Package: ohcleandat (via r-universe)

September 13, 2024

Type Package

Title One Health Data Cleaning and Quality Checking Package

Version 0.3.11

Maintainer Collin Schwantes <schwantes@ecohealthalliance.org>

Description This package provides useful functions to orchestrate analytics and data cleaning pipelines for One Health projects.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2.9000

Suggests knitr, rmarkdown

VignetteBuilder knitr

Imports arsenal, containerTemplateUtils (>= 0.0.0.9006), dplyr, frictionless, fs, googledrive, googlesheets4, here, purrr, rdrop2 (>= 0.8.2.2), readr, readxl, rlang, ruODK (>= 1.4.9.9002), stringi, stringr, tibble, tidyr, tidyselect, utils, validate

Remotes ecohealthalliance/containerTemplateUtils, fcampelo/rdrop2, ropensci/ruODK

URL https://ecohealthalliance.github.io/ohcleandat/

Depends R (>= 2.10)

Roxygen list(markdown = TRUE)

Repository https://ecohealthalliance.r-universe.dev

RemoteUrl https://github.com/ecohealthalliance/ohcleandat

RemoteRef main

RemoteSha 1113e40b471aa25553c183ee1eda3f61631ab5e1

Contents

`autobot` *Autobot Function*

Description

This compares two columns. Where there are differences, it extracts the values and compiles a correctly formatted validation log. This is intended to be used when an automated formatting correction is proposed in the data, but the actual updating of the records is required to happen via the validation log.

Usage

```
autobot(data, old_col, new_col, key)
```

Arguments

data	data.frame or tibble
old_col	The existing column with formatting issues
new_col	The new column with corrections applied
key	column that uniquely identifies the records in data

Value

tibble formatted as validation log

---

check_id_existence *Check existence of ID columns across two tables*

Description

This returns rows in x without a match in y. Returning selected columns only. It is a this wrapper around `dplyr::anti_join`.

Usage

```
check_id_existence(x, y, by, select_cols, ...)
```

Arguments

x	data.frame or tibble containing match id to check for non existence in y
y	data.frame or tibble to check for non-existence of match id from x
by	character containing match id, or if named different, a named character vector like c("a" = "b")
select_cols	character vector of columns to select in the output. Note that during the join, columns with identical names in both data sets will have a suffix of .x or .y added to disambiguate. These need to be added to ensur the correct column is returned.
...	other variables passed to dplyr::anti_join

Value

tibble rows from x without a match in y

See Also

`dplyr::anti_join`

Examples

```
## Not run:
check_id_existence(x,
                   y,
                   by =  c("Batch_ID" = "batch_id"),
                   select_cols = c("Batch_ID", "iDate", "Farm_ID"))

## End(Not run)
```

`class_to_col_type` *Class to Column Type lookup table*

Description

A table that links classes to `readr` column types. Created from csv file of the same name in inst/

Usage

```
class_to_col_type
```

Format

`class_to_col_type`**:**

A data frame with 9 rows and 3 columns:

col_type Type of column as described in `readr`

col_class Class of R object that matches that column type

col_abv Abbreviation for that column type from `readr` ...

Details

class_to_col_type <- read.csv(file = "inst/class_to_col_type.csv") usethis::use_data(class_to_col_type,overwrite = TRUE)

See Also

`readr::cols()`

`combine_logs` *Combine Validation Logs*

Description

Checks for the existence of an existing validation log and appends new records from the current run.

Usage

```
combine_logs(existing_log, new_log)
```

Arguments

`existing_log`	tibble existing validation log
`new_log`	tibble newly generated validation log

Value

tibble appended validation log for upload

`correct_data` *Correct data using validation log*

Description

Takes a validation log and applies the required changes to the data

Usage

```
correct_data(validation_log, data, primary_key)
```

Arguments

`validation_log`	tibble a validation log
`data`	tibble the original unclean data
`primary_key`	character the quoted column name for the unique identifier in data

Value

tibble the semi-clean data set

`create_freetext_log` *Create Free Text Log*

Description

Creates custom validation log for 'other: explain' free text responses that may contain valid multi-choice options.

