking issues
result$warnings  # warnings that can be opted out of on a case by case basis

# suppress known warnings for a specific dataset
result <- ds.validate_for_db_import(dir_csv = "./processed_data", cdi_expected = TRUE,
                                     suppress_warnings = c("cdi_collision"))

## End(Not run)
```

ds.validate_table	<i>Check if a dataframe/table is compliant to peekbank json before database import</i>
-------------------	--

Description

Check if a dataframe/table is compliant to peekbank json before database import

Usage

```
ds.validate_table(
  df_table,
  table_type,
  cdi_expected,
  dir_csv,
  is_null_field_required = TRUE
)
```

Arguments

df_table	the dataframe to be saved
table_type	the type of dataframe, for the most updated table types specified by schema, please use ds.list_ds_tables()
cdi_expected	specifies whether cdi_data is to be expected to be present in the imported data; only relevant for subjects table
dir_csv	the folder directory containing all the csv files, used for stimulus image path validation
is_null_field_required	by default is set to TRUE which means that all the columns in the json file are required; when user specifically sets this to FALSE, then the fields that are allowed null values are not required.

Value

A list with two elements:

errors Character vector of validation errors (blocking), or NULL if none.

warnings Named list of validation warnings (suppressable). Names are warning IDs (e.g. "cdi_collision").

Examples

```
## Not run:
result <- ds.validate_table(df_table = df_table, table_type = "xy_data",
                           cdi_expected = FALSE,
                           dir_csv = "../processed_data")
result$errors    # blocking issues
result$warnings  # warnings that can be opted out of on a case by case basis

## End(Not run)
```

```
ds.validate_trial_uniqueness_constraint
```

Check if within aoi_timepoints table, there is no duplication in all the administration_ids associated with each individual trial_id

Description

Check if within aoi_timepoints table, there is no duplication in all the administration_ids associated with each individual trial_id

Usage

```
ds.validate_trial_uniqueness_constraint(df_aoi_timepoints)
```

Arguments

df_aoi_timepoints
the aoi_timepoints dataframe

Value

an empty string when all the administration_ids are unique within each trial_id; Otherwise, the error message will be returned.

Examples

```
## Not run:  
is_valid <- ds.validate_trial_uniqueness_constraint(df_aoi_timepoints = aoi_timepoints)  
  
## End(Not run)
```

file.exists.case.sensitive

Check if a file exists with exact case sensitivity

Description

Check if a file exists with exact case sensitivity

Usage

```
file.exists.case.sensitive(...)
```

Arguments

... character vectors, containing file paths

Value

logical value: TRUE if the file exists with the exact same case, FALSE otherwise

Examples

```
## Not run:  
exists <- file.exists.case.sensitive("path/to/image.jpg")  
  
## End(Not run)
```

get_administrations	<i>Get administrations</i>
---------------------	----------------------------

Description

Get administrations

Usage

```
get_administrations(  
  age = NULL,  
  dataset_id = NULL,  
  dataset_name = NULL,  
  connection = NULL  
)
```

Arguments

age	A numeric vector of a single age or a min age and max age (inclusive), in months
dataset_id	An integer vector of one or more dataset ids
dataset_name	A character vector of one or more dataset names
connection	A connection to the peekbank database

Value

A tibble of Administrations data, filtered down by supplied arguments.

Examples

```
## Not run:  
get_administrations()  
get_administrations(age = c())  
get_administrations(dataset_name = "pomper_saffran_2016")  
  
## End(Not run)
```

get_aoi_region_sets	<i>Get AOI region sets</i>
---------------------	----------------------------

Description

Get AOI region sets

Usage

```
get_aoi_region_sets(connection = NULL)
```

Arguments

connection A connection to the peekbank database

Value

A tibble of AOI Region Sets data, filtered down by supplied arguments.

Examples

```
## Not run:
get_aoi_region_sets()

## End(Not run)
```

get_aoi_timepoints	<i>Get AOI timepoints</i>
--------------------	---------------------------

Description

Get AOI timepoints

Usage

```
get_aoi_timepoints(
  dataset_id = NULL,
  dataset_name = NULL,
  age = NULL,
  rle = TRUE,
  connection = NULL
)
```

Arguments

dataset_id An integer vector of one or more dataset ids
dataset_name A character vector of one or more dataset names
age A numeric vector of a single age or a min age and max age (inclusive), in months
rle Logical indicating whether to use RLE data representation or not
connection A connection to the peekbank database

Value

A tibble of AOI Timepoints data, filtered down by supplied arguments.

Examples

