Package 'jointcenteR'

January 7, 2025

Title Perform routine Joint Center tasks

Version 0.0.0.9000

Description Makes routine tasks easier, from loading data to running tabs and exporting them to excel. Loads American Community Survey datasets into global environment. Splits multiyear ACS file into manageable dataframes. Applies ACS and JCHS variable labels. Processes common public datasets, like New Residential Construction, Housing Vacancy Survey, and HUD AHAR. Calculates weighted and unweighted estimates with counts and shares or medians for large survey datasets. Inflates continuous variables to target year dollars using CPI-U All Items, and inflates rents to target year dollars using CPI-U Less Shelter. Exports tabs to an excel workbook. Splits text document into spreadsheet rows for facthchecking.

Imports dplyr,
Hmisc,
lubridate,
openxlsx,
readr,
rlang,
stringr,
tidycensus,
tidyquant,
tidyr

Suggests testthat

License `use_mit_license()`

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

R topics documented:

`acs22` *random sample of 2022 American Community Survey PUMS*

Description

Household data for a random sample of 300 observations taken from the 2022 American Community Survey Public Use Microdata 1-Year Estimates.

Usage

```
acs22
```

Format

A data frame with columns:

serialno Unique household idnetifier

region Region of the country the household lives in

ten Household tenure

dis Disability status of householder

hincp Household income

rntp Monthly contract rent

wgtp Household weight

Source

U.S. Census Bureau, 2022 American Community Survey 1-Year PUMS.

Examples

```
data(acs22)
```

add_ACSlabels *Add labels to ACS files*

Description

Uses ACS PUMS data dictionary to add labels to household plus file.

Usage

```
add_ACSlabels(df, dict_url)
```

Arguments

df	A dataframe created from load_acs of a single ACS year.
dict_url	The URL of the data dictionary .csv file that corresponds to the df year.

Value

Dataframe with ACS factor variables labeled.

Note

Labels are for standard ACS variables only. JCHS variables will need to be labeled separately.

Examples

```
dict_url <-
"https://www2.census.gov/programs-surveys/acs/tech_docs/pums/data_dict/
 PUMS_Data_Dictionary_2023.csv"
load_acs(2023)
acs2023 <- add_ACSlabels(acs2023, dict_url)
acs2023 |>
  filter(as.integer(ten) >2) |>
  group_by(ten) |>
  summarise(tot = sum(wgtp))
```

add_JCHSlabels *Add JCHS variable labels to ACS files*

Description

Add JCHS variable labels to household plus file.

Usage

```
add_JCHSlabels(df)
```

Arguments

df	A dataframe created from load_acs of a single ACS year.

Value

Dataframe with ACS factor variables labeled.

Note

Labels are for standard JCHS variables only. Use add_ACSlabels for other ACS labels. Created for 2023; check variables for changes before using for prior years.

Examples

```
load_acs(2023)
acs2023_l <- add_JCHSlabels(acs2023)
acs2023_l |> filter(as.integer(tenurecat) < 3) |> group_by(tenurecat) |> summarise(tot = sum(wgtp))
```

---

export_table *Export tables to workbook*

Description

Exports tibbles and named objects to a worksheet of the same name in either a new or existing workbook

Usage

```
export_table(tab, input_wb, output_wb)
```

Arguments

tab	The named tibble or object to write.
input_wb	The .xlsx workbook to read, including the file path, in quotations.
output_wb	The .xlsx workbook to write to, including the file path, in quotations. If the output is the same as the input_wb, the input_wb will be overwritten or added to.

Value

Workbook with sheets named after input tab.

Note

Existing tab will be overwritten if worksheet already exists.

Examples

```
tenTable <- acs2023 |> tab(tensimp, w = wgtp)
export_table(tenTable, "C:/Data/tabswb.xlsx", "C:/Data/tabswb.xlsx")
```

factcheck_format *Start factchecking spreadsheet*

Description

Splits chapter at sentence breaks and puts each new sentence on a new row

Usage

```
factcheck_format(txtfile)
```

Arguments

txtfile	The full path and name of the text file in quotations.

Value

A dataframe where each line is a sentence or header.

Note

Word files should be saved to .txt using Unicode (UTF-8) encoding. Splits based on occurrence of ". " or a carriage return. Can use with jointcenteR::export_table() to export to its own tab of a spreadsheet.

Examples

```
chapter1 <- factcheck_format("C:/Data/testtext.txt")
```

get_hvs *Get Housing Vacancy Survey data*

Description

Pulls and organizes Housing Vacancy Survey data using tidyquant

Usage

```
get_hvs()
```

Value

A tibble of HVS variables for each quarter from 2Q2001 through most recent available.