Usage

```
create_freetext_log(response_data, form_schema, url, lookup)
```

Arguments

`response_data`	data.frame ODK questionnaire response data
`form_schema`	data.frame ODK flattened form schema data
`url`	The ODK submission URL excluding the uuid identifier
`lookup`	a tibble formatted as a lookup to match questions with their free text responses. The format must match the output of `othertext_lookup()`. This function can be passed to this function argument as a convenient handler for this value.

Details

This function needs to link a survey question with its corresponding free text response. Users can use the `othertext_lookup()` function to handle this, or provide their own tibble in the same format. See below: tibble::tribble(~name, ~other_name, question_1, question_1_other)

Value

data.frame validation log

See Also

`othertext_lookup()`

Examples

```
## Not run:
# Using othertext_lookup helper
test_a <- create_freetext_log(response_data = animal_owner_semiclean,
                              form_schema = animal_owner_schema,
                      url = "https://odk.xyz.io/#/projects/5/forms/project/submissions",
                    lookup = ohcleandat::othertext_lookup(questionnaire = "animal_owner")
                              )

# using custom lookup table
mylookup <- tibble::tribble(
  ~name, ~other_name,
```

```
  "f2_species_own", "f2a_species_own_oexp"
  )

  test_b <- create_freetext_log(response_data = animal_owner_semiclean,
                                form_schema = animal_owner_schema,
                      url = "https://odk.xyz.io/#/projects/5/forms/project/submissions",
                                lookup = mylookup
                                )

## End(Not run)
```

`create_questionnaire_log` *Create Validation Log for Questionnaire data*

Description

Create Validation Log for Questionnaire data

Usage

```
create_questionnaire_log(data, form_schema, pkey, rule_set, url)
```

Arguments

`data`	data fame Input data to be validated
`form_schema`	data frame The ODK form schema data
`pkey`	character A character vector giving the column name of the primary key or unique row identifier in the data
`rule_set`	a rule set of class validator from the validate package
`url`	The ODK submission URL excluding the uuid identifier

Value

a data frame formatted as a validation log for human review

create_rules_from_template *Create a "rules" file from a template*

Description

Creates a rules file from a template to show general structure of the rule file.

Usage

```
create_rules_from_template(
  name,
  dir = "R",
  open = TRUE,
  showWarnings = FALSE,
  overwrite_file = FALSE
)
```

Arguments

name	String. Name of rule set function e.g. create_rules_my_dataset
dir	String. Name of directory where file should be created. If it doesnt exist, a folder will be created.
open	Logical. Should the file be opened?
showWarnings	Logical. Should dir.create show warnings?
overwrite_file	Logical. Should a rules file with the same name be overwritten?

Value

String. File path of newly created file

Examples

```
## Not run:
# create a ruleset and immediately open it
    create_rules_from_template(name = "create_rules_field_data")
# create a ruleset and don't open it
    create_rules_from_template(name = "create_rules_lab_data", open = FALSE)
# create a ruleset and store it in a different folder
    create_rules_from_template(name = "create_rules_lab_data",
    dir = "/path/to/rulesets" open = FALSE)

## End(Not run)
```

`create_structural_metadata` *Create Structural Metadata from a dataframe*

Description

This is the metadata that describes the data themselves. This metadata can be generated then joined to pre-existing metadata via field names.

Usage

```
create_structural_metadata(
  data,
  primary_key = "",
  foreign_key = "",
  additional_elements = tibble::tibble()
)
```

Arguments

`data`	Any named object. Expects a table but will work superficially with lists or named vectors.
`primary_key`	Character. name of field that serves as a primary key
`foreign_key`	Character. Field or fields that are foreign keys
`additional_elements`	Empty tibble with structural metadata elements and their types.