```
## Not run:
get_aoi_timepoints(dataset_name = "pomper_saffran_2016")

## End(Not run)
```

`get_datasets`*Get datasets*

Description

Get datasets

Usage

```
get_datasets(connection = NULL)
```

Arguments

`connection` A connection to the peekbank database

Value

A tibble of Datasets data.

Examples

```
## Not run:  
get_datasets()  
  
## End(Not run)
```

`get_db_info`*Get information on database connection options*

Description

As of peekbankr 0.4, data are retrieved from the versioned peekbank dataset on Redivis (<https://redivis.com/datapages/datasets/peekbank>) rather than a MySQL server. Version information is discovered from the Redivis dataset itself: each Redivis version carries a ‘release_info’ table naming the Peekbank release it holds.

Usage

```
get_db_info()
```

Value

List of database info: ‘current’ (the Peekbank release that is the latest Redivis version) and ‘versions’ (a named vector mapping Peekbank release names to Redivis dataset version tags)

Examples

```
## Not run:  
get_db_info()  
  
## End(Not run)
```

get_readmes	<i>Download dataset README files</i>
-------------	--------------------------------------

Description

Downloads README files for Peekbank datasets from the peekbank_files dataset on Redivis. Note that READMEs reflect the latest released version of the files dataset.

Usage

```
get_readmes(datasets = c(), local_base_dir = "dataset_readmes")
```

Arguments

datasets	Character vector of dataset names. If empty (default), downloads READMEs for all datasets.
local_base_dir	Directory to save README files to (default: "dataset_readmes")

Value

No return value, called for side effects. README files are saved to the specified directory and the path is printed via message.

Examples

```
## Not run:  
get_readmes()  
get_readmes(datasets = c("pomper_saffran_2016"))  
  
## End(Not run)
```

get_sql_query	<i>Run a SQL query against peekbank</i>
---------------	---

Description

Queries are BigQuery Standard SQL, run against the peekbank dataset on Redivis. Note that string comparison is case-sensitive.

Usage

```
get_sql_query(sql_query_string, connection = NULL)
```

Arguments

sql_query_string	A valid SQL query string
connection	A connection to the peekbank database

Value

A tibble of the query results

Examples

```
## Not run:
con <- connect_to_peekbank()
get_sql_query("SELECT * FROM datasets", connection = con)

## End(Not run)
```

get_stimuli	<i>Get stimuli</i>
-------------	--------------------

Description

Get stimuli

Usage

```
get_stimuli(dataset_id = NULL, dataset_name = NULL, connection = NULL)
```

Arguments

dataset_id	An integer vector of one or more dataset ids
dataset_name	A character vector of one or more dataset names
connection	A connection to the peekbank database

Value

A tibble of Stimuli data, filtered down by supplied arguments.

Examples

```
## Not run:  
get_stimuli()  
get_stimuli(dataset_name = "pomper_saffran_2016")  
  
## End(Not run)
```

get_subjects	<i>Get subjects</i>
--------------	---------------------

Description

Get subjects

Usage

```
get_subjects(connection = NULL)
```

Arguments

connection A connection to the peekbank database

Value

A tibble of Subjects data. Note that Subjects is a table used to link longitudinal Administrations, which is the primary table you probably want.

Examples

```
## Not run:  
get_subjects()  
  
## End(Not run)
```

get_trial_types	<i>Get trial types</i>
-----------------	------------------------

Description

Get trial types

Usage

```
get_trial_types(dataset_id = NULL, dataset_name = NULL, connection = NULL)
```

Arguments

dataset_id	An integer vector of one or more dataset ids
dataset_name	A character vector of one or more dataset names
connection	A connection to the peekbank database

Value

A tibble of Trial Types data, filtered down by supplied arguments.

Examples

```
## Not run:  
get_trial_types()  
get_trial_types(dataset_name = "pomper_saffran_2016")  
  
## End(Not run)
```

get_trials	<i>Get trials</i>
------------	-------------------

Description

Get trials

Usage

```
get_trials(dataset_id = NULL, dataset_name = NULL, connection = NULL)
```

Arguments

dataset_id	An integer vector of one or more dataset ids
dataset_name	A character vector of one or more dataset names
connection	A connection to the peekbank database

Value

A tibble of Trials data, filtered down by supplied arguments.

Examples

