Package: glamr (via r-universe)

October 30, 2024

Type Package

Title SI Utilities Package

Version 2.0.4

Description Provides a series of base functions useful to the GH OHA SI team. This includes project setup, pulling from DATIM, and key functions for working with the MSD.

License MIT + file LICENSE

URL https://github.com/USAID-OHA-SI/glamr

BugReports https://github.com/USAID-OHA-SI/glamr/issues

Encoding UTF-8

LazyData true

RoxygenNote 7.3.1

Imports crayon, dplyr, glue, googledrive, lubridate, magrittr, purrr, stringr, tidyr, usethis, utils, keyring, digest, rstudioapi

Suggests curl, fs, zip, httr, googlesheets4, janitor, jsonlite, knitr, plyr, rmarkdown, readxl, rvest, testthat (>= 2.1.0), tibble, xml2, vroom, countrycode, stringi

Remotes USAID-OHA-SI/gagglr

VignetteBuilder knitr

Depends R (>= 2.10)

Repository https://usaid-oha-si.r-universe.dev

RemoteUrl https://github.com/USAID-OHA-SI/glamr

RemoteRef HEAD

RemoteSha d1b3cc41c8c92f486a374247873042933234ec39

Contents

`clean_countries` *Clean Natural Earth Country names to match PEPFAR Data*

Description

'clean_countries' is used to adjust Natural Earth country names to match PEPFAR's Operatingunit / country. This function can also be used to shorten OU/Country names by setting the parameter short to TRUE.

Usage

```
clean_countries(.data, colname = "admin", language = "en", short = TRUE)
```

Arguments

`.data`	Reference Datasets
`colname`	Column name to be updated
`language`	language of reference, default is set to 'en'. Options are: 'fr', 'de', 'es', 'ar'
`short`	If TRUE, shorten OU/Country names instead, default is TRUE

Value

Cleaned DataFrame

Examples

```
## Not run:
 library(sf)
 library(rnaturalearth)
 library(glamr)

 spdf <- ne_countries(type = "sovereignty", scale = 110, returnclass = "sf") %>%
   sf::st_drop_geometry() %>%
   dplyr::select(sovereignt, admin, name, adm0_a3) %>%
   glamr::clean_countries(colname = "admin")
## End(Not run)
```

clean_filename *Clean Filename*

Description

This function is primarily useful for removing any apostrophe from the filename since this will get rejected by Google Drive, but also includes features like replacing spaces with an underscore, converting to all lowercase, and adding a date prefix or suffix.

Usage

```
clean_filename(
  x,
  rm_apostrophe = TRUE,
  rp_space = FALSE,
  mk_lower = FALSE,
  add_date = NULL
)
```

Arguments

x	filepath or file name
rm_apostrophe	remove all apostrophes, default = TRUE
rp_space	replace spaces with underscore, default = FALSE
mk_lower	make lowercase, default = FALSE
add_date	add date "prefix" or "suffix"

Value

clean filename

Examples

```
## Not run:
file <- "Submission_Coted'Ivoire_data.csv"
new_file <- clean_filename(file, rm_apostrophe = TRUE, add_date = 'prefix')
## End(Not run)
```

connect_text *Clean and connect parts of text together*

Description

Clean and connect parts of text together

Usage

```
connect_text(txt, connections = "[^a-zA-Z0-9]", connector = "_")
```

Arguments

txt	String charactors
connections	Characters to be replaced by connector
connector	Charactor used as connector

Value

cleaned text

Examples

```
## Not run:
connect_text("THIS - is complex (very bad)")
## End(Not run)
```

convert_date_to_qtr *Convert Date to FY Quarter/Period*

Description

Convert Date to FY Quarter/Period

Usage

```
convert_date_to_qtr(date)
```

Arguments

date	date formatted like 2021-10-01

Value

vector of FY period, eg FY22Q1

See Also

Other period: convert_datim_pd_to_qtr(), convert_fy_qtr_to_pd(), convert_qtr_to_date()

Examples

```
## Not run:
dates <- c("2021-10-01", "2021-11-15")
convert_date_to_qtr(dates)
## End(Not run)
```

---

convert_datim_pd_to_qtr

Convert DATIM CY Quarter/Period to FY Quarter

Description

Convert a period a DATIM API in the format of FY22Q1 or FY22 (for targets/cumulative). This function is built into 'extract_datim'.

Usage

```
convert_datim_pd_to_qtr(df, pd_col = "Period")
```

Arguments

df	dataframe from DATIM API, grabr::extract_datim()
pd_col	name of the period column, default = "Period"

Value

Convert periods from long CY dates to PEPFAR standard FY

See Also

[set_datim()] to store DATIM authentication; [load_secrets()] to store DATIM authentication

Other period: convert_date_to_qtr(), convert_fy_qtr_to_pd(), convert_qtr_to_date()

Examples

```
## Not run:
df <- tibble::tibble(Periods = c("October - December 2023",
                                 "January - March 2024",
                                 "October 2023 to September 2024"))
df <- convert_datim_pd_to_qtr(df)
## End(Not run)
```

convert_fy_qtr_to_pd *Convert Fiscal Year and Quarter into Period*

Description

Using ‘gophr::reshape_msd()‘ often creates the a perfable long dataset when working with the MSD, but may restrict the user to certain default during the process. Creating a clean period (eg FY22Q1) requires a number of lines of code to get right, so this function provides stopgap when you are working with a long dataset that has a fiscal year and quarter column and desire a period variable.