Details

The metadata table produced has the following elements

`name` = The name of the field. This is taken as is from `data`. `description` = Description of that field. May be provided by controlled vocabulary `units` = Units of measure for that field. May or may not apply `term_uri` = Universal Resource Identifier for a term from a controlled vocabulary or schema `comments` = Free text providing additional details about the field `primary_key` = `TRUE` or `FALSE`, Uniquely identifies each record in the data `foreign_key` = `TRUE` or `FALSE`, Allows for linkages between data sets. Uniquely identifies records in a different data set

Value

dataframe with standard metadata requirements

Examples

```
## Not run:
df <- data.frame(a = 1:10, b = letters[1:10])
df_metadata  <- ohcleandat::create_structural_metadata(df)
write.csv(df_metadata,"df_metadata.csv")
```

```
Additional elements can be added via a tibble
additional_elements <- tibble::tibble(table_name = NA_character_,
created_by = NA_character_,
updated = NA
)
df_metadata  <- ohcleandat::create_structural_metadata(df,
    additional_elements = additional_elements)

# lets pretend we are using a dataset which already has
## in airtable, you can add field descriptions directly
## in the base. We want those exported and properly formatted
## in our ohcleandat workflow

 base <- "appMyBaseID"
 table_name <- "My Table"

 airtable_metadata  <- airtabler::air_generate_metadata_from_api(base = base,
    field_names_to_snake_case = FALSE ) |>
    dplyr::filter(table_name == {table_name}) |>
    dplyr::select(field_name,field_desc,primary_key)

 airtable_df <- airtabler::fetch_all(base = base, table_name = table_name)

 airtable_df_metadata <- ohcleandat::create_structural_metadata(airtable_df)

 metadata_joined <- dplyr::left_join(airtable_df_metadata,airtable_metadata,
 by = c("name"="field_name"))

 metdata_updated <- metadata_joined |>
 dplyr::mutate(description = field_desc,
               primary_key = primary_key.y,
               ) |>
 dplyr::select(-matches('\\.[xy]|field_desc'))

# ODK
# get all choices from ODK form

dotenv::load_dot_env()

ruODK::ru_setup(
  svc = "https://odk.server.org/v1/projects/5/forms/myproject.svc",
  un = Sys.getenv("ODK_USERNAME"),
  pw = Sys.getenv("ODK_PASSWORD"),
  tz = "GMT",
  odkc_version = "1.1.2")


schema <- ruODK::form_schema_ext()

schema$choices_flat <-schema$`choices_english_(en)` |>
  purrr::map_chr(\(x){
```

```
    if("labels" %in% names(x)){
      paste(x$labels,collapse = ", ")
    } else {
      ""
    }

  })

  data_odk <- ruODK::odata_submission_get()
  data_odk_rect <- ruODK::odata_submission_rectangle(data_odk)
  odk_metadata <- ohcleandat::create_structural_metadata(data_odk_rect)


  odk_metadata_joined  <- dplyr::left_join(odk_metadata,schema_simple,
  by = c("name" = "ruodk_name"))

  odk_metadata_choices <- odk_metadata_joined |>
  mutate(description = choices_flat) |>
  select(-choices_flat)



## End(Not run)
```

`create_translation_log` *Create Translation Log*

Description

Collates free text responses from 'other' and 'notes' fields in the survey data. Some language detection is performed and placed in the log notes section for possible translation.

Usage

```
create_translation_log(response_data, form_schema, url)
```

Arguments

`response_data`	data.frame of ODK questionnaire responses
`form_schema`	data.frame or flattened ODK form schema
`url`	The ODK submission URL excluding the uuid identifier

Value

data.frame validation log

Examples

```
## Not run:
create_translation_log(response_data = semi_clean_data,
                       form_schema = odk_schema_data,
                       url = "https://odk.xyz.io/#/projects/project-name/submissions"))

## End(Not run)
```

`create_validation_log` *Create Validation Log*

Description

Create Validation Log

Usage

```
create_validation_log(data, pkey, rule_set, ...)
```

Arguments

`data`	data fame Input data to be validated
`pkey`	character a character vector giving the column name of the primary key or unique row identifier in the data
`rule_set`	a rule set of class validator from the validate package
`...`	other arguments passed to validate::confront

Value

a data frame formatted as a validation log for human review

`detect_language` *Detect Language*

Description

A function that extracts the top guess of the language of a piece of text.

Usage

```
detect_language(text)
```

Arguments

`text`	character any text string

Details

Utilizes the stringi package encoding detector as the means to infer language.

