Package: tameDP (via r-universe)

October 23, 2024

Title Import targets and PLHIV data from COP Target Setting Tool (formerly Data Pack)

Version 6.2.4

Description Import PSNUxIM targets and PLHIV data from COP Data Pack. The purpose is to make the data tidy and more usable than their current structure in the Excel data packs.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Imports curl, dplyr (>= 1.0.0), glue, lifecycle, magrittr, readxl, stringr, tidyr, tools, cli

Remotes USAID-OHA-SI/glamr, USAID-OHA-SI/gophr, USAID-OHA-SI/gagglr, USAID-OHA-SI/grabr,

Suggests getPass, testthat (>= 3.0.0), purrr, knitr, rmarkdown

Config/testthat/edition 3

Depends R (>= 2.10)

VignetteBuilder knitr

URL https://usaid-oha-si.github.io/tameDP/

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

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

RemoteRef HEAD

RemoteSha e62274593dc42e185ae6f6d9dbb478fc4d01ea48

Contents

`age_band_crosswalk` *Crosswalk to collapse age bands in MSD to match TST for COP23*

Description

A dataframe containing the age bands in the MSD and the collapsed age bands in the COP23 Target Setting TOol

Usage

```
age_band_crosswalk
```

Format

A data frame with 18 rows and 2 variables:

age_msd Age bands in the MER structured Dataset

age_dp collapsed age bands in the TST

---

agg_dp *Aggregate Targets to IM or PSNU level*

Description

To ensure there are no duplicate rows on the reshape, this function first aggregates the data by the key columns to minimize any issues. If desiring to work at the PSNU level, the parameter 'psnu_lvl' allows you to aggregate to the PSNU level instead of the PSNUxIM level.

Usage

```
agg_dp(df, psnu_lvl = FALSE)
```

Arguments

df	data frame to aggregate
psnu_lvl	default aggregate is to IM level; if TRUE, aggregates to PSNU level

---

align_msd_disagg *Align MSD extract to disaggregates in Target Setting Tool*

Description

This function pulls in a PSNUxIM MSD and datapack filepath to align the MSD extract to the indicators and disaggregates in the datapack, as well as historic results and targets. This function also addresses when OUs set targets at a higher level than PSNU for alignment.

Usage

```
align_msd_disagg(msd_path)
```

Arguments

msd_path	path to PSNUxIM extract

Examples

```
 ## Not run:
   df_msd <- align_msd_disagg(msd_path = msd_path)

## End(Not run)
```

`apply_class` *Apply variable class*

Description

Ensure that fiscal year, cumulative, and targets are numeric and all other variables are stored as characters.

Usage

```
apply_class(df)
```

Arguments

`df`	dataframe output to reorder

`apply_fy` *Apply Fiscal Year*

Description

Apply fiscal year to each row, using the T or T_1 or R in ‘indicator_code‘ to determine whether it’s the current or a prior fiscal year. The fiscal year can be identified dynamically through ‘grab_info()‘.

Usage

```
apply_fy(df, year)
```

Arguments

`df`	DP dataframe to apply fiscal year to
`year`	fiscal year, derived from ‘grab_info(filepath, "year")‘

Value

data frame with fiscal year

apply_prioritization *Apply Prioritization*

Description

Join the new COP prioritization onto the target data frame.

Usage

```
apply_prioritization(df, df_prioritization)
```

Arguments

df	Target Setting Tool data frame
df_prioritization	dataframe from ‘grab_prioritization()‘

See Also

Other prioritization: grab_prioritization()

apply_snu1 *Apply SNU1 to dataframe*

Description

Join the SNU1 onto the PSNUxIM data frame.