Usage

```
convert_fy_qtr_to_pd(df, fy_ind = "fiscal_year", qtr_ind = "qtr")
```

Arguments

df	MSD data frame reshaped long, eg ‘pivot_longer‘
fy_ind	indicator name in df for the fiscal year, default = "fiscal_year"
qtr_ind	indicator name in the df for quarters, default = "qtrs"

Value

united period column combining and cleaning fiscal year and quarter

See Also

Other period: convert_date_to_qtr(), convert_datim_pd_to_qtr(), convert_qtr_to_date()

Examples

```
df_summary <- df_msd %>%
       filter(indicator == "TX_CURR",
              standardizeddisaggregate == "Total Numerator",
              operatingunit == "Jupiter") %>%
       group_by(mech_code, fiscal_year) %>%
       summarise(across(starts_with("qtr"), sum, na.rm = TRUE),
                 .groups = "drop")

df_summary <- df_summary %>%
       pivot_longer(-c(mech_code, fiscal_year), names_to = "qtrs")

df_summary <- convert_fy_qtr_to_pd(df_summary)
```

convert_qtr_to_date *Convert FY Quarter/Period to Date*

Description

Convert a period (from reshape_msd()) in the format of FY22Q1 or FY22 (for targets/cumulative).

Usage

```
convert_qtr_to_date(period, type = "start")
```

Arguments

period	period formated like FY22Q1 or FY22
type	start or end date of quarter/period, default = "start"

Value

date vector

See Also

Other period: convert_date_to_qtr(), convert_datim_pd_to_qtr(), convert_fy_qtr_to_pd()

Examples

```
## Not run:
df <- read_msd(path)
df <- df %>%
  filter(df,
         operatingunit == "Jupiter",
         indicator == "TX_NEW",
         standarddisaggregate == "Total Numerator") %>%
  group_by(fiscal_year, primepartner) %>%
  summarize(across(start_with("qtr"), sum, na.rm = TRUE)) %>%
  ungroup()
df <- df %>%
  reshape_msd() %>%
  mutate(date = convert_qtr_to_date(period), .after = period)
## End(Not run)
```

curr_date *Get formatted current date*

Description

Get formatted current date

Usage

```
curr_date(fmt = "%Y-%m-%d")
```

Arguments

fmt	Date format

Examples

```
## Not run:
 curr_date()
 curr_date(fmt = "%m/%d/%Y")

## End(Not run)
```

datim_pwd *Return DATIM password*

Description

To setup/store, run 'glamr::set_datim()'.

Usage

```
datim_pwd()
```

Value

access DATIM password from keyring

See Also

Other authentication: `datim_user()`, `get_account()`, `get_key()`, `get_keys()`, `get_s3key()`, `get_services()`, `load_secrets()`, `pano_pwd()`, `pano_user()`, `pdap_access()`, `pdap_bucket()`, `pdap_secret()`, `set_account()`, `set_datim()`, `set_email()`, `set_key()`, `set_pano()`, `set_s3keys()`

Examples

```
## Not run:
load_secrets()
ou_table <- datim_outable(datim_user(), datim_pwd())
## End(Not run)
```

`datim_user` *Return DATIM username*

Description

To setup/store, run 'glamr::set_datim()'.

Usage

```
datim_user()
```

Value

access DATIM username from keyring

See Also

Other authentication: `datim_pwd()`, `get_account()`, `get_key()`, `get_keys()`, `get_s3key()`, `get_services()`, `load_secrets()`, `pano_pwd()`, `pano_user()`, `pdap_access()`, `pdap_bucket()`, `pdap_secret()`, `set_account()`, `set_datim()`, `set_email()`, `set_key()`, `set_pano()`, `set_s3keys()`

Examples

```
## Not run:
load_secrets()
ou_table <- datim_outable(datim_user(), datim_pwd())
## End(Not run)
```

`export_drivefile` *Export local files for googledrive folder*

Description

'r lifecycle::badge("experimental")'

'export_drivefile() is designed to move files googledrive'

Usage

```
export_drivefile(
  filename,
  to_drive,
  to_folder = NULL,
  add_folder = TRUE,
  overwrite = TRUE,
  ...
)
```

Arguments

filename	Character, Full name of the file to be uploaded
to_drive	Character, Google drive id
to_folder	Character, Google drive sub-folder
add_folder	Logical. If TRUE, add sub-folders if they are not present
overwrite	Logical. If yes, existing files will be overwritten
...	Additional parameters to be passed to ‘googledrive::drive_upload()‘

Value

Googledrive file(s) id(s)

Examples

```
## Not run:
 library(glamr)

 list.files("./Graphics", "NIGERIA", full.names = TRUE) %>%
      export_drivefile(filename = .,
                       to_drive = "<path-id>",
                       to_folder = "FY99Q4/VL Suppression",
                       add_folder = TRUE)


## End(Not run)
```

---

extract_excel_data *Extract data from excel link*

Description

Extract data from excel link

Usage

```
extract_excel_data(src_page, link_id, file_sheet = 2, file_ext = "xlsx")
```