Value

character estimate for language abbreviation

See Also

`stringi::stri_enc_detect()`

Examples

```
detect_language(text = "buongiorno")
```

---

download_dropbox *Download Drop Box Files*

Description

Downloads files from dropbox into a given directory

Usage

```
download_dropbox(dropbox_path, dropbox_filename, download_path, ...)
```

Arguments

`dropbox_path`	character The formal folder path on dropbox
`dropbox_filename`	character The formal file name on dropbox
`download_path`	character Local file path to download file to
`...`	other arguments passed to rdrop2::drop_download

Value

returns file path if successful

See Also

`rdrop2::drop_download()`

Examples

```
## Not run:
   download_dropbox(dropbox_path = "XYZ/Project-Datasets",
   dropbox_filename = "Project dataset as at 01-02-2024.xlsx",
   download_path = here::here("data"),
   overwrite = TRUE)

## End(Not run)
```

`download_googledrive_files` *Download Google Drive Files*

Description

For a given Google Drive folder this function will find and download all files matching a given pattern.

Usage

```
download_googledrive_files(
  key_path,
  drive_path,
  search_pattern,
  MIME_type = NULL,
  out_path
)
```

Arguments

`key_path`	character path to Google authentication key
`drive_path`	character The Google drive folder path
`search_pattern`	character A search pattern for files in the Google drive
`MIME_type`	character Google Drive file type, file extension, or MIME type.
`out_path`	character The local file directory for files to be downloaded to

Details

Note: This relies on the `googledrive::drive_ls()` function which uses a search function and is not deterministic when recursively searching. Please pay attention to what is returned.

Value

a character vector of files downloaded

See Also

googledrive::drive_ls()

Examples

```
## Not run:
  download_googledrive_files(
  key_path = here::here("./key.json"),
  drive_path = "https://drive.google.com/drive/u/0/folders/asdjfnasiffas8ef7y7y89rf",
  search_pattern = ".*\\.xlsx",
  out_path = here::here("data/project_data/")
  )

## End(Not run)
```

---

dropbox_upload *Dropbox Upload*

Description

Upload a local file to dropbox and handle authentication. Automatically zips files over 300mb by default.

Usage

```
dropbox_upload(log, file_path, dropbox_path, compress = TRUE)
```

Arguments

log	dataframe. Validation Log for OH cleaning pipelines. Will work with any tabular data.
file_path	character. local file path for upload
dropbox_path	character. relative dropbox path
compress	logical. Should files over 300mb be compressed?

Details

This is a wrapper of rdrop2::drop_upload() which first reads in a local CSV file and then uploads to a DropBox path.

Value

performs drop box upload

Examples

```
## Not run:
    dropbox_upload(
    kzn_animal_ship_semiclean,
    file_path = here::here("outputs/data.csv"),
    dropbox_path = "XYZ/Data/semi_clean_data"
    )

## End(Not run)
```

`expand_frictionless_metadata` *Expand Frictionless Metadata with structural metadata*

Description

Loops over elements in the structural metadata and adds them to frictionless metadata schema. Will overwrite existing values.

Usage

```
expand_frictionless_metadata(
  structural_metadata,
  resource_name,
  resource_path,
  data_package_path,
  prune_datapackage = TRUE
)
```

Arguments

`structural_metadata`	Dataframe. Structural metadata from `create_structural_metadata` or `update_structural_metadata`
`resource_name`	Character. Item within the datapackage to be updated
`resource_path`	Character. Path to csv file
`data_package_path`	Character. Path to datapackage.json file
`prune_datapackage`	Logical. Should properties not in the structural metadata be removed?