Usage

```
apply_snu1(df, df_snu1)
```

Arguments

df	Target Setting Tool data frame
df_snu1	dataframe from ‘grab_snu1()‘

See Also

Other snu1: grab_snu1()

apply_stamps *Apply Source File Name and Date Stamp*

Description

This function applies metadata from the source file to the tidied dataset including the file name, last modified date, and

Usage

```
apply_stamps(df, filepath)
```

Arguments

df	data frame read in and reshaped by import_dp and reshape_dp
filepath	file path to the Target Setting Tool importing, must be .xlsx

Value

new columns in df with source information

clean_indicators *Clean Up Indicators and Disaggregates*

Description

The indicator and disaggregates used in the Target Setting Tool skew towards machine readable and do not necessary match the MER indicators in the MSD/DATIM. This function makes adjustments to indicators and disaggregates to make them easier to work with and more closely align to the MSD. This function also uses 'convert_mods()', which creates the testing modalities that match the MSD and create new HTS_TST and HTS_TST_POS indicator from indicator that feed into them (eg HTS_INDEX, TB_STAT, PMTCT_STAT, VMMC_CIRC).

Usage

```
clean_indicators(df, fy)
```

Arguments

df	data frame to adjust
fy	fiscal year for targeting

`convert_mods` *Duplicate and convert modalities to HTS_TST*

Description

This function matches the testing modalities from the MSD and create new HTS_TST and HTS_TST_POS indicator from indicator that feed into them (eg HTS_INDEX, TB_STAT, PMTCT_STAT, VMMC_CIRC).

Usage

```
convert_mods(df)
```

Arguments

`df`	data frame

`get_names` *Import mechanism specific info from DATIM*

Description

The Target Setting Tool does not contain information on the mechanism (names or partners). By running this function, you are connecting to DATIM's SQLView file that contains the list of all current mechanisms. This requires providing your DATIM credentials. If left blank in the function, you will have two dialogue boxes popping up asking for your DATIM username and password. If running 'tame_dp()' across multiple Target Setting Tools, it's advisable to run 'get_names()' on the file dataset produced by 'tame_dp'.

Usage

```
get_names(
  df,
  map_names = TRUE,
  psnu_lvl = FALSE,
  cntry,
  datim_user,
  datim_password
)
```

Arguments

`df`	data frame to add mechanism info to
`map_names`	import names from DATIM (OU, mechanism, partner) associated with mech_code
`psnu_lvl`	aggregate to the PSNU level instead of IM
`cntry`	country, from grab_info() if not connecting to DATIM
`datim_user`	DATIM user name (if not provided, you will be prompted with a pop up)
`datim_password`	DATIM password (if not provided, you will be prompted with a pop up)

Examples

```
#load package
  library(purrr)
#identify all the Target Setting Tool files
  files <- list.files("../Downloads/DataPacks", full.names = TRUE)
#read in all DPs and combine into one data frame
  df_all <- map_dfr(.x = files,
                    .f = ~ tame_dp(.x, map_names = FALSE))
#apply mech_name and primepartner names from DATIM
#you will need to provide your DATIM credentials
  datim_user_nm <- "spower" #replace with your username
  datim_pwd <- getPass::getPass() #pop up prompting for your password
  df_all <- get_names(df_all, datim_user = datim_user_nm, datim_password = datim_pwd)
```

---

grab_info *Pull Information from Target Setting Tool "Home" Tab*

Description

This function extract information stored in the Target Setting Tool Home tab to identify either the country or what the fiscal year is.

Usage

```
grab_info(filepath, type)
```

Arguments

filepath	file path to the Target Setting Tool importing, must be .xlsx
type	either "country" or "year"

Examples

```
path <- "../Downloads/DataPack_Jupiter_20200218.xlsx"
cntry <- grab_info(path, "country")
fy <- grab_info(path, "year")
```

grab_prioritization *Identify Prioritization*

Description

Pull from the prioritization tab to have a table of PSNU prioritization for the current COP.

Usage

```
grab_prioritization(filepath)
```

Arguments

filepath	file path to the Target Setting Tool importing, must be .xlsx

Value

dataframe from the Prioritization tab

See Also

Other prioritization: `apply_prioritization()`

grab_snu1 *Identify SNU1 associated with PSNU*

Description

Pull SNU1 from the prioritization tab to have a table to align/apply with the PSNUxIM tab

Usage

```
grab_snu1(filepath)
```

Arguments

filepath	file path to the Target Setting Tool importing, must be .xlsx

Value

dataframe from the Prioritization tab

See Also

Other snu1: `apply_snu1()`

import_dp *Import Tabs from the Target Setting Tool*

Description

Initial reading in of tabs of the Target Setting Tool. This function reads in the necessary tab or tabs, removes unused columns and cleans up the column names so there are no duplicates. For the PSNUxIM, it identified columns as as a share or value.