Arguments

src_page	The http(s)link to the source web page
link_id	A CSS identifer of the hyperlinked element
file_sheet	The file sheet number or name
file_ext	The extension of the file

Value

File content as a data frame

---

extract_tbl_data *Extract table data from web page*

Description

Extract table data from web page

Usage

```
extract_tbl_data(src_url, tbl_id)
```

Arguments

src_url	The http(s)link to the source web page
tbl_id	A unique identifer of the target table

Value

A data frme

extract_text *Extract text*

Description

Extract text

Usage

```
extract_text(txt, limits = "()")
```

Arguments

txt	text containing parenthesis
limits	area to extract text from, c("()", "", "[]")

Value

text within limits

Examples

```
## Not run:
 extract_text(txt = "Saint Mary Hopital (SMH)")
 extract_text(txt = "TDB [Placeholder - New Mechanism]")
## End(Not run)
```

folder_setup *Setup Folder Structure in Project*

Description

'folder_setup()' creates an organizational structure that is common across OHA/SI projects so every analyst knows what to expect and where when picking up a new project or one cloned from a co-worker. This function can be used as a stand alone function but primarily serves 'si_setup()'

Usage

```
folder_setup(
 folder_list = list("Data", "Images", "Scripts", "AI", "Dataout", "Data_public", "GIS",
    "Documents", "Graphics", "markdown")
)
```

Arguments

folder_list	list of folders to install

Details

The standard setup provides the following folders for these uses: * Data - where any raw/input data (**xlsx/csv/rds**) specific to the project are stored * Dataout - where any intermediary or final data (**xlsx/csv/rds**) are output as a product of your code * Data_public - where all public data lives * * Scripts - where all the code (**R/py**) are stored (if there is a local order, make sure to add prefixes to each script, e.g. 00_init.R, 01_data-access.R, 02_data-munging.R, ...) * Images - any **png/jpeg** visual outputs from your code * Graphics - any **svg/pdf** visual outputs that will be edited in vector graphics editor, eg Adobe Illustrator or Inkscape * AI - any **ai** files or other files from a graphics editor (exported pngs products will be stored in Images) * GIS - any **shp** files or other GIS releated inputs * Documents - any **docx/xlsx/pptx/pdf** documents that relate to the process or are final outputs * markdown - exported **md** files from a knitr report

See Also

Other project setup: `setup_gitignore()`, `setup_readme()`, `si_setup()`

Examples

```
## Not run:
#standard
  folder_setup()
#specific
  fldrs <- c("Data", "Tableau", "AI")
  folder_setup(fldrs)

## End(Not run)
```

`gdrive_folder` *Get id of googledrive folder*

Description

Get id of googledrive folder

Usage

```
gdrive_folder(name, path, add = FALSE, ...)
```

Arguments

`name`	Googledrive folder name
`path`	Googledrive parent path id
`add`	Should folder be added if missing, default is true
`...`	Other arguments to passed on to ‘drive_mkdir‘

Value

Googledrive folder item it or NULL for non existing folder

Note

This function will create a new folder if add is set to TRUE

Examples

```
## Not run:
 library(glamr)

 gdrive_folder("Test-Folder", "ID-adfdfsdfdfdfs")

## End(Not run)
```

`gdrive_metadata` *Unpack Google Drive Metadata*

Description

Google API provides extra metadata stored as a list in the dribble returned, eg modified time, permissions, owner, etc.

Usage

```
gdrive_metadata(df, show_details = FALSE)
```

Arguments

`df`	Results from Google Drive drive_ls
`show_details`	Show all metadata fields, default is FALSE

Value

adds extra meta data to data frame

Examples

```
## Not run:
library(googledrive)
drive_auth()
fldr <- as_id("<google-folder-id>")
drive_ls(fldr) %>% gdrive_metadata()

## End(Not run)
```

gen_ref_id *Generate Reference ID*

Description

This function returns a unique reference id that can be used in scripts and cited in associated plots to help find the associated code on GitHub.

Usage

```
gen_ref_id()
```

Details

A best practice would be to store the character string output as an object called ‘ref_id‘ in the top matter of your script. If you run ‘gophr::get_metadata‘ after this is stored as an object, it will automatically store this for use in ‘metadata$caption‘.

Value

8 character string

Examples

```
## Not run:
library(glamr)
library(gophr)
library(ggplot2)
library(glue)
#create a reference id to include in a plot
gen_ref_id()
ref_id <- "1e64716c"
get_metadata()
#plot with ref id
ggplot(iris, aes(Sepal.Length, Sepal.Width)) +
  geom_point() +
  labs(caption = glue("Source: Edgar Anderson's Iris Data | Ref id: {ref_id}"))
Or
ggplot(iris, aes(Sepal.Length, Sepal.Width)) +
  geom_point() +
  labs(caption = metadata$caption)

## End(Not run)
```

get_account *Get account details*

Description

Get account details

Usage

```
get_account(name)
```

Arguments

name	Service name of the account

Value

key / value pair as list containing details of the account (invisible)

