cvtdbLoad

July 21, 2025

Title Load CvTdb Data

Version 1.0

Description Functions and custom scripts to run workflows to load CvTdb manually curated data into the CvTdb database. This includes normalization, unit conversion, and QC data upload workflows.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Suggests knitr, rmarkdown

VignetteBuilder knitr

Imports DBI (>= 1.2.3),
downloader (>= 0.4.1),
dplyr (>= 1.1.4),
httk (>= 2.6.0),
httr (>= 1.4.7),
magrittr (>= 2.0.3),
mgsub (>= 1.7.3),
purrr (>= 1.0.4),
readr (>= 2.1.5),
readxl (>= 1.4.5),
renv (>= 1.1.4),
RPostgres (>= 1.4.8),
RSQLite (>= 2.3.9),
stats (>= 4.4.2),
stringr (>= 1.5.1),
tidyr (>= 1.3.1),
tools (>= 4.4.2),
utils (>= 4.4.2),
validate (>= 1.1.5),
writexl (>= 1.5.2)

Contents

`check_convert_failed` *check_convert_failed*

Description

Check if a conversion failed, resulting in an NA value. This can be due to a lack of necessary equation parameters (e.g., molecular weight, subject weight).

Usage

```
check_convert_failed(x, f, col, log_path, id_col = "id")
```

Arguments

`x`	Input list of datasets being processed.
`f`	Optional filename for logging purposes.
`col`	String of name of column to check.
`log_path`	File path where to save the log file.
`id_col`	Column to use to log index value to help with logging.

Value

Modified input x dataframe list with new "convert_failed" dataframe.

See Also

filter

check_empty_sheet *check_empty_sheet*

Description

Function to check if any sheet of loaded template is empty.

Usage

```
check_empty_sheet(template)
```

Arguments

template	List of loaded template sheets.

Value

Boolean TRUE or FALSE if an empty sheet exists.

See Also

is_empty

check_missing *check_missing*

Description

Function to check for missing values for input column.

Usage

```
check_missing(x, miss_col, f, flag = TRUE, log_path)
```

Arguments

`x`	Input list of datasets being processed.
`miss_col`	String of the name of the column to check.
`f`	Optional filename for logging purposes.
`flag`	Whether to log a flag for the missing column.
`log_path`	File path where to save the log file.

Value

Modified input `x` dataframe list with new "missing" dataframe.

See Also

filter

`check_missing_units` *check_missing_units*

Description

Function to check for missing units for input column.

Usage

```
check_missing_units(x, f, units_col, log_path, flag = TRUE)
```

Arguments

`x`	Input list of datasets being processed.
`f`	Optional filename for logging purposes.
`units_col`	String of the name of the column to check.
`log_path`	File path where to save the log file.
`flag`	Whether to log a flag for the missing column.

Value

Modified input `x` dataframe list with new "missing_units" dataframe.

See Also

filter

check_non_numeric *check_non_numeric*

Description

Function to check for non-numeric values for input column.

Usage

```
check_non_numeric(x, f, col, log_path)
```

Arguments

`x`	Input list of datasets being processed.
`f`	Optional filename for logging purposes.
`col`	String of the name of the column to check.
`log_path`	File path where to save the log file.

Value

Modified input `x` dataframe list with new "non_numeric" dataframe.

See Also

mutate, filter, select

check_required_fields *check_required_fields*

Description

Function to check if processed template is missing required fields.

Usage

```
check_required_fields(df, f)
```

Arguments

`df`	List of dataframes for the sheets within an extraction template.
`f`	Optional filename for logging purposes.

Value

None. Logs any flags.

See Also

filter

`check_subject_list` *check_subject_list*

Description

Function to check if an input field has a ; separated list, therefore should be split.

Usage

```
check_subject_list(x, f, col, log_path)
```

Arguments

`x`	Input list of datasets being processed.
`f`	Optional filename for logging purposes.
`col`	String of the name of the column to check.
`log_path`	File path where to save the log file.

Value

Modified input `x` dataframe list with new "split_subject" dataframe.

See Also

filter

`check_unit_ci` *check_unit_ci*

Description

Function to check for confidence intervals for an input field.

Usage

```
check_unit_ci(x, f, col, log_path)
```

Arguments

`x`	Input list of datasets being processed.
`f`	Optional filename for logging purposes.
`col`	String of the name of the column to check.
`log_path`	PARAM_DESCRIPTION

Value

Modified input x dataframe list with new "ci" dataframe.

See Also

filter, mutate, across all_of

check_unit_range *check_unit_range*

Description

Function to check for ranges for input column.

Usage

```
check_unit_range(x, f, col, log_path)
```

Arguments

x	Input list of datasets being processed.
f	Optional filename for logging purposes.
col	String of the name of the column to check.
log_path	File path where to save the log file.