Value

Updates the datapackage, returns nothing

Examples

```
## Not run:

# read in file
data_path <- "my/data.csv"
data <- read.csv(data_path)

# create structural metadata
data_codebook  <- create_structural_metadata(data)

# update structural metadata
write.csv(data_codebook,"my/codebook.csv", row.names = FALSE)

data_codebook_updated <- read.csv(""my/codebook.csv"")

# create frictionless package - this is done automatically with the
# deposits package
my_package <-
 create_package() |>
 add_resource(resource_name = "data", data = data_path)

 write_package(my_package,"my")

expand_frictionless_metadata(structural_metadata = data_codebook_updated,
                             resource_name = "data",
                             resource_path = data_path,
                             data_package_path = "my/datapackage.json"
                             )


## End(Not run)
```

---

get_dropbox_val_logs *Get Dropbox Validation Logs*

Description

Downloads existing validation logs that are stored on dropbox

Usage

```
get_dropbox_val_logs(file_name, folder, path_name)
```

Arguments

file_name	character file name with extension of the validation log. Note that file may have been zipped on upload if its over 300mb. This file will be automatically unzipped on download so provide the file extenstion for the compressed file, not the zipped file. E.g. "val_log.csv" even if on dropbox its stored as "val_log.zip".

folder	character the folder the log is saved in on drop box. Can be NULL if not in subfolder.
path_name	character the default drop box path

Details

This function will check if the log exists and return NULL if not. Else it will locally download the file to 'dropbox_validations' directory and read in to the session.

Value

tibble a Validation Log

Examples

```
## Not run:
 get_dropbox_val_logs(file_name = "log.csv", folder = NULL)

## End(Not run)
```

---

get_odk_form_schema *Get ODK Questionnaire Schema Info*

Description

This function handles the authentication and pulling of questionnaire form schema information.

Usage

```
get_odk_form_schema(
  url,
  un = Sys.getenv("ODK_USERNAME"),
  pw = Sys.getenv("ODK_PASSWORD"),
  odkc_version = Sys.getenv("ODKC_VERSION")
)
```

Arguments

url	character The survey URL
un	character The ODK account username
pw	character The ODK account password
odkc_version	character The ODKC Version string

Details

This is a wrapper around the ruODK package. It handles the setup and authentication. See https://github.com/ropensci/ruODK

Value

data frame of survey responses

See Also

ruODK::form_schema_ext()

Examples

```
## Not run:
    get_odk_form_schema(url ="https://odk.xyz.io/v1/projects/5/forms/survey.svc",
    un = Sys.getenv("ODK_USERNAME"),
    pw = Sys.getenv("ODK_PASSWORD"),
    odkc_version = Sys.getenv("ODKC_VERSION"))

## End(Not run)
```

---

get_odk_responses *Get ODK Questionnaire Response Data*

Description

This function handles the authentication and pulling of responses data for ODK Questionnaires. The raw return list is 'rectangularized' into a data frame first. See the ruODK package for more info on how this happens.

Usage

```
get_odk_responses(
  url,
  un = Sys.getenv("ODK_USERNAME"),
  pw = Sys.getenv("ODK_PASSWORD"),
  odkc_version = Sys.getenv("ODKC_VERSION")
)
```

Arguments

url	character The survey URL
un	character The ODK account username
pw	character The ODK account password
odkc_version	character The ODK version

Details

This is a wrapper around the ruODK package. It handles the setup and authentication. See https://github.com/ropensci/ruODK

Value

data.frame of flattened survey responses

See Also

ruODK::form_schema_ext()

Examples

```
## Not run:
    get_odk_responses(url ="https://odk.xyz.io/v1/projects/5/forms/survey.svc",
    un = Sys.getenv("ODK_USERNAME"),
    pw = Sys.getenv("ODK_PASSWORD"),
    odkc_version = Sys.getenv("ODKC_VERSION"))

## End(Not run)
```

---

get_precision *Get Precision*

Description

Get Precision

Usage

```
get_precision(x, func = c, ...)
```

Arguments

x	Numeric. Vector of gps points
func	Function. Apply some function to the vector of precisions. Default is c so that all values are returned
...	Additional arguments to pass to func.

Value

output of func - likely a vector

Author(s)

Nathan Layman

Examples

```
x <- c(1,100,1.11)
get_precision(x,func = min)
```

get_species_letter *Get Species Letter*

Description

This function maps the relationship between animal species and hum_anim_id codes. This is for use in id_checker()

Usage

```
get_species_letter(
  species = c("human", "cattle", "small_mammal", "sheep", "goat")
)
```

Arguments

species	character The species identifier. See argument options

Value

character The hum_anim_id code

guess_col_type *Guess the column type*

Description

uses column class to set readr column type

Usage

```
guess_col_type(data, default_col_abv = "c")
```

Arguments

data	data.frame Data who column types you would like to guess
default_col_abv	string. Column type abbreviation from readr::cols(). Use "g" to guess the column type.