Note

Inspired by ‘grabr::lazy_secrets()‘

See Also

Other authentication: datim_pwd(), datim_user(), get_key(), get_keys(), get_s3key(), get_services(), load_secrets(), pano_pwd(), pano_user(), pdap_access(), pdap_bucket(), pdap_secret(), set_account(), set_datim(), set_email(), set_key(), set_pano(), set_s3keys()

Examples

```
## Not run:
get_account(name = 's3')
## End(Not run)
```

get_key *Get value of service key name*

Description

Get value of service key name

Usage

```
get_key(service, name)
```

Arguments

service	Name of the service
name	Name of the key

Value

key value

See Also

Other authentication: datim_pwd(), datim_user(), get_account(), get_keys(), get_s3key(), get_services(), load_secrets(), pano_pwd(), pano_user(), pdap_access(), pdap_bucket(), pdap_secret(), set_account(), set_datim(), set_email(), set_key(), set_pano(), set_s3keys()

Examples

```
## Not run:
get_key(service = '<service-name>', name = '<key-name>')
## End(Not run)
```

---

get_keys *Get Service Keys*

Description

Get Service Keys

Usage

```
get_keys(service)
```

Arguments

service	Account Service name

Value

list of key names for active services

See Also

Other authentication: datim_pwd(), datim_user(), get_account(), get_key(), get_s3key(), get_services(), load_secrets(), pano_pwd(), pano_user(), pdap_access(), pdap_bucket(), pdap_secret(), set_account(), set_datim(), set_email(), set_key(), set_pano(), set_s3keys()

Examples

```
## Not run:
get_keys('<service-name>')
## End(Not run)
```

`get_s3key` *Get S3 Credentials - Access or Secret Access Key*

Description

'get_s3key' retrieves your S3 keys using the 'keyring' package. Set name to 'access' for 'Access Key', 'name' to 'secret' for 'Secret Access Key'

Usage

```
get_s3key(name = "access")
```

Arguments

`name`	S3 account key

Value

stored key

See Also

Other authentication: `datim_pwd()`, `datim_user()`, `get_account()`, `get_key()`, `get_keys()`, `get_services()`, `load_secrets()`, `pano_pwd()`, `pano_user()`, `pdap_access()`, `pdap_bucket()`, `pdap_secret()`, `set_account()`, `set_datim()`, `set_email()`, `set_key()`, `set_pano()`, `set_s3keys()`

Examples

```
## Not run:
get_s3key(name = "access")
## End(Not run)
```

get_services *Get Services*

Description

Get Services

Usage

```
get_services()
```

Value

list of active services

See Also

Other authentication: datim_pwd(), datim_user(), get_account(), get_key(), get_keys(), get_s3key(), load_secrets(), pano_pwd(), pano_user(), pdap_access(), pdap_bucket(), pdap_secret(), set_account(), set_datim(), set_email(), set_key(), set_pano(), set_s3keys()

Examples

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

import_drivefile *Import file from a Google Drive folder*

Description

'import_drivefile' is a wrapper around 'googledrive::drive_download', useful for pulling multiple files from a given Google Drive folder (with a Google ID provided) to download by default to the Data folder of a project.

Usage

```
import_drivefile(drive_folder, filename, folderpath = "Data", zip = TRUE)
```

Arguments

drive_folder	Google id for Google Drive Folder
filename	exact name of file on Googl Drive to download
folderpath	path where you want file stored, default = "Data"
zip	should the file be zipped? default = TRUE

Value

stores file from Google Drive as a zipped file

Examples

```
## Not run:
library(googledrive)
googledrive::drive_auth()
fldr <- "Spp-y8DYsdRTrzDqUmK4fX5v"
import_drivefile(fldr, "TestFile.csv")

## End(Not run)
```

---

is_stored *Test if service is stored in credential manager*

Description

Test if service is stored in credential manager

Usage

```
is_stored(service = c("datim", "email", "pano", "s3", "pdap"))
```

Arguments

service	account, either "email", "datim", "pano", "s3", "pdap"

Value

A boolean

---

load_secrets *Load credentials*

Description

'load_secrets' should be set at the beginning of a script to store your email and DATIM user name under Options for the current session. This allows analysts to more easily share their scripts without having to manually update or remove use names.

Usage

```
load_secrets(service = c("email", "datim", "pano", "s3", "pdap"))
```

Arguments

service	account, either "email", "datim", "pano", "s3", or "pdap"; by default, all are loaded if they are available

Details

To initially store your credentials, you will first need to run ‘set_email()‘, ‘set_datim()‘, ‘set_pano()‘, and/or ‘set_key‘ (for s3)

‘load_secrets‘ utilizes ‘keyring‘ package to access the OS credentials store. Storing in a centralized, secure location allows analysts to other analysts code without having to manually change user names/email address to access DATIM or Google Drive.