Value

Modified input x dataframe list with new "unit_range" dataframe.

See Also

filter, mutate, rowwise, across, select separate, all_of

`clowder_get_dataset_files` *clowder_get_dataset_files*

Description

Pull a dataframe of file metadata stored in a Clowder dataset.

Usage

```
clowder_get_dataset_files(dsID, baseurl, apiKey)
```

Arguments

`dsID`	Clowder dataset identifier.
`baseurl`	Clowder base URL.
`apiKey`	Clowder API key.

Value

Dataframe with Clowder file identifier, folder name, and file name.

See Also

`GET`, `content_type`, `add_headers`, `content` `unnest` `bind_rows`, `select`

`clowder_get_dataset_folders` *FUNCTION_TITLE*

Description

Pull a dataframe of folder metadata stored in a Clowder dataset.

Usage

```
clowder_get_dataset_folders(dsID, baseurl, apiKey)
```

Arguments

`dsID`	Clowder dataset identifier.
`baseurl`	Clowder base URL.
`apiKey`	Clowder API key.

Value

Dataframe with Clowder folder identifier and folder name.

See Also

`GET`, `content_type`, `add_headers`, `content` `unnest` `bind_rows`, `select`

`clowder_get_file_metadata`

clowder_get_file_metadata

Description

Pull a dataframe of folder metadata stored in a Clowder dataset.

Usage

```
clowder_get_file_metadata(fileID, baseurl, apiKey)
```

Arguments

`fileID`	Vector of Clowder file identifiers.
`baseurl`	Clowder base URL.
`apiKey`	Clowder API key.

Value

Dataframe with Clowder file metadata.

See Also

`GET`, `content_type`, `add_headers`, `content` `pluck`, `keep` `unnest` `mutate`, `across`, `reexports`, `bind_cols`, `bind_rows`

`clowder_match_docs` *clowder_match_docs*

Description

Function to match template documents to Clowder files.

Usage

```
clowder_match_docs(
  df = NULL,
  dsID = NULL,
  baseurl = NULL,
  apiKey = NULL,
  clowder_file_list = NULL
)
```

Arguments

`df`	Input template document's sheet for mapping.
`dsID`	Clowder dataset ID to pull from.
`baseurl`	Clowder base URL.
`apiKey`	API key to access Clowder repo.
`clowder_file_list`	Opitonal input list of Clowder files from clowder_get_dataset_files().

Value

Modified `df` with clowder_file_id field for matched documents.

See Also

filter, mutate, left_join, arrange

`clowder_match_post_upload` *clowder_match_post_upload*

Description

This is a helper function to match already loaded CvT document entries to Clowder docs.

Usage

```
clowder_match_post_upload(dsID = NULL, baseurl = NULL, apiKey = NULL)
```

Arguments

dsID	Clowder dataset identifier.
baseurl	Clowder base URL.
apiKey	Clowder API key.

Value

None. An update query is performed to update the Documents table "clowder_file_id" field.

See Also

filter, select

clowder_upload_file_metadata

clowder_upload_file_metadata

Description

Function to process add metadata field values to Clowder files.

Usage

```
clowder_upload_file_metadata(metadata, dsID, userID, baseurl, apiKey)
```

Arguments

metadata	Dataframe of metadata to add to Clowder files.
dsID	Clowder dataset identifier.
userID	Clowder User identifier.
baseurl	Clowder base URL.
apiKey	Clowder API key.

Value

None. Clowder APi calls are performed to add metadata to Clowder files.

See Also

`unite`, `separate` `mutate-joins`, `rename`, `select`, `filter` `flatten` `POST`, `content_type`, `add_headers`

`convert_cols_to_NA` *convert_cols_to_NA*

Description

A function to convert input dataframe column list to NA.

Usage

```
convert_cols_to_NA(df, col_list)
```

Arguments

`df`	Input dataframe.
`col_list`	String of name of column to convert to NA.

Value

Modified input `df` dataframe with `col_list` as NA.

`convert_get_conversion_factor` *convert_get_conversion_factor*

Description

A helper function to convert input values to desired units.

Usage

```
convert_get_conversion_factor(conv_factor = 1)
```

Arguments

`conv_factor`	Conversion factor to use (such as Molecular weight, tissue Density, etc.). Default of 1.

Value

List of conversion factors

convert_units *convert_units*

Description

Function to convert input values to desired units.

Usage

```
convert_units(
  x,
  num,
  units,
  desired,
  conv_factor = NA,
  overwrite_units = FALSE
)
```

Arguments

`x`	Input dataframe to convert.
`num`	Name of column with values to convert.
`units`	Name of column with units to convert from.
`desired`	Desired units to convert the input value into.
`conv_factor`	Conversion factor to use (such as Molecular weight, tissue Density, etc.).
`overwrite_units`	Boolean to overwrite the ’units’ with desired units.