Note

Adapted from Len Kiefer: http://lenkiefer.com/2017/09/18/a-tidyquant-um-of-solace/

Examples

```
hvs <- get_hvs()
```

get_nrc *Get New Residential Construction data*

Description

Pulls and organizes New Residential Construction data using tidyquant

Usage

```
get_nrc(type = "monthly")
```

Arguments

type	Specifies the interval of interest, either "monthly" or "annual"

Value

A tibble of HVS variables for each quarter from 2Q2001 through most recent available.

Note

Adapted from Len Kiefer: http://lenkiefer.com/2017/09/18/a-tidyquant-um-of-solace/ Annual counts are totals of monthly NSA values, except annual under construction counts are the value in December of a given year. If current year is incomplete, it is omitted from annual table. Monthly counts are seasonally adjusted annual rates, except under construction counts are the seasonally adjusted rate at end of month (not annualized).

Examples

```
# pull annual data
nrc_a <- get_nrc(type = "annual") |>
  # keep vars for year and units in buildings with 2+ units
  select(year, contains('_2pl')) |>
  # keep years in range of interest
  filter(year >= 2000)

# pull monthly data to calculate ytd SA average
nrc_m <- get_nrc("monthly") |>
  # reshape to make calculations easier
  pivot_longer(!c("year", "month"), names_to = "var", values_to = "value") |>
  # keep current year & mf vars
  filter(stringr::str_detect(var, '_2pl') & year == 2024) |>
  group_by(year, var) |>
  mutate(avg = mean(value)) |>
  filter(month == 1) |>
  select(-month, -value) |>
  pivot_wider(names_from = var, values_from = avg)

#combine annual and ytd dfs
nrc <- rbind(nrc_a, nrc_m)
```

get_permits *Get Building Permits Survey*

Description

Cleans census URLs to directly download and format annual county or state permit datasets

Usage

```
get_permits(geography, year)
```

Arguments

geography	Select the geography to pull. Valid inputs are "state" or "county"
year	The year to extract, starting with 1980 for states and 1990 for counties.

Value

A tibble with 31 columns.

Note

Adapted from Chris Goodman: "https://gist.github.com/cbgoodman/8b78153e93148cdff370e3c6f3629ade.js" Can extract multiple years at once using purrr::map (see examples).

Examples

```
# pull state permits for 2023 and select unit variables
sp23 <- get_permits("state", 2023)
sp23 |>
  select(year, region, state_name, starts_with("units_") & ends_with("unit"))

# pull multiple years at once and bind into long df
sp_list <- map(2003:2023, get_permits, geography = "state")
sp <- bind_rows(sp_list)
```

inflate_ai *Inflate with CPI-U All Items*

Description

Inflate values from reference to target dollars using CPI-U All Items, annual NSA

Usage

```
inflate_ai(initial_amount, reference_year, target_year)
```

Arguments

initial_amount	A numeric value or vector to be inflated.
reference_year	The year the values to be inflated are from.
target_year	The year values should be inflated to.

Value

A numeric vector of inflated values.

Note

CPI-U inflation rates are pulled from FRED using tidyquant. Monthly values are averaged for the year.

Examples

```
# inflate value from 2000 dollars to 2023 dollars
inflate_ai(5000, 2000, 2023)

data("acs22")
# inflate household incomes to target year dollars in new variable
acs22 |> mutate(hincp_infl_23 = inflate_ai(hincp, 2022, 2023))
```

---

inflate_ls *Inflate with CPI-U Less Shelter*

Description

Inflate rents from reference to target dollars using CPI-U Less Shelter, NSA.

Usage

```
inflate_ls(initial_rent, reference_year, target_year)
```

Arguments

initial_rent	A numeric value or vector to be inflated.
reference_year	The year the values to be inflated are from.
target_year	The year values should be inflated to.

Value

A numeric vector of inflated values.

Note

CPI-U inflation rates are pulled from FRED using tidyquant. Monthly values are averaged for the year.

Examples

```
# inflate rent from 2000 dollars to 2023 dollars
inflate_ls(500, 2000, 2023)

data("acs22")
# inflate contract rents to target year dollars in new variable
acs22 |> mutate(rntp_infl_23 = inflate_ls(rntp, unique(year), 2023))
```

`load_acs` *Load annual ACS household plus files*

Description

Reads annual ACS household plus files into global environment

Usage

```
load_acs(yr, currentyear = 2023, path = acspath)
```

Arguments

`yr`	The year of the data file to load in
`currentyear`	The most recent year of data available, used for coding variables that don't need to be inflated.
`path`	A path to the location of ACS .csv files.

Value

Dataframes loaded into global environment named with convention acs2023.