Value

stores Google, DATIM, PEFPFAR Panorama, s3, and PDAP credentials in session

See Also

Other authentication: datim_pwd(), datim_user(), get_account(), get_key(), get_keys(), get_s3key(), get_services(), pano_pwd(), pano_user(), pdap_access(), pdap_bucket(), pdap_secret(), set_account(), set_datim(), set_email(), set_key(), set_pano(), set_s3keys()

Examples

```
## Not run:
load_secrets()
ou_table <- datim_outable(datim_user(), datim_pwd())
## End(Not run)
```

---

lookup_country *Lookup official Country name*

Description

Lookup official Country name

Usage

```
lookup_country(country, language = "en")
```

Arguments

country	country name
language	language to use for lookup

Value

cleaned country name

Examples

```
{
 cntry <- "Cote d'Ivoire"

 name <- lookup_country(cntry)

 name
}
```

---

open_path *Open directory explorer or files*

Description

Open directory explorer or files

Usage

```
open_path(path)
```

Arguments

path	Full path of the file to be opened

Note

This assumes default applications are set for various file type

Examples

```
## Not run:
 dir_name <- "C:/Users/<username>/Downloads"
 open_path(dir_name)

 file_name <- "C:/Users/<username>/Downloads/test.csv"
 open_path(file_name)

## End(Not run)
```

pano_pwd *Return PEPFAR Panorama password*

Description

To setup/store, run 'glamr::set_pano()'.

Usage

```
pano_pwd()
```

Value

access Panorama password from keyring

See Also

Other authentication: datim_pwd(), datim_user(), get_account(), get_key(), get_keys(), get_s3key(), get_services(), load_secrets(), pano_user(), pdap_access(), pdap_bucket(), pdap_secret(), set_account(), set_datim(), set_email(), set_key(), set_pano(), set_s3keys()

pano_user *Return PEPFAR Panorama username*

Description

To setup/store, run 'glamr::set_pano()'.

Usage

```
pano_user()
```

Value

access Panorama username from keyring

See Also

Other authentication: datim_pwd(), datim_user(), get_account(), get_key(), get_keys(), get_s3key(), get_services(), load_secrets(), pano_pwd(), pdap_access(), pdap_bucket(), pdap_secret(), set_account(), set_datim(), set_email(), set_key(), set_pano(), set_s3keys()

pdap_access *Store PDAP Access Key credentials*

Description

When working with PDAP, you will need to access data from either the read or write buckets and need the credentials to do so. This function stores the Access Key associated with your account, 'Sys.getenv("AWS_ACCESS_KEY_ID")'. To use locally, the user will need to store 'set_key('pdap', 'access')', which securely stores this information with 'keyring' (we can only write, not read from a local machine).

Usage

```
pdap_access()
```

See Also

Other authentication: datim_pwd(), datim_user(), get_account(), get_key(), get_keys(), get_s3key(), get_services(), load_secrets(), pano_pwd(), pano_user(), pdap_bucket(), pdap_secret(), set_account(), set_datim(), set_email(), set_key(), set_pano(), set_s3keys()

Examples

```
## Not run:
library(grabr)
s3_upload(upload_file_path,
          bucket = pdap_bucket("write"),
          prefix = "usaid/",
          access_key = pdap_access(),
          secret_key = pdap_secret())

#identify path to dataset uploaded
  path_wrkbnch <- s3_objects(bucket = pdap_bucket("write"),
                             prefix = "usaid/",
                             access_key = pdap_access(),
                             secret_key = pdap_secret()) %>%
    filter(str_detect(key, "Moz")) %>%
    pull(key)

#read
df_msd <- s3read_using(read_psd,
                       bucket = pdap_bucket("write"),
                       object = path_wrkbnch)
## End(Not run)
```

pdap_bucket *Access PDAP read/write bucket*

Description

When working with PDAP, you will need to access data from either the read or write buckets. The read bucket ("S3_READ") is where PEPFAR Systems stores the MSDs and the write bucket ("S3_WRITE") is where users can upload files (USAID users will have access and write to the "usaid/" sub bucket).

Usage

```
pdap_bucket(type = c("read", "write"))
```

Arguments

type	is the bucket read (default) or write?

Details

When access from PDAP Posit Workbench, the function will access the system environment variables ‘Sys.getenv("S3_READ")‘ or ‘Sys.getenv("S3_WRITE")‘ where as it accessing locally, the user will need to store the read bucket location with ‘set_key()‘, which securely stores this information with ‘keyring‘ (we can only write, not read from a local machine).

Value

character string of AWS bucket location

See Also

Other authentication: datim_pwd(), datim_user(), get_account(), get_key(), get_keys(), get_s3key(), get_services(), load_secrets(), pano_pwd(), pano_user(), pdap_access(), pdap_secret(), set_account(), set_datim(), set_email(), set_key(), set_pano(), set_s3keys()

Examples

```
## Not run:
library(grabr)
s3_upload(upload_file_path,
          bucket = pdap_bucket("write"),
          prefix = "usaid/",
          access_key = pdap_access(),
          secret_key = pdap_secret())

#identify path to dataset uploaded
  path_wrkbnch <- s3_objects(bucket = pdap_bucket("write"),
                             prefix = "usaid/",
                             access_key = pdap_access(),
```

```
                              secret_key = pdap_secret()) %>%
    filter(str_detect(key, "Moz")) %>%
    pull(key)

#read
df_msd <- s3read_using(read_psd,
                       bucket = pdap_bucket("write"),
                       object = path_wrkbnch)
## End(Not run)
```

---

pdap_secret *Store PDAP Secret Access Key credentials*

Description

When working with PDAP, you will need to access data from either the read or write buckets and need the credentials to do so. This function stores the Secret Access Key associated with your account, 'Sys.getenv("AWS_SECRET_ACCESS_KEY")'. To use locally, the user will need to store 'set_key('pdap', 'secret')', which securely stores this information with 'keyring' (we can only write, not read from a local machine).