Value

Modified dataframe of input `x` dataframe with converted column using `convert_get_conversion_factor()`.

convert_units_grepl *convert_units_grepl*

Description

Function to get various grepl statements for unit name standardization.

Usage

```
convert_units_grepl(unit_type)
```

Arguments

`unit_type` Input unit type (e.g., weight, height, age, dose_duration, conc)

Value

List of unit name standardizations by input unit type.

`db_connect_to_CvT` *db_connect_to_CvT*

Description

A function to create a connection to the CvT database using the .Renviron file parameters.

Usage

```
db_connect_to_CvT()
```

Value

A database connection object.

See Also

dbConnect, PostgreSQL

`db_get_tbl_id` *db_get_tbl_id*

Description

A function to pull table fk identification from a specified table by a SQL filter statement, or the entire table.

Usage

```
db_get_tbl_id(tblName = NULL, idFilter = NULL)
```

Arguments

`tblName` The name of the table to pull the ID from.

`idFilter` A SQL WHERE statement to filter idName column to. If empty, pulls all data.

Value

A list of ID values from the specified database table.

See Also

dbSendQuery, dbClearResult, dbDisconnect dbFetch

`db_push_tbl_to_db` *db_push_tbl_to_db*

Description

A function to push a dataframe to a specified table in the database.

Usage

```
db_push_tbl_to_db(
  dat = NULL,
  tblName = NULL,
  fieldTypes = NULL,
  overwrite = FALSE,
  customSQL = NULL,
  append = FALSE
)
```

Arguments

`dat` A dataframe to write to the database, Default: NULL

`tblName` Name of database table to create or append and fill with input `dat` dataframe data, Default: NULL

`fieldTypes` Named list of field types for columns, Default: NULL

`overwrite` Boolean of whether to overwrite the `tblName` table with input `dat` dataframe data, Default: FALSE

`customSQL` Optional custom SQL statement to push, Default: NULL

`append` Boolean of whether to append the `tblName` table with input `dat` dataframe data, Default: FALSE

Value

None. Updates are pushed to the database.

See Also

dbWriteTable, dbSendQuery, dbDisconnect

`db_query_cvt` *db_query_cvt*

Description

Function to query or send statements to the database. Handles errors/warnings with tryCatch.

Usage

```
db_query_cvt(query = NULL, query_type = "query")
```

Arguments

`query`	A SQL query string.
`query_type`	Input string specifying if the query is a SQL "query" (retrieve data) or "statement" (update database). Default: 'query'.

Value

Return objects from DBI `dbGetQuery()` and `dbSendStatement()` functions.

See Also

dbGetQuery, dbSendStatement, dbDisconnect

`db_update_tbl` *db_update_tbl*

Description

A function to update database table entries based on dataframe

Usage

```
db_update_tbl(df = NULL, tblName = NULL)
```

Arguments

`df`	A dataframe to write to the database, Default: NULL
`tblName`	Name of database table to create or append and fill with input `df` dataframe data, Default: NULL

Value

None. Updates are pushed to the database.

See Also

dbWriteTable, dbClearResult, dbDisconnect dbSendStatement

`download_jira_update_clowder_info`

download_jira_update_clowder_info

Description

Function to generate a dataframe of Jira ticket metadata with attachment information. Can be used to download the attachments locally and upload metadata to Clowder files once they have been uploaded to Clowder separately.

Usage

```
download_jira_update_clowder_info(
  jira_project,
  in_file = NULL,
  auth_token,
  reset_attachments = FALSE,
  update_clowder_metadata = FALSE,
  dsID,
  baseurl,
  userID,
  apiKey,
  labels_filter = NULL,
  epic_filter = c(),
  attachment_filter = c()
)
```

Arguments

`jira_project` Jira project identifier (e.g., CVTDB).

`in_file` Optional param for the file path to previously pulled Jira information CSV file.

`auth_token` Jira API token.

`reset_attachments`
Boolean whether to re-download Jira ticket attachments. Default: FALSE.

`update_clowder_metadata`
Boolean whether to update Clowder file metadata. Only set to TRUE once Jira ticket attachments have been uploaded to Clowder. Default: FALSE.

`dsID` Clowder dataset identifier.

`baseurl` Clowder base URL.

`userID` Clowder user identifier.

`apiKey` Clowder API key.

`labels_filter` Vector list of Jira ticket labels to filter to.

`epic_filter` Custom filtering to a specific ticket Epic link by name (single or vector). Default: empty vector.

`attachment_filter`
Filename regex string vector to filter to select Jira ticket attachments, Default: empty vector.