Usage

```
import_dp(filepath, tab)
```

Arguments

filepath	file path to the Target Setting Tool importing, must be .xlsx
tab	which sheet to read in

Examples

```
path <- "../Downloads/DataPack_Jupiter_20200218.xlsx"
df_tst <- import_dp(path, tab = "PSNUxIM")
```

is_file *Is the filepath correct for the Target Setting Tool*

Description

Is the filepath correct for the Target Setting Tool

Usage

```
is_file(filepath)
```

Arguments

filepath	filepath of Target Setting Tool

See Also

Other validation: is_sheet(), is_xls(), no_connection()

is_sheet *Check if a sheet exits in Target Setting Tool*

Description

Check if a sheet exits in Target Setting Tool

Usage

```
is_sheet(filepath, tab = "PSNUxIM")
```

Arguments

filepath	filepath of Target Setting Tool
tab	sheet to check in Target Setting Tool, "PSNUxIM" (default)

See Also

Other validation: is_file(), is_xls(), no_connection()

is_xls *Check if the filepath is .xls or .xlsx*

Description

Check if the filepath is .xls or .xlsx

Usage

```
is_xls(filepath)
```

Arguments

filepath	filepath of COP Target Setting Tool

See Also

Other validation: is_file(), is_sheet(), no_connection()

join_dp_msd *Join TST output with MSD output*

Description

Deprecated! See ‘tame_join‘.

Usage

```
join_dp_msd(dp_filepath, msd_filepath, fy_as_str = TRUE, map_names = FALSE)
```

Arguments

dp_filepath	file path to the Target Setting Tool importing, must be .xlsx
msd_filepath	filepath to the latest PSNUxIM MSD for corresponding OU
fy_as_str	should FY be converted to a string (2025 > FY25) for Tableau? (default = TRUE)
map_names	import names from DATIM (OU, mechanism, partner) associated with mech_code when working with PSNUxIM (default = FALSE)

Value

dataframe that combines targets from the TST with corresponding historic results/targets from MSD

Examples

```
#DP file path
  tst_path <- "../Downloads/DataPack_Jupiter_20500101.xlsx"
# MSD filepath
 msd_path <- "../Data/MER_Structured_TRAINING_Datasets_PSNU_IM_FY59-61_20240215_v1_1.zip"

#run join function (depricated)
  df_join <- join_dp_msd(tst_path, msd_path)
```

limit_datatype *Limit Dataset Type*

Description

This function limits the output of the Target Setting Tool data to either MER or SUBNAT (e.g. PLHIV, TX_CURR_SUBNAT) data. It will not be run if processing the PSNUxIM tab since that does not include any SUBNAT data.

Usage

```
limit_datatype(df, type)
```

Arguments

df	data frame read in and reshaped by import_dp and reshape_dp
type	dataset type, either "MER" or "PLHIV"

Value

data frame limited to either MER or SUBNAT data

---

map_disaggs *Map Standardized Disaggregate*

Description

To align with DATIM datasets, the standardized disaggregates for each indicators will be aligned to the Target Setting Tool for FY22 Targets.

Usage

```
map_disaggs(df)
```

Arguments

df	dataframe from clean_indicators

---

match_col_type *Match Column Type*

Description

This function utilizes the meta data stored in row 6 of each tab of the Data Pack to determine what column type is - "assumption", "calculation", "past", "result", "reference", "row_header", "target". The primary columns we want are meta data (row_header), targets, and past (prior year result/targets for reference).

Usage

```
match_col_type(filepath, tab, pattern = "(row_header|target|past)")
```

Arguments

filepath	file path to the Target Setting Tool importing, must be .xlsx
tab	which sheet to read in
pattern	type of column, "assumption", "calculation", "past", "result", "reference", "row_header", "target"; default = "(row_header|target|past)"

Value

Boolean list of matches

---

mer_disagg_mapping *Table of indicators and their disaggs*

Description

A dataset containing the mapping between MER/SUBNAT/IMPATT indicators from the Target Setting Tool and their official disaggregates in DATIM from FY23/ COP22 targets.