```
## Not run:
get_trials()
get_trials(dataset_name = "pomper_saffran_2016")

## End(Not run)
```

get_xy_timepoints	<i>Get XY timepoints</i>
-------------------	--------------------------

Description

Get XY timepoints

Usage

```
get_xy_timepoints(
  dataset_id = NULL,
  dataset_name = NULL,
  age = NULL,
  connection = NULL
)
```

Arguments

dataset_id	An integer vector of one or more dataset ids
dataset_name	A character vector of one or more dataset names
age	A numeric vector of a single age or a min age and max age (inclusive), in months
connection	A connection to the peekbank database

Value

A tibble of XY timepoints data, filtered down by supplied arguments.

Examples

```
## Not run:
get_xy_timepoints(dataset_name = "reflook_v4")

## End(Not run)
```

`list_peekbank_tables` *List of peekbank tables*

Description

List of peekbank tables

Usage

```
list_peekbank_tables(connection)
```

Arguments

`connection` A connection to the peekbank database

Value

A vector of the names of tables in peekbank

Examples

```
## Not run:
con <- connect_to_peekbank()
list_peekbank_tables(con)

## End(Not run)
```

`populate_cdi_percentiles`

Populate the provided cdi data with percentile values for that specific age, instrument_type, measure and language. Loosely based on the work from this repo <https://github.com/kachergis/cdi-percentiles/tree/main> by George Kachergis and Jess Mankewitz with advice from Virginia Marchman.

Description

Populate the provided cdi data with percentile values for that specific age, instrument_type, measure and language. Loosely based on the work from this repo <https://github.com/kachergis/cdi-percentiles/tree/main> by George Kachergis and Jess Mankewitz with advice from Virginia Marchman.

Usage

```
populate_cdi_percentiles(subjects_table)
```

Arguments

subjects_table a subjects table with unnested cdi data, needs columns "subject_id", "language", "instrument_type", "age", "sex", "measure", "rawscore"

Value

the input table with added columns containing the reference age used, the reference year used, and both gender specific and general percentile values for the cdi score

Examples

```
## Not run:
full_cdi_data <- all_subjects %>%
  unnest(subject_aux_data) %>%
  filter(!is.na(cdi_responses)) %>%
  unnest(cdi_responses) %>%
  peekbankr::cleanup_cdi_data() %>%
  peekbankr::populate_cdi_percentiles()

## End(Not run)
```

unpack_aux_data	<i>Unpack the json sting in the *_aux_data column and turns it into a nested R list</i>
-----------------	---

Description

Unpack the json sting in the *_aux_data column and turns it into a nested R list

Usage

```
unpack_aux_data(df)
```

Arguments

df a dataframe in the peekbank format that has an aux data column

Value

the input dataframe, with the *_aux_data column unpacked

Examples

```
## Not run:
subjects_table <- unpack_aux_data(df = subjects_table)

## End(Not run)
```

Index

`append_relative_cdi_scores`, [2](#)

`cleanup_cdi_data`, [3](#)

`connect_to_peekbank`, [4](#)

`download_stimuli`, [5](#)

`ds.compute_aois`, [6](#)

`ds.get_json_fields`, [6](#)

`ds.get_peekjson`, [7](#)

`ds.is_table_required`, [7](#)

`ds.list_coding_methods`, [8](#)

`ds.list_ds_tables`, [8](#)

`ds.list_language_choices`, [9](#)

`ds.map_columns`, [10](#)

`ds.normalize_times`, [10](#)

`ds.resample_times`, [11](#)

`ds.rezero_times`, [12](#)

`ds.validate_for_db_import`, [12](#)

`ds.validate_table`, [13](#)

`ds.validate_trial_uniqueness_constraint`,
[14](#)

`file.exists.case.sensitive`, [15](#)

`get_administrations`, [16](#)

`get_aoi_region_sets`, [16](#)

`get_aoi_timepoints`, [17](#)

`get_datasets`, [18](#)

`get_db_info`, [18](#)

`get_readmes`, [19](#)

`get_sql_query`, [20](#)

`get_stimuli`, [20](#)

`get_subjects`, [21](#)

`get_trial_types`, [22](#)

`get_trials`, [22](#)

`get_xy_timepoints`, [23](#)

`list_peekbank_tables`, [24](#)

`populate_cdi_percentiles`, [24](#)

`unpack_aux_data`, [25](#)