Value

character vector of column abbreviations

Examples

```
data <- data.frame(time = Sys.time(),
char = "hello", num = 1, log = TRUE,
date = Sys.Date(), list_col = list("hello") )

guess_col_type(data)

## change default value of default column abbreviation

guess_col_type(data, default_col_abv = "g")
```

`id_checker` *ID Checker*

Description

General function for checking and correcting ID columns.

Usage

```
id_checker(col, type = c("animal", "hum_anim", "site"), ...)
```

Arguments

`col`	The vector of ID's to be checked
`type`	The ID type, see argument options for allowable settings
`...`	other function arguments passed to get_species_letter

Details

In order to use the autobot process for correcting ID columns, a new 'corrected' column is created by the user using the id_checker() function. It will take an existing vector of ID's, and an ID type (animal, mosquito, etc) and apply the bespoke corrections. This can then be consumed by the autobot log.

Value

vector of corrected ID's

Examples

```
## Not run:
# with a species identifier
   data |> mutate(animal_id_new = id_checker(animal_id, type = "animal", species = "cattle"))
    data |> mutate(farm_id_new = id_checker(farm_id, type = "site"))

## End(Not run)
```

make_report_urls *Make the URLs for the reports*

Description

Several HTML reports are emailed via an automated process. To do this a secure URL is to be generated as a download link. This function is to be used in an opinionated targets pipeline.

Usage

```
make_report_urls(aws_deploy_target, pattern = "")
```

Arguments

aws_deploy_target	List. Output from aws_s3_upload
pattern	String. Regex pattern for matching file paths

Value

character URL for report

Author(s)

Collin Schwantes

make_zip_path *Get make a zip file path*

Description

Take a file path, remove the extension, replace the extension with .zip

Usage

```
make_zip_path(file_path)
```

Arguments

file_path	character.

Value

character. String where extension is replaced by zip

Examples

```
file_path <- "hello.csv"
make_zip_path(file_path)

file_path_with_dir <- "foo/bar/hello.csv"
make_zip_path(file_path_with_dir)
```

`obfuscate_gps` *Obfuscate GPS*

Description

This function fuzzes gps points by first adding error then rounding to a certain number of digits.

Usage

```
obfuscate_gps(
  x,
  precision = 2,
  fuzz = 0.125,
  type = c("lat", "lon"),
  func = min,
  ...
)

obfuscate_lat(x, precision = 2, fuzz = 0.125)

obfuscate_lon(x, precision = 2, fuzz = 0.125)
```

Arguments

`x`	Numeric. Vector of gps points
`precision`	Integer. Number of digits to keep. See `round` for more details
`fuzz`	Numeric. Positive number indicating how much error to introduce to the gps measurements. This is used to generate the random uniform distribution `runif(1,min = -fuzz, max = fuzz)`
`type`	Character. One of "lat" or "lon"
`func`	Function. Function used in `get_precision`
`...`	Additional arguments for func.