Usage

```
mer_disagg_mapping
```

Format

A data frame with 60 rows and 4 variables:

indicator MER indicator name

numeratordenom designates whether the indicator type

standardizeddisaggregate indicator disaggregation, eg Age/Sex/HIVStatus

kp_disagg whether the disaggregation is for Key Populations

Source

https://datim.zendesk.com/hc/en-us/articles/360001143166-DATIM-Data-Entry-Form-Screen-Shot-Reposit

mer_historic_disagg_mapping_2024

Table of MER indicators and disaggs including historic results disaggs from 2024

Description

A dataset containing the mapping between MER/SUBNAT/IMPATT indicators from the Target Setting Tool and their official disaggregates in DATIM from FY24/ COP23 targets, as well as historic results/targets disaggregates from DATIM from FY22-FY24.

Usage

```
mer_historic_disagg_mapping_2024
```

Format

A data frame with 225 rows and 5 variables:

indicator MER indicator name

numeratordenom designates whether the indicator type

standardizeddisaggregate indicator disaggregation, eg Age/Sex/HIVStatus

fiscal_year fiscal year

kp_disagg whether the disaggregation is for Key Populations

Source

https://datim.zendesk.com/hc/en-us/articles/360001143166-DATIM-Data-Entry-Form-Screen-Shot-Reposit

msd_historic_disagg_mapping

Table of MER indicators and disaggs including historic results disaggs

Description

A dataset containing the mapping between MER/SUBNAT/IMPATT indicators from the Target Setting Tool and their official disaggregates in DATIM from FY23/ COP22 targets, as well as historic results/targets disaggregates from DATIM from FY21-FY23.

Usage

```
msd_historic_disagg_mapping
```

Format

A data frame with 163 rows and 5 variables:

indicator MER indicator name

numeratordenom designates whether the indicator type

standardizeddisaggregate indicator disaggregation, eg Age/Sex/HIVStatus

fiscal_year fiscal year

kp_disagg whether the disaggregation is for Key Populations

Source

https://datim.zendesk.com/hc/en-us/articles/360001143166-DATIM-Data-Entry-Form-Screen-Shot-Reposit

---

`no_connection` *Check if computer has internet connection*

Description

Check if computer has internet connection

Usage

```
no_connection()
```

See Also

Other validation: `is_file()`, `is_sheet()`, `is_xls()`

`order_vars` *Order variables*

Description

Ensure variables in the exported data frame are correctly ordered.

Usage

```
order_vars(df)
```

Arguments

`df`	dataframe output to reorder

ou_ctry_mapping *Current Table of PEPFAR Operating Units and Counties*

Description

A dataset containing the mapping countries and operating units. Most countries are also Operating Units, expect for those in regional programs.

Usage

```
ou_ctry_mapping
```

Format

A data frame with 60 rows and 2 variables:

operatingunit PEPFAR Operating Unit (countries + 3 regional programs)

country PEPFAR Country Name

Source

https://final.datim.org/api/organisationUnits

pivot_results *Pivot Results*

Description

If there are any historic results in the dataset (found in some of the non- PSNUxIM tabs), we want to separate these from the target values to ensure the dataset is tidy and results/targets will not be indavertently aggregated. The reshape will create a cumulative column if results exist in the provided dataframe.

Usage

```
pivot_results(df)
```

Arguments

df	data frame after it's been aggregated

Value

data frame with a cumulative column (when/where results exist)

reshape_dp *Reshape Target Setting Tool Long*

Description

This reshapes the relevant columns from a given tab to long, making it tidy and more usable. It relies on either ‘reshape_tab()‘ or ‘reshape_psnuim()‘ depending on the tab being processed.

Usage

```
reshape_dp(df)
```

Arguments

df	data frame from import_dp()

See Also

Other reshape: reshape_psnuim(), reshape_tab()

reshape_psnuim *Reshape Target Setting Tool Long*

Description

This function limits the columns from the PSNUxIM tab and reshapes it long, so that it is more usable. Three values columns are created in the output - datapacktarget, value, share. This function also splits out the PSNU uid from the PSNU column.