Value

Dataframe of metadata from Jira tickets to associate to Clowder files.

See Also

`filter`, `select`, `mutate` `unite` `View`, `download.file`

`extract_units` *extract_units*

Description

Generic function to extract units from input columns. The goal is to try to fill in missing unit values from `units_col` based on string values in `conv_col`.

Usage

```
extract_units(x, units_col, conv_col, unit_type)
```

Arguments

`x` Input dataframe.

`units_col` String of the name of the column containing units information with missing values.

`conv_col` String of the name of the column to try to extract unit information from.

`unit_type` Input unit type (e.g., weight, height, age, dose_duration, conc), used by `convert_units_grepl()`.

Value

Modified version of input `x` dataframe where the input `units_col` will have filled in units values from input `conv_col` or be "missing_units" tagged instead of NA.

See Also

filter, mutate, across, bind_rows all_of

`get.dictionary.entries.to.curate` *get.dictionary.entries.to.curate*

Description

Function to generate file of dictionary entries to curate normalized field values. Includes foreign key identifiers to help query database.

Usage

```
get.dictionary.entries.to.curate(schema, full.report = FALSE)
```

Arguments

`schema`	Database schema for PostgreSQL.
`full.report`	Boolean of whether to generate a full report and include foreign key connections to provide additional context, Default: FALSE.

Value

Dataframe list of dictionaries.

See Also

`flatten`, `keep` `write_xlsx`

`get_cvtdb_sqlite` *get_cvtdb_sqlite*

Description

Pull CvTdb tables and write sqlite file.

Usage

```
get_cvtdb_sqlite(schema, outdir)
```

Arguments

`schema`	Database postgreSQL schema.
`outdir`	String for desired output directory.

Value

None. SQLite file is written to `outdir` folder.

See Also

pull dbConnect, dbWriteTable, dbDisconnect SQLite

get_cvt_push_ready *get_cvt_push_ready*

Description

Function to return a dataframe of files ready to push due to all flags being '0'.

Usage

```
get_cvt_push_ready()
```

Value

Dataframe filtered from the template normalization log of templates without logged issues.

See Also

read_xlsx filter, across

get_cvt_template *get_cvt_template*

Description

Load the CvT template file into a list of empty dataframes.

Usage

```
get_cvt_template(template_path)
```

Arguments

`template_path`	Path to the CvT template file.

Value

Named list of template file dataframes.

See Also

excel_sheets, read_xlsx

`get_dict_update_ids` *get_dict_update_ids*

Description

Update database dictionaries and match foreign keys to input identifiers.

Usage

```
get_dict_update_ids(sheet_list, schema)
```

Arguments

`sheet_list`	Dataframe list of template sheets with dictionary columns to process and receive database dictionary table ID matches.
`schema`	String for the PostgreSQL schema information to pull.

Value

Named list of dataframes with modified columns mapping input identifiers to database identifiers if present. If not present, new database entries are created and the ID returned.

See Also

`rename`, `filter`, `reexports`, `select`, `distinct`, `mutate`, `across`, `mutate-joins` `unite`

`get_mw_chemicals_api` *get_mw_chemicals_api*

Description

Function to get molecular weight dictionary from CCTE Chemicals API in batches.

Usage

```
get_mw_chemicals_api(dtxsid_list, api_key)
```

Arguments

`dtxsid_list`	List of DTXSIDs to query.
`api_key`	API key for the CCTE Chemicals API.

Value

Dataframe of molecular weight information by DTXSID.

See Also

GET, content, POST, content_type, add_headers bind_rows, select, filter

`get_next_tbl_id` *get_next_tbl_id*

Description

Function to get the next ID increment value for schema tables.

Usage

```
get_next_tbl_id(schema)
```

Arguments

schema	PostgreSQL database schema.

Value

Named list of the next autoincrement identifier for a database table.

See Also

pull

`jira_download_templates` *jira_download_templates*

Description

Function to download Jira ticket attachments.

Usage

```
jira_download_templates(in_data, auth_token)
```

Arguments

in_data	Input dataframe with Jira ticket attachment information.
auth_token	Jira API token.

Value

None. Jira API calls are made to download files to the "output" folder.

See Also

unite mutate download.file, unzip

load_file_from_api *load_file_from_api*

Description

Function to load a CSV or XLSX file as a dataframe or list of dataframes using an API call. Typically, this is to load a CvTdb template file from Clowder.

Usage

```
load_file_from_api(url, headers, file_type, mode = "w")
```

Arguments

url	URL where the file will be downloaded.
headers	Optional headers to complete the download call.
file_type	String of the file extension. Currently only supports "csv" and "xlsx".
mode	Download file mode, Default: 'w' for "write".

Value

Dataframe or list of dataframes from a file downloaded from a URL.