Usage

```
pdap_secret()
```

See Also

Other authentication: datim_pwd(), datim_user(), get_account(), get_key(), get_keys(), get_s3key(), get_services(), load_secrets(), pano_pwd(), pano_user(), pdap_access(), pdap_bucket(), set_account(), set_datim(), set_email(), set_key(), set_pano(), set_s3keys()

Examples

```
## Not run:
library(grabr)
s3_upload(upload_file_path,
          bucket = pdap_bucket("write"),
          prefix = "usaid/",
          access_key = pdap_access(),
          secret_key = pdap_secret())

#identify path to dataset uploaded
  path_wrkbnch <- s3_objects(bucket = pdap_bucket("write"),
                             prefix = "usaid/",
                             access_key = pdap_access(),
                             secret_key = pdap_secret()) %>%
    filter(str_detect(key, "Moz")) %>%
    pull(key)
```

```
#read
df_msd <- s3read_using(read_psd,
                       bucket = pdap_bucket("write"),
                       object = path_wrkbnch)
## End(Not run)
```

`pepfar_country_list` *Current List of PEPFAR OUs/Countries and their ISO codes*

Description

A dataset PEPFAR Operating Units and Countries along with their ISO codes. This is a useful dataset for having a full set of PEPFAR countries or to align ISO codes with external datasources. Pulled from DATIM and the FSD.

Usage

```
pepfar_country_list
```

Format

A data frame with 55 rows and 7 variables:

operatingunit PEPFAR Operating Unit (countries + 3 regional programs)

operatingunit_iso PEPFAR Operating Unit ISO-3

operatingunit_uid PEPFAR Operating Unit unique id from DATIM

country PEPFAR Country Name

country_iso PEPFAR Country Name ISO-3

country_uid PEPFAR Country unique id from DATIM

pepfar_accel Is the country prioritized in the 8+1+1 group defined by SGAC?

Details

The list of PEPFAR acceleration countries was defined by Amb. Nkengasong during a DP's retreat for the [Zaidi 2023-06-08 re: Moving countries to green!]. These are countries "where enhanced attention and focus might help 'move the dial' on achieving and sustaining the UNAIDS 95-95-95 targets by 2025"

Source

https://final.datim.org/

`pepfar_country_xwalk` *Cross-walk of Country names*

Description

A dataset PEPFAR Operating Units and Countries along with their ISO codes, alternative names from other sources. This is a useful dataset designed to help with data cleaning / matching from different sources.

Usage

```
pepfar_country_xwalk
```

Format

A data frame with 55 rows and 29 variables:

iso3c ISO-3 Code

continent Continent name

region World Region Name

region23 Alternative name for World Region

un_region_name US Region Name

iso_name_en ISO Country name in English

iso_name_fr SO Country name in French

country_name_de Regular Expression of Country name in German

country_name_de_regex Country name in German

country_name_en Country name in English

country_name_en_regex Regular Expression of Country name in English

country_name_fr Regular Expression of Country name in French

country_name_fr_regex Country name in French

country_name_it Regular Expression of Country name in Italian

country_name_it_regex Country name in Italian

un_name_ar UN Name in Arabic

un_name_en UN Name in English

un_name_es UN Name in Spanish

un_name_fr UN Name in French

un_name_ru UN Name in ru

un_name_zh UN Name in zh

sovereignt rnaturalearth sovereign territory name

admin rnaturalearth administrative unit name

name rnaturalearth country name

operatingunit PEPFAR Operating Unit (countries + 3 regional programs)

operatingunit_iso PEPFAR Operating Unit ISO-3

operatingunit_uid PEPFAR Operating Unit unique id from DATIM

country PEPFAR Country Name

country_uid PEPFAR Country unique id from DATIM

Source

https://final.datim.org/ https://www.naturalearthdata.com/ https://vincentarelbundock.github.io/countrycode/

`pepfar_data_calendar` *Current PEPFAR Data Calendar*

Description

A dataset the dates for the release of the MER Structured Dataset

Usage

```
pepfar_data_calendar
```

Format

A data frame with 16 rows and 6 variables:

fiscal_year fiscal year, start = October

quarter fiscal quarter, integer 1-4

type data release type, initial or clean

entry_open date entry begins into DATIM

entry_close date DATIM is closed and data are frozen

msd_release date the MSD is released on PEPFAR Panorma

Source

https://datim.zendesk.com/hc/en-us/articles/115001940503-PEPFAR-Data-Calendar

prinf *Print All Rows*

Description

'prinf' is a wrapper around 'print' that returns all rows rather than just the first 10 by default.

Usage

```
prinf(df, ...)
```

Arguments

df	data frame
...	Any other valid option for 'print()'.

Value

prints out all rows rather than default 10 rows.