Usage

```
reshape_psnuim(df)
```

Arguments

df	data frame from import_dp()

See Also

Other reshape: reshape_dp(), reshape_tab()

reshape_tab *Reshape Target Setting Tool Tab Long*

Description

This function limits the columns from a target tab (non PSNUxIM) to extract data and reshapes it long, so that it is tidy and more usable. This function also splits out the PSNU uid from the PSNU column.

Usage

```
reshape_tab(df)
```

Arguments

df	data frame from import_dp()

See Also

Other reshape: reshape_dp(), reshape_psnuim()

return_tab *Return Tab*

Description

Identify which tab to import based on what you want to use - PSNUxIM, SUBNAT, or ALL (non mechanism tabs). You can also provide a specific tab name that matches the Target Setting Tool

Usage

```
return_tab(type)
```

Arguments

type	dataset to extract "PSNUxIM", "SUBNAT" (formerly "PLHIV"), "ALL", or a specific tab

Value

tabs to import

split_psnu *Clean & Separate PSNU and PSNU UIDS*

Description

This function removes the contacanated data contained in the same cell. The psnu column in the Target Setting Tool contains both the psnu, psnuuid, and meta data on type - Country/SNU/DREAMS/Military. ‘split_psnu‘ breaks out psnu and psnuuid into two columns and removes any other extraneous information.

Usage

```
split_psnu(df)
```

Arguments

df	Target Setting Tool data frame from tameDP

subset_prioritization *Subset Prioritization Tab*

Description

Subsets the columns of the massive Target Setting Tool tab down to only those that are needed. This depends on the type of tab that is being imported. The Prioritization tab keeps the PSNU and prioritization column.

Usage

```
subset_prioritization(df)
```

Arguments

df	data frame after import

Value

limits to correct columns in data frame from DP tab

See Also

Other subset: `subset_psnuxim()`, `subset_standard()`

subset_psnuxim *Subset PSNUxIM Tab*

Description

Subsets the columns of the massive Target Setting Tool tab down to only those that are needed. This depends on the type of tab that is being imported. PSNUxIM keep all meta data and taget share/value columns.

Usage

```
subset_psnuxim(df)
```

Arguments

df	data frame after import

Value

limits to correct columns in data frame from DP tab

See Also

Other subset: subset_prioritization(), subset_standard()

subset_standard *Subset Standard Tabs*

Description

Subsets the columns of the massive Target Setting Tool tab down to only those that are needed. This depends on the type of tab that is being imported. Standard, non-PSNUxIM/Prioritization) keep column types specified in the Target Setting Tool as row_header, target, or past.

Usage

```
subset_standard(df, filepath, tab)
```

Arguments

df	data frame after import
filepath	file path to the Target Setting Tool importing, must be .xlsx
tab	sheet being imported

Value

limits to correct columns in data frame from DP tab

See Also

Other subset: `subset_prioritization()`, `subset_psnuxim()`

`tame_dp` *Export Tidy data from Target Setting Tool*

Description

tame_dp is the primary function of the tameDP package, reading in the Data Pack and munging in into a tidy data frame to make it more usable to interact with the data than the way it is stored in the Target Setting Tool. **Given the changes to the Target Setting Tool each year, the function only works for the current COP year: COP24.**

Usage

```
tame_dp(filepath, type = "ALL", map_names = FALSE, psnu_lvl = FALSE)
```

Arguments

`filepath`	file path to the Target Setting Tool importing, must be .xlsx
`type`	dataset to extract "PSNUxIM", "SUBNAT", or "ALL" [default] or a specific tab
`map_names`	import names from DATIM (OU, mechanism, partner) associated with mech_code
`psnu_lvl`	aggregate to the PSNU level instead of IM

Details

The main function of 'tameDP' is to bring import a COP Target Setting Tool into R and make it tidy. The function aggregates the COP targets up to the mechanism level, imports the mechanism information from DATIM, and breaks out the data elements to make the dataset more usable.