Value

Numeric. A vector of fuzzed and rounded GPS points

Numeric vector

Numeric vector

Examples

```
# make data
gps_data  <- data.frame(lat = c(1.0001, 10.22223, 4.00588),
                        lon = c(2.39595, 4.506930, -60.09999901))

# Default obfuscation settings correspont to roughly a 27 by 27 km area
gps_data$lat |>
  obfuscate_gps(type = "lat")

# Obfuscation can be made more or less precise by changing the number of
# decimal points included or modifying the amount of fuzz (error)
# introduced
gps_data$lon |>
  obfuscate_gps(precision = 4, fuzz = 0.002, type = "lon")

### working at the poles
gps_data_poles  <- data.frame(lat = c(89.0001, 89.22223, -89.8881),
                              lon = c(2.39595, 4.506930, -60.09999901))


gps_data_poles$lat |>
  obfuscate_gps(fuzz = 1, type = "lat")


### working at the 180th meridian
gps_data_180  <- data.frame(lat = c(2, 3, 4),
                            lon = c(179.39595, -179.506930, -178.09999901))
gps_data_180$lon |>
  obfuscate_gps(fuzz = 1, type = "lon")

### working NA GPS data
gps_data_180  <- data.frame(lat = c(2, 3, 4),
                            lon = c(179.39595, NA, -178.09999901))
gps_data_180$lon |>
  obfuscate_gps(fuzz = 1, type = "lon")

### GPS is on the fritz!
## Not run:
gps_data_fritz <- data.frame(lat = c(91, -91, 90),
                             lon = c(181.0001, -181.9877, -178.09999901))
gps_data_fritz$lon |>
  obfuscate_gps(fuzz = 1, type = "lon")

gps_data_fritz$lat |>
  obfuscate_gps(fuzz = 1, type = "lat")

## End(Not run)
```

`othertext_lookup` *Look-up table for 'Other' questions*

Description

Provides a look up table matching ODK survey questions with their free text response question.

Usage

```
othertext_lookup(questionnaire = c("animal_owner"))
```

Arguments

questionnaire	The ODK questionnaire. Used to ensure the correct look up table is found.

Details

In many ODK surveys, a multiple choice question can have a response for 'other' where the respondent can add free text as a response. There is no consistent link in the response data to match the captured responses and the other free-text collected. This function provides a manual look up reference so free text responses can be compared to the original questions in the validation workflow.

This function can be expanded by providing a tibble with two columns: `name` and `other_name` which maps the question name in ODK to the question name containing 'other' or 'free text'.

Value

tibble

Examples

```
othertext_lookup(questionnaire = c("animal_owner"))
```

---

prune_datapackage *Prune data pacakge*

Description

method to remove properties from the metadata for a dataset in a datapackage

Usage

```
prune_datapackage(my_data_schema, structural_metadata)
```

Arguments

my_data_schema	list. schema object from frictionless
structural_metadata	dataframe. structural metadata for a dataset

Value

pruned data_schema -

`read_excel_all_sheets` *Reads all tabs from an excel workbook*

Description

For a given excel file, this will detect all sheets, and iteratively read all sheets and place them in a list.

If primary keys are added, the primary key is the triplet of the file, sheet name, and row number e.g. "file_xlsx_sheet1_1". Row numbering is based on the data ingested into R. R automatically skips empty rows at the beginning of the spreadsheet so id 1 in the primary key will belong to the first row with data.

Usage

```
read_excel_all_sheets(
  file,
  add_primary_key_field = FALSE,
  primary_key = "primary_key"
)
```

Arguments

`file`	character. File path to an excel file
`add_primary_key_field`	Logical. Should a primary key field be added?
`primary_key`	character. The column name for the unique identifier to be added to the data.

Value

list

Note

The primary key method is possible because Excel forces sheet names to be unique.

Examples

```
 ## Not run:
# Adding primary key field
read_excel_all_sheet(file = "test_pk.xlsx",add_primary_key_field = TRUE)

# Don't add primary key field
read_excel_all_sheet(file = "test_pk.xlsx")
```

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

`read_googlesheets` *Read Google Sheets Data*

Description

For a given sheet id, this handles authentication and reads in a specified sheet, or all sheets.

Usage

```
read_googlesheets(
  key_path,
  sheet = "all",
  ss,
  add_primary_key_field = FALSE,
  primary_key = "primary_key",
  ...
)
```

Arguments

`key_path`	character path to Google authentication key json file
`sheet`	Sheet to read, in the sense of "worksheet" or "tab".
`ss`	Something that identifies a Google Sheet such as drive id or URL
`add_primary_key_field`	Logical. Should a primary key field be added?
`primary_key`	character. The column name for the unique identifier to be added to the data.
`...`	other arguments passed to `googlesheets4::range_read()`

Value

tibble

See Also

`googlesheets4::range_read()`

Examples

```
## Not run:
read_googlesheets(ss = kzn_animal_ship_sheets, sheet = "all",)

## End(Not run)
```

remove_deletions *Utility function to identify records for deletion*

Description

Filters for records matching a given string.