See Also

download.file read_delim, cols mutate, across str_trim excel_sheets, read_excel

load_sheet_group *load_sheet_group*

Description

A function to load an input template and map it to the provided template format. It corrects for missing required column names from a template file by filling with NA values.

Usage

```
load_sheet_group(fileName = "", template_path = "")
```

Arguments

`fileName`	The file name or path for the file of interest.
`template_path`	The file path for the extraction template. If not supplied, hard coded columns will be used.

Value

A named list of dataframes that fits the format of the input template file.

See Also

excel_sheets, read_excel, read_xlsx select all_of

`log_CvT_doc_load` *log_CvT_doc_load*

Description

Function to write a log entry for a template or dataframe being processed.

Usage

```
log_CvT_doc_load(
  f,
  m = NULL,
  reset = FALSE,
  val = NULL,
  log_path = "output/template_normalization_log.xlsx"
)
```

Arguments

`f`	Filename to flag.
`m`	Log field name, Default: NULL.
`reset`	Boolean to reset a row's flags, Default: FALSE.
`val`	Custom field value. Default: NULL.
`log_path`	File path where to save the log file. Default "output/template_normalization_log.xlsx".

Value

None. Log file is updated.

See Also

read_xlsx setNames write_xlsx

`map_age_category` *map_age_category*

Description

Function to use a standard age category dictionary file to map species age to an age category.

Usage

```
map_age_category(x, dict)
```

Arguments

`x`	Input dataframe with species and age value/units fields.
`dict`	Input dataframe with species age category assignment fields.

Value

Modified input `x` dataframe with age_category field filled in where possible.

See Also

filter, select

`match_cvt_doc_to_db_doc` *match_cvt_doc_to_db_doc*

Description

Function to match input document metadata to CvTdb document records by pmid, other_study_identifier, doi, url hierarchy.

Usage

```
match_cvt_doc_to_db_doc(df = NULL)
```

Arguments

`df`	Input dataframe of Document records to try to match to database Documents table entries, Default: NULL.

Value

Modified input `df` dataframe with matched database Documents table ID values.

See Also

select, tidyeval-compat, filter, distinct, pull, mutate, across, reexports, mutate-joins, bind_rows, arrange str_trim

normalization_prep *normalization_prep*

Description

Helper function to add temp ID column and new empty columns before normalization.

Usage

```
normalization_prep(x, newcols)
```

Arguments

x	Input dataframe being prepped for normalization.
newcols	String vector of column names to add with NA values.

Value

Modified version of the input x parameter

normalize_age *normalize_age*

Description

A helper function to normalize dose.

Usage

```
normalize_age(raw, f, log_path, debug = FALSE)
```

Arguments

raw	Input dataframe of data with data to normalize.
f	Optional filename for logging purposes.
log_path	File path where to save the log file.
debug	Boolean of whether to stop conversion logic early for debugging purpose. Default: FALSE.

Value

Normalized version of the input `raw` parameter.

See Also

read_xlsx mutate, filter, select, bind_rows, rename, arrange

`normalize_age_units` *normalize_age_units*

Description

Helper function to normalize age unit names.

Usage

```
normalize_age_units(x)
```

Arguments

`x`	Input vector of conc units.

Value

Modified input `x` vector with normalized age unit names.

`normalize_boolean` *normalize_boolean*

Description

FUNCTION_DESCRIPTION

Usage

```
normalize_boolean(x, col)
```

Arguments

`x`	Input dataframe
`col`	Stirng vector of fields to update boolean values to "0" or "1".

Value

Modified dataframe `x` with boolean fields updated to "0" or "1".

normalize_conc *normalize_conc*

Description

A helper function to normalize dose.

Usage

```
normalize_conc(raw, f, log_path, debug = FALSE)
```

Arguments

`raw`	Input dataframe of data with data to normalize.
`f`	Optional filename for logging purposes.
`log_path`	File path where to save the log file.
`debug`	Boolean of whether to stop conversion logic early for debugging purpose. Default: FALSE.

Value

Normalized version of the input `raw` parameter.

See Also

mutate, filter, across, bind_rows, arrange POST

normalize_conc_units *normalize_conc_units*

Description

Helper function to normalize concentration unit names.

Usage

```
normalize_conc_units(x)
```

Arguments

`x`	Input vector of conc units.

Value

Modified input `x` vector with normalized concentration unit names.

normalize_dose *normalize_dose*

Description

A helper function to normalize dose.

Usage

```
normalize_dose(raw, f, log_path, debug = FALSE)
```

Arguments

`raw`	Input dataframe of data with data to normalize.
`f`	Optional filename for logging purposes.
`log_path`	File path where to save the log file.
`debug`	Boolean of whether to stop conversion logic early for debugging purpose. Default: FALSE.