- Imports Target Setting Tool as tidy data frame - Breaks up data elements stored in the indicatorCode column into distinct columns - Cleans up the HTS variables, separating modalities out of the indicator name - Creates a statushiv column - Cleans and separates PSNU and PSNU UID into distinct columns - Adds in mechanism information from DATIM, including operatingunit, funding agency, partner and mechanism name - Removes any rows with no targets - Allows for aggregate to the PSNU level

See Also

Other primary: `tame_plhiv()`, `tame_subnat()`

Examples

```
#DP file path
  path <- "../Downloads/DataPack_Jupiter_20500101.xlsx"
#read in Target Setting Tool (straight from sheets, not PSNUxIM tab)
  df_tst <- tame_dp(path)
#read in PLHIV/SUBNAT data
  df_tst <- tame_dp(path, type = "SUBNAT")
#read in PSNUxIM data
  df_tst <- tame_dp(path, type = "PSNUxIM")
#apply mechanism names
  df_tst_named <- tame_dp(path, type = "PSNUxIM", map_names = TRUE)
#aggregate to the PSNU level
  df_tst_psnu <- tame_dp(path, type = "PSNUxIM", psnu_lvl = TRUE)
#reading in multiple files and then applying mechanism names (for PSNUxIM)
  df_all <- map_dfr(.x = list.files("../Downloads/DataPacks", full.names = TRUE),
                    .f = ~ tame_dp(.x, map_names = FALSE))
  df_all <- get_names(df_all)
```

`tame_join` *Join TST output with MSD output*

Description

Join TST output with MSD output

Usage

```
tame_join(tst_filepath, msd_filepath, fy_as_str = TRUE, map_names = FALSE)
```

Arguments

`tst_filepath`	file path to the Target Setting Tool importing, must be .xlsx
`msd_filepath`	filepath to the latest PSNUxIM MSD for corresponding OU
`fy_as_str`	should FY be converted to a string (2025 > FY25) for Tableau? (default = TRUE)
`map_names`	import names from DATIM (OU, mechanism, partner) associated with mech_code when working with PSNUxIM (default = FALSE)

Value

dataframe that combines targets from the TST with corresponding historic results/targets from MSD

Examples

```
#TST file path
  tst_path <- "../Downloads/DataPack_Jupiter_20500101.xlsx"
# MSD filepath
 msd_path <- "../Data/MER_Structured_TRAINING_Datasets_PSNU_IM_FY59-61_20240215_v1_1.zip"

#run join function
  df_join <- tame_join(tst_path, msd_path)

#run join function without converting the fiscal year to a string (used in Tableau)
  df_join <- tame_join(tst_path, msd_path, fy_as_str = FALSE)

#run join function with PSNUxIM & map on mechanism info to TST dataframe
  df_join <- tame_join(tst_path, msd_path, map_names = TRUE)
```

---

tame_plhiv *Export Tidy PLHIV data from Target Setting Tool*

Description

Deprecated. Use ‘tame_subnat‘ instead.

Usage

```
tame_plhiv(filepath)
```

Arguments

filepath	file path to the Target Setting Tool importing, must be .xlsx

See Also

Other primary: tame_dp(), tame_subnat()

Examples

```
#DP file path
  path <- "../Downloads/DataPack_Jupiter_20200218.xlsx"
#read in Target Setting Tool
  df_subnat <- tame_subnat(path)
```

tame_subnat *Export Tidy SUBNAT data from Target Setting Tool*

Description

tame_subnat is a sister function to tame_dp, which readings in the SUBNAT and PLHIV data from the Target Setting Tool and munging in into a tidy data frame to make it more usable to interact with the data than the way it is stored in the Target Setting Tool. **Given the changes to the Target Setting Tool each year, the function only works going back to COP21.**

Usage

```
tame_subnat(filepath)
```

Arguments

filepath	file path to the Target Setting Tool importing, must be .xlsx

See Also

Other primary: tame_dp(), tame_plhiv()

Examples

```
#DP file path
  path <- "../Downloads/DataPack_Jupiter_20200218.xlsx"
#read in Target Setting Tool
  df_subnat <- tame_subnat(path)
```

Index