Examples

```
## Not run:
df_geo %>% prinf()
## End(Not run)
```

return_latest *Return Latest File*

Description

'return_latest' checks for a pattern in a folder and provides the most recent file bases on the time the file was modified

Usage

```
return_latest(folderpath, pattern, quiet = FALSE, ...)
```

Arguments

folderpath	path to folder where file(s) are located.
pattern	pattern in file name, regex expressions. If not parttern is provided, the last file in the folder will be returned.
quiet	suppresses the output message related to the file name creation, for use in sub functions, default = FALSE
...	Any other valid option for 'base::list.files()'.

Value

a vector of the full filepath for the most recent version of a file stub

Examples

```
## Not run:
file_stub <- "MER_Structured_Datasets_OU_IM_FY18-20"
filepath <- return_latest("Data", file_stub)
df <- read_rds(filepath)
## End(Not run)
```

`setup_gitignore` *Setup gitignore in Project*

Description

'setup_gitignore()' creates a .gitignore file (or appends to an existing) one) the standard file types/folders that should not be published to GitHub due to the sensitive nature of PEPFAR data. This function can be used as a stand alone function but primarily serves 'si_setup()'.

Usage

```
setup_gitignore()
```

Value

adds standard gitignore, plus specific ignores

See Also

Other project setup: `folder_setup()`, `setup_readme()`, `si_setup()`

Examples

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

setup_readme *Setup README with Disclaimer in Project*

Description

'setup_readme()' establishes a README.md with the standard USAID disclaimer (or appends to one that currently exists). This function can be used as a stand alone function but primarily serves 'si_setup()'.

Usage

```
setup_readme(add_disclaimer = TRUE)
```

Arguments

add_disclaimer	should the standard disclaimer be added, default = TRUE

Value

adds/appends disclaimer to README

See Also

Other project setup: folder_setup(), setup_gitignore(), si_setup()

Examples

```
## Not run:
#standard (appends dislaimer if README exists)
  setup_readme()
## End(Not run)
```

set_account *Create / Update account*

Description

Create / Update account

Usage

```
set_account(name, keys = c("username", "password"), update = FALSE)
```

Arguments

name	Service name of the account
keys	List of account key names
update	Should an existing account be overwriten

See Also

Other authentication: datim_pwd(), datim_user(), get_account(), get_key(), get_keys(), get_s3key(), get_services(), load_secrets(), pano_pwd(), pano_user(), pdap_access(), pdap_bucket(), pdap_secret(), set_datim(), set_email(), set_key(), set_pano(), set_s3keys()

Examples

```
## Not run:
set_account(name = 's3', keys = c("access", "secret"))
## End(Not run)
```

---

set_datim *Store DATIM credentials*

Description

'set_datim' stores your DATIM credentials email using the 'keyring' package. This will only need to done once. After running 'set_datim(user)', you will be promoted to enter your password through the RStudio API which will then store the username and password in your OS credential store using 'keyring'.

Usage

```
set_datim(datim_username)
```

Arguments

datim_username	DATIM account

Details

The 'keyring' package utilized the OS credentials store. Storing in a centralized, secure location allows analysts to other analysts code without having to manually change user names/email address to access DATIM or Google Drive.

After 'set_datim' has been run once, an analyst can set 'load_secrets' at the beginning of a script, storing their email and DATIM username under Options for the current session.

Value

stores USAID email in using keyring

See Also

Other authentication: datim_pwd(), datim_user(), get_account(), get_key(), get_keys(), get_s3key(), get_services(), load_secrets(), pano_pwd(), pano_user(), pdap_access(), pdap_bucket(), pdap_secret(), set_account(), set_email(), set_key(), set_pano(), set_s3keys()

Examples

```
## Not run:
set_datim("rshah")
## End(Not run)
```

`set_email` *Store USAID email*

Description

'set_email' stores your USAID email using the 'keyring' package. This will only need to run once.

Usage

```
set_email(usaid_email)
```

Arguments

`usaid_email`	full USAID email address

Details

The 'keyring' package utilized the OS credentials store. Storing in a centralized, secure location allows analysts to other analysts code without having to manually change user names/email address to access DATIM or Google Drive.

After 'set_email' has been run once, an analyst can set 'load_secrets' at the beginning of a script, storing their email and DATIM username under Options for the current session.

This function also stores the email locally in your .Rprofile, allowing to be used automatically as the default for 'googledrive::drive_auth()' and 'googlesheets4::gs4_auth()'

Value

stores USAID email using keyring and .Rprofile

See Also

Other authentication: datim_pwd(), datim_user(), get_account(), get_key(), get_keys(), get_s3key(), get_services(), load_secrets(), pano_pwd(), pano_user(), pdap_access(), pdap_bucket(), pdap_secret(), set_account(), set_datim(), set_key(), set_pano(), set_s3keys()

Examples

```
## Not run:
set_email("rshah@usaid.gov")
## End(Not run)
```

`set_key` *Set value for service name*

Description

Set value for service name

Usage

```
set_key(service, name)
```

Arguments

`service`	Name of the service
`name`	Name of the key

See Also

Other authentication: `datim_pwd()`, `datim_user()`, `get_account()`, `get_key()`, `get_keys()`, `get_s3key()`, `get_services()`, `load_secrets()`, `pano_pwd()`, `pano_user()`, `pdap_access()`, `pdap_bucket()`, `pdap_secret()`, `set_account()`, `set_datim()`, `set_email()`, `set_pano()`, `set_s3keys()`

Examples

```
## Not run:
set_key(service = '<service-name>', name = '<key-name>')
## End(Not run)
```

`set_pano` *Store PEPFAR Panorama credentials*

Description

'set_pano' stores your PEPFAR Panoram credentials email using the 'keyring' package. This will only need to done once. After running 'set_pano(user)', you will be promoted to enter your password through the RStudio API which will then store the username and password in your OS credential store using 'keyring'.