Value

Normalized version of the input `raw` parameter.

See Also

filter, mutate, bind_rows, arrange, select get_physchem_param

normalize_dose_duration *normalize_dose_duration*

Description

Extraction of units from dose_duration field (similar to height/weight).

Usage

```
normalize_dose_duration(raw, f)
```

Arguments

`raw`	A dataframe of weight information to normalize.
`f`	The file name of the template being processed. Used for error logging.

Value

Normalized version of the input `raw` parameter.

See Also

mutate, bind_rows, arrange, select

`normalize_height` *normalize_height*

Description

A function to normalize subject height.

Usage

```
normalize_height(raw, f, log_path, debug = FALSE)
```

Arguments

`raw`	Input dataframe of data with data to normalize.
`f`	Optional filename for logging purposes.
`log_path`	File path where to save the log file.
`debug`	Boolean of whether to stop conversion logic early for debugging purpose. Default: FALSE.

Value

Normalized version of the input `raw` parameter.

See Also

mutate, filter, bind_rows, arrange

normalize_species *normalize_species*

Description

Normalize species field for input dataframe.

Usage

```
normalize_species(x, log_path = NULL)
```

Arguments

x	Dataframe with species information to normalize.
log_path	File path where to save the log file.

Value

Modified input x dataframe with normalized species information.

normalize_time *normalize_time*

Description

Normalize time to hours for input dataframe.

Usage

```
normalize_time(raw, f, log_path, debug = FALSE)
```

Arguments

raw	Input dataframe of data with data to normalize.
f	Optional filename for logging purposes.
log_path	File path where to save the log file.
debug	Boolean of whether to stop conversion logic early for debugging purpose. Default: FALSE.

Value

Modified dataframe of raw with normalized values.

See Also

mutate, bind_rows, arrange

`normalize_time_units` *normalize_time_units*

Description

Function to normalize time unit names to a standard format.

Usage

```
normalize_time_units(x)
```

Arguments

`x`	Input vector of time units.

Value

Modified `x` vector with normalized time unit names.

`normalize_weight` *normalize_weight*

Description

A helper function to normalize dose.

Usage

```
normalize_weight(raw, f, log_path, debug = FALSE)
```

Arguments

`raw`	Input dataframe of data with data to normalize.
`f`	Optional filename for logging purposes.
`log_path`	File path where to save the log file.
`debug`	Boolean of whether to stop conversion logic early for debugging purpose. Default: FALSE.

Value

Normalized version of the input `raw` parameter.

See Also

mutate, filter, bind_rows, arrange, select `pull_clowder_files_to_load`

pull_clowder_files_to_load

Description

Function to pull Clowder templates to process based on metadata "cvt_to_load"

Usage

```
pull_clowder_files_to_load(
  dsID,
  baseurl,
  apiKey,
  curation_set_tag,
  metadata_filter_tag = NULL
)
```

Arguments

`dsID`	Clowder dataset identifier.
`baseurl`	Clowder base URL.
`apiKey`	Clowder API token.
`curation_set_tag`	String curation dataset tag to filter pulled files to.
`metadata_filter_tag`	String vector of metadata fields to filter/select to for pulled files.

Value

Dataframe of Clowder file metadata.

See Also

`mutate`, `rename`, `mutate-joins`, `filter`, `select`, `bind_rows`, `distinct` `separate`, `pivot_longer`

`pull_jira_info` *pull_jira_info*

Description

Script to process CSV export of Jira into a status log with attachment metadata.

Usage

```
pull_jira_info(
  jira_project,
  in_file = NULL,
  auth_token = NULL,
  status_filter = "Done",
  epic_filter = c()
)
```

Arguments

`jira_project`	Jira project code (e.g. CVTDB).
`in_file`	Filepath to previously downloaded CSV file summary of Jira tickets.
`auth_token`	Authorization token for Jira.
`status_filter`	Custom filtering to a ticket status or vector of statuses. Default: NULL.
`epic_filter`	Custom filtering to a specific ticket Epic link by name (single or vector). Default: empty vector.

Value

Named list of dataframes that summarize Jira tickets and attachment metadata.

See Also

download.file, unzip read_csv, cols select, contains, mutate, everything, filter, distinct, left_join, group_by, summarise, n unite str_squish

`qc_add_record` *qc_add_record*

Description

Function to add record from QC template.

Usage

```
qc_add_record(df, tbl_field_list, load_doc_sheet_only, col_exclude)
```

Arguments

`df` Input dataframe of field values to add to database table.

`tbl_field_list` Dataframe of CvT tables and fields.

`load_doc_sheet_only` Boolean whether just to add document sheet only.

`col_exclude` List of columns to exclude from database pushes.