Usage

```
remove_deletions(x, val = "Delete")
```

Arguments

x	input vector
val	The value to check for inequality. Defaults to 'Delete'

Details

To be used within `dplyr::filter()`. The function returns a logical vector with TRUE resulting from values that are not equal to the `val` argument. Also protects from NA values.

Used within verbs such as `tidyselect::all_of()` this can work effectively across all columns in a data frame. See examples

Value

logical vector

Examples

```
## Not run:
data |> filter(if_all(everything(), remove_deletions))

## End(Not run)
```

set_diff *Get items that differ between x and y*

Description

Unlike setdiff, this function creates the union of x and y then removes values that are in the intersect, providing values that are unique to X and values that are unique to Y.

Usage

```
set_diff(x, y)
```

Arguments

x	a set of values.
y	a set of values.

Value

Unique values from X and Y, NULL if no unique values.

Examples

```
a <- 1:3
b <- 2:4

set_diff(a,b)
# returns 1,4

x <- 1:3
y <- 1:3

set_diff(x,y)
# returns NULL
```

---

update_frictionless_metadata

Update descriptive metadata in frictionless datapackage

Description

This function overwrites the descriptive metadata associated with a frictionless datapackage. It does *NOT* validate the metadata, or check for conflicts with existing descriptive metadata. It is very easy to create invalid metadata.

Usage

```
update_frictionless_metadata(descriptive_metadata, data_package_path)
```

Arguments

descriptive_metadata	List of descriptive metadata terms.
data_package_path	Character. Path to datapackage.json file

Value

invisibly writes datapackage.json

Examples

```
## Not run:
descriptive_metadata <- list (
title = "Example Dataset",
description = "This is the abstract but it needs more detail",
creator = list (list (name = "A. Person"), list (name = "B. Person"),
list (name = "C. Person"),list (name = "F. Person"))
# , accessRights = "open"
)
update_frictionless_metadata(descriptive_metadata = descriptive_metadata,
                             data_package_path = "data_examples/datapackage.json"
)

## End(Not run)
```

`update_structural_metadata` *Update structural metadata*

Description

Appends rows and/or columns to existing metadata, change primary key and/or adds foreign keys.

Usage

```
update_structural_metadata(
  data,
  metadata,
  primary_key = "",
  foreign_key = "",
  additional_elements = tibble::tibble()
)
```

Arguments

`data`	Any named object. Expects a table but will work superficially with lists or named vectors.
`metadata`	Data frame. Output from `create_structural_metadata`
`primary_key`	Character. OPTIONAL Primary key in the data
`foreign_key`	Character. OPTIONAL Foreign key or keys in the data
`additional_elements`	data frame. OPTIONAL Empty tibble with structural metadata elements and their types.

Value

data.frame

Note

See vignette on metadata for examples

`validation_checks` *Validation Correction Checks*

Description

Validation correction tests to be run on data before and after validation to test expectations.

Usage

```
validation_checks(validation_log, before_data, after_data, idcol)
```

Arguments

`validation_log`	tibble Validation log
`before_data`	tibble Data before corrections
`after_data`	tibble Data after corrections
`idcol`	character the primary key for the 'after_data'

Details

As part of the OH cleaning pipelines, raw data is converted to 'semi-clean' data through a process of upserting records from an external Validation Log. To ensure these corrections were made as expected, some checks are performed in this function.

1. If no existing log exists > no changes are make to data
    - Same variables
    - same Rows
    - No unequal values
2. If log exists but no changes are recommended > no changes to data.
    - Same variables
    - same Rows
    - No unequal values
3. Log exists and changes recommended > number of changes are same as log
    - Same variables
    - same Rows
    - Number of changing records in data match records in log
4. Correct fields and records are being updated
    - Checks before and after variables and rows are the same
    - Checks the variable names and row indexes are the same in the logs and the changed data.

Value

NULL if passed or stops with error

Examples

```
## Not run:
    validation_checks(
    validation_log = kzn_animal_ship_existing_log,
    before_data = kzn_animal_ship,
    after_data = kzn_animal_ship_semiclean,
    idcol = "animal_id"
    )

## End(Not run)
```

Index