Usage

```
set_pano(pano_username)
```

Arguments

pano_username	Panorama user name (email)

Details

The 'keyring' package utilized the OS credentials store. Storing in a centralized, secure location allows analysts to other analysts code without having to manually change user names/email address to access DATIM, Panorama, or Google Drive.

After 'set_pano' has been run once, an analyst can set 'load_secrets' at the beginning of a script, storing their PEPFAR Panorama credentials under Options for the current session.

Value

stores Panorama username and password in using keyring

See Also

Other authentication: datim_pwd(), datim_user(), get_account(), get_key(), get_keys(), get_s3key(), get_services(), load_secrets(), pano_pwd(), pano_user(), pdap_access(), pdap_bucket(), pdap_secret(), set_account(), set_datim(), set_email(), set_key(), set_s3keys()

Examples

```
## Not run:
set_pano("rshah@usaid.gov")
## End(Not run)
```

---

set_paths *Set SI local folder paths*

Description

'set_paths' stores store local folder paths where larger data are stored centrally and outside of projects. Accessed through use of 'si_path()'.

Usage

```
set_paths(
  folderpath_msd,
  folderpath_datim,
  folderpath_raster,
  folderpath_vector,
  folderpath_downloads
)
```

Arguments

folderpath_msd	folderpath where the MSDs are stored
folderpath_datim	folderpath where DATIM data are store eg (org hierarchy, mech table)
folderpath_raster	folderpath where GIS raster data are stored
folderpath_vector	folderpath where GIS vector data are stored
folderpath_downloads	folderpath to local Downloads folder

Value

code chunk to paste into .Rprofile

See Also

Other stored paths: si_path()

Examples

```
## Not run:
set_paths <- set_path(folderpath_msd = "C:/Users/rshah/Documents/Data")
## End(Not run)
```

---

set_s3keys *Store S3 Credentials*

Description

'set_s3keys' stores your s3 keys using the 'keyring' package. This will only need to done once. After running 'set_s3keys(access, secret)', RStudio API which will then store the keys in your OS credential store using 'keyring'.

Usage

```
set_s3keys(access, secret)
```

Arguments

access	S3 Account Access Key
secret	S3 Account Secret Key

Value

stored access key

See Also

Other authentication: datim_pwd(), datim_user(), get_account(), get_key(), get_keys(), get_s3key(), get_services(), load_secrets(), pano_pwd(), pano_user(), pdap_access(), pdap_bucket(), pdap_secret(), set_account(), set_datim(), set_email(), set_key(), set_pano()

Examples

```
## Not run:
set_s3access("ABDCEDFF", "MLIZD998SD")
## End(Not run)
```

---

si_path *Access SI folderpath*

Description

'si_path' accesses folder paths stored in global options to make it easier work across analysts/machines. Analysts will first setup the paths using 'set_paths()' which then store local folder paths where larger data are stored centrally and outside of projects. This will also work on PEPFAR Workbench to return the location of the MSD, 'Sys.getenv("S3_READ")'.

Usage

```
si_path(type = "path_msd")
```

Arguments

type	folderpath, eg "path_msd" (default), "path_datim", "path_raster", "path_vector", "path_downloads"

Value

folderpath stored in global options

See Also

Other stored paths: set_paths()

Examples

```
## Not run:
#old
  list.files("C:/Users/rshah/Documents/Data", "OU_IM", full.names = TRUE)
#new
  list.files(si_path("path_msd"), "OU_IM", full.names = TRUE)

## End(Not run)
```

si_setup *Default SI Project Setup*

Description

'si_setup()' combines three function - 'folder_setup()', 'setup_gitignore()', and 'setup_readme()' to create the base OHA/SI project structure with the necessary folders, disclaimers, and ignored files.

Usage

```
si_setup()
```

Value

creates folders, readme, and gitignore

See Also

Other project setup: `folder_setup()`, `setup_gitignore()`, `setup_readme()`

temp_folder *Generate Temporary Folder*

Description

'temp_folder' created a temporary folder in your AppData directory, which will be automatically removed after you close your RStudio session.

Usage

```
temp_folder(launch = FALSE, quiet = FALSE)
```

Arguments

launch	do you want to launch the temp folder in the Windows Explorer? default = FALSE
quiet	suppresses the output message related to the folder creation and location, for use in sub functions, default = FALSE

Value

creates a temp directory and stores it as 'folderpath_tmp'

Examples

```
## Not run:
load_secrets()
temp_folder(launch = TRUE)
purrr::walk2(.x = df_googlefiles$id,
             .y = df_googlefiles$filename,
             .f = ~googledrive::drive_download(googledrive::as_id(.x),
                                               file.path(folderpath_tmp, .y)))
## End(Not run)
```

`%ni%` *Negate in*

Description

negate ‘

Usage

```
... %ni% NA
```

Index