Value

None. SQL statements are submitted to add a record to the database tables.

See Also

`keep` `filter`, `select`, `reexports`, `mutate`, `case_when`

`qc_remove_record` *qc_remove_record*

Description

Function to remove QC record flagged for removal. Function will account for needed cascade of removal for foreign key table connections

Usage

```
qc_remove_record(df, tbl_name, reset_extraction = FALSE, del_qc_note = NULL)
```

Arguments

`df` Input dataframe of id, qc_notes, and qc_flags for records to remove

`tbl_name` Name of table the records are from

`reset_extraction` Boolean whether to reset_extraction or remove whole record. Defaul FALSE.

`del_qc_note` Optional note to add to qc_notes and qc_flags field for tbl_name records.

Value

None. SQL statements are run to delete records in a cascading fashion if they have foreign key linkages to core tables.

See Also

`keep` `mutate`

`reorganize_file_flags` *reorganize_file_flags*

Description

A function to check if a file has logged issues (changed to 1 for select columns) and move it to appropriate subfolder.

Usage

```
reorganize_file_flags()
```

Value

None. File are reorganized in output directory.

See Also

read_xlsx

`save_normalized_template` *save_normalized_template*

Description

A helper function to cache the normalized templates during the normalization workflow.

Usage

```
save_normalized_template(df, f)
```

Arguments

`df`	Named list of dataframes of normalized template data.
`f`	Filename of the input template that is modified to save a normalized copy.

Value

None. Input dataframe is saved to output folder.

See Also

file_ext write_xlsx

`set_original_fields` *set_original_fields*

Description

Pull dictionary of original fields from database tables and rename in `sheet_list`.

Usage

```
set_original_fields(sheet_list, schema)
```

Arguments

`sheet_list`	Dataframe list to rename with original columns.
`schema`	String for the PostgreSQL schema information to pull.

Value

Modified `sheet_list` list of dataframes with "_original" fields added.

See Also

`filter`, `select`, `mutate`, `distinct`, `rename`, `reexports`

`species_get_unique_to_curate` *species_get_unique_to_curate*

Description

Load a list of templates or query the database to check for species cases to add to `normalize_species()`.

Usage

```
species_get_unique_to_curate(fileList, template_path)
```

Arguments

`fileList`	Optional list of template files to load. If NULL, queries the database to check for cases.
`template_path`	Path to input CvTdb template file.

Value

Console output of species cases to add to `normalize_species()`.

See Also

select

`update.dictionary.entries.from.file`
update.dictionary.entries.from.file

Description

Function to update dictionary entries from file generated from get.dictionary.entries.to.curate().

Usage

```
## S3 method for class 'dictionary.entries.from.file'
update(schema, in_file)
```

Arguments

`schema`	Database schema for PostgreSQL.
`in_file`	Input file with dictionary updates to process.

Value

Dataframe list of dictionaries.

See Also

`excel_sheets` `select`, `reexports`

`validate_cvt` *validate_cvt*

Description

Function to validate a CvTdb template based on a ruleset.

Usage

```
validate_cvt(
  clowder_file_id = NULL,
  clowder_api_key = NULL,
  file_path = NULL,
  db_identifier = NULL,
  df = NULL,
  df_identifier = NULL,
  log_path = "output/validation/validate_cvt_log.xlsx",
  ignore_present = FALSE,
  ignore_nonempty = FALSE,
  ignore_required = FALSE,
  ignore_qc = FALSE,
  ignore_field_types = FALSE,
  ignore_field_entries = FALSE,
  ignore_field_uniqueness = FALSE,
  ignore_foreign_keys = FALSE,
  ignore_expected_fields = FALSE,
  verbose = FALSE
)
```

Arguments

`clowder_file_id`
Clowder template file identifier, Default: NULL

`clowder_api_key`
Clowder API token, Default: NULL

`file_path` Path to template file, Default: NULL

`db_identifier` String or numeric identifier for a record in the database Documents table, Default: NULL

`df` Input named list of dataframes, Default: NULL

`df_identifier` String identifier for input `df`, required if `df` is not NULL, Default: NULL

`log_path` Path to log, Default: 'output/validation/validate_cvt_log.xlsx'

`ignore_present` Boolean of whether to ignore `validate_sheets_present()`, Default: FALSE

`ignore_nonempty`
Boolean of whether to ignore `validate_sheets_nonempty()`, Default: FALSE

`ignore_required`
Boolean of whether to ignore `validate_required_fields()`, Default: FALSE

`ignore_qc` Boolean of whether to ignore `validate_qc_fields()` and/or use the QC template for validation, Default: FALSE

`ignore_field_types`
Boolean of whether to ignore `validate_field_types()`, Default: FALSE

`ignore_field_entries`
Boolean of whether to ignore `validate_field_entries()`, Default: FALSE

`ignore_field_uniqueness`	Boolean of whether to ignore `validate_field_uniqueness()`, Default: FALSE
`ignore_foreign_keys`	Boolean of whether to ignore `validate_foreign_keys()`, Default: FALSE
`ignore_expected_fields`	Boolean of whether to ignore `validate_expected_fields()`, Default: FALSE
`verbose`	Boolean of whether to print additional console messages, Default: FALSE.

Value

Boolean of whether the input template passed the validation.

See Also

`mutate`

`validate_expected_fields` *validate_expected_fields*

Description

Function to check if processed document's column names appropriately match the blank template.

Usage

```
validate_expected_fields(df, f, log_path, template_path, verbose = FALSE)
```

Arguments

`df`	Input named list of dataframes.
`f`	Filename for logging purposes.
`log_path`	Path to log.
`template_path`	Path to blank template file for comparison.
`verbose`	Boolean of whether to print additional console messages, Default: FALSE.

Value

Boolean of whether the input template passed the validation.

See Also

validator

validate_field_entries *validate_field_entries*

Description

Function to check if processed document has invalid field entries.

Usage

```
validate_field_entries(df, f, log_path, verbose = FALSE)
```

Arguments

df	Input named list of dataframes.
f	Filename for logging purposes.
log_path	Path to log.
verbose	Boolean of whether to print additional console messages, Default: FALSE.

Value

Boolean of whether the input template passed the validation.

See Also

validator

validate_field_types *validate_field_types*

Description

Function to check if processed document has invalid field types.

Usage

```
validate_field_types(df, f, log_path, verbose = FALSE)
```

Arguments

df	Input named list of dataframes.
f	Filename for logging purposes.
log_path	Path to log.
verbose	Boolean of whether to print additional console messages, Default: FALSE.

Value

Boolean of whether the input template passed the validation.

See Also

validator

---

validate_field_uniqueness

validate_field_uniqueness

Description

Function to check if processed document has unique fields where applicable.

Usage

```
validate_field_uniqueness(df, f, log_path, verbose = FALSE)
```

Arguments

`df`	Input named list of dataframes.
`f`	Filename for logging purposes.
`log_path`	Path to log.
`verbose`	Boolean of whether to print additional console messages, Default: FALSE.

Value

Boolean of whether the input template passed the validation.

See Also

validator

validate_foreign_keys *validate_foreign_keys*

Description

Function to check if processed document's foreign keys match to a value in their respective sheets.

Usage

```
validate_foreign_keys(df, f, log_path, verbose = FALSE)
```

Arguments

df	Input named list of dataframes.
f	Filename for logging purposes.
log_path	Path to log.
verbose	Boolean of whether to print additional console messages, Default: FALSE.

Value

Boolean of whether the input template passed the validation.

See Also

validator

validate_qc_fields *validate_qc_fields*

Description

Function to check if processed QC template was filled out appropriately.

Usage

```
validate_qc_fields(df, f, log_path, verbose = FALSE)
```

Arguments

df	Input named list of dataframes.
f	Filename for logging purposes.
log_path	Path to log.
verbose	Boolean of whether to print additional console messages, Default: FALSE.

Value

Boolean of whether the input template passed the validation.

See Also

`validator`, `confront`, `summary`, `meta`, `satisfying filter`

---

validate_required_fields

validate_required_fields

Description

Function to check if processed document is missing required fields.

Usage

```
validate_required_fields(df, f, log_path, verbose = FALSE)
```

Arguments

`df`	Input named list of dataframes.
`f`	Filename for logging purposes.
`log_path`	Path to log.
`verbose`	Boolean of whether to print additional console messages, Default: FALSE.

Value

Boolean of whether the input template passed the validation.

See Also

validator

`validate_sheets_nonempty` *validate_sheets_nonempty*

Description

Function to check if processed template contains expected non-empty sheets.

Usage

```
validate_sheets_nonempty(df, f, log_path)
```

Arguments

`df`	Input named list of dataframes.
`f`	Filename for logging purposes.
`log_path`	Path to log.

Value

Boolean of whether the input template passed the validation.

`validate_sheets_present` *validate_sheets_present*

Description

Validate if template contains expected sheets

Usage

```
validate_sheets_present(df, f, log_path)
```

Arguments

`df`	Input named list of dataframes.
`f`	Filename for logging purposes.
`log_path`	Path to log.

Value

Boolean of whether the input template passed the validation.

%>% *Pipe operator*

Description

See `magrittr::%>%` for details.

Usage

```
lhs %>% rhs
```

Arguments

`lhs`	A value or the magrittr placeholder.
`rhs`	A function call using the magrittr semantics.

Value

The result of calling `rhs(lhs)`.

Index