Note

Annual ACS files should be in one location and saved as .csv files. Load multiple years with purrr::map.

Examples

```
acspath <- "C:/Data/ACS/"
# single year
load_acs(2022, currentyear = 2023, path = acspath)

# multiple years
purrr::map(c(2019, 2021:2023), load_acs, path = acspath)
```

load_met *Load annual ACS metro files*

Description

Reads annual ACS metro files into global environment

Usage

```
load_met(yr, path = acspath)
```

Arguments

yr	The year of the data file to load in
path	A path to the location of metro .csv files.

Value

Dataframes loaded into global environment named with convention met2023.

Note

Annual metro files should be in one location and saved as .csv files. Load multiple years with purrr::map.

Examples

```
acspath <- "C:/Data/ACS/"
# single year
load_met(2022, path = acspath)

# multiple years
purrr::map(c(2019, 2023), load_met, path = acspath)
```

load_mult *Load JCHS multiyear ACS file*

Description

Reads JCHS multiyear ACS file into global environment

Usage

```
load_mult(path = acspath, filename = multfilename)
```

Arguments

path	A path to the location of multiyear .csv files.
filename	The name of the file to load.

Value

Dataframes loaded into global environment named multiyear.

Note

Multiyear files should be in one location and saved as .csv files. Splitting the multiyear file for renters or owners will help them load faster if you are routinely using one or the other.

Examples

```
acspath <- "C:/Data/ACS/"
multfilename<- "ACS_multiyear_hhplus_RentersPlusVacant.csv"
load_mult()
load_mult(acspath, multfilename)
```

`process_ahar` *Clean AHAR data*

Description

Puts HUD AHAR data into a usable table format with sheltered, unsheltered, and overall homelessness counts by state and for the US

Usage

```
process_ahar(year, file)
```

Arguments

`year`	Year of data to pull, from 2007 to 2023.
`file`	File path and name of saved .xlsx point-in-time count file.

Value

A dataframe of sheltered, unsheltered, and overall homelessness counts by state and for the US for years selected.

Note

HUD posts the data as an xlsb, which is very hard to work with. First download the data from https://www.huduser.gov/portal/sites/default/files/xls/ 2007-2023-PIT-Counts-by-CoC.xlsb and save locally as an xlsx. Use 2021 data with caution: unsheltered and overall counts were not complete.

Examples

```
file <- "C:/Data/2007-2023-PIT-Counts-by-State.xlsx"
# single year
process_ahar(2023, file)

# multiple years
yrs <- 2007:2023
ahar_st <- map(yrs, ahar_table, file = file) |>
  reduce(left_join, by = "State")
```

---

split_mult *Load JCHS multiyear ACS file*

Description

Reads JCHS multiyear ACS file into global environment

Usage

```
split_mult(yr, multdf = multiyear)
```

Arguments

yr	The year to extract.
multdf	The name of the dataframe to split. If using load_mult first, the name should be multiyear.

Value

Dataframes loaded into global environment named with convention acs2001.

Note

Can extract multiple years at once using purrr::map (see examples). Recommend running rm(multiyear) once years of interest are extracted.

Examples

```
acspath <- "C:/Data/ACS/"
multfilename<- "ACS_multiyear_hhplus_RentersPlusVacant.csv"

# split out multiple years
load_mult()
purrr::map(c(2001:2005), split_mult)

split_mult(2006)
```

tab *Count and share tables*

Description

Calculate weighted and weighted tables using up to three variables

Usage

```
tab(df, var, var2 = NULL, var3 = NULL, weight = NULL)
```

Arguments

df	A dataframe containing the variable of interest and weight vector
var, var2, var3	A categorical variable to tabulate.
weight	A numeric weight variable. If no weight is provided, the tabulation is unweighted.

Value

A tibble of weighted or unweighted sums and shares.

Note

The first two variables entered are the grouping variables for calculating shares.

Examples

```
data("acs22")
# calculate the number and share of households by region, tenure, and
disability using household weights
tab(acs22, region, ten, dis, wgtp)

# unweighted observations
tab(acs22, region, ten, dis)
```

wtd_med *Grouped medians*

Description

Calculate weighted and weighted medians with up to one grouping variable

Usage

```
wtd_med(df, var, var2 = NULL, weight = NULL)
```

Arguments

df	A dataframe containing the variable(s) of interest and weight vector
var	A continuous variable to tabulate median.
var2	A categorical variable to group by.
weight	A numeric weight variable. If no weight is provided, the tabulation is unweighted.

Value

A tibble of weighted or unweighted medians.

Note

The second variable entered is the grouping variable. Does not currently work in combination with the mutate or summarise function. To calculate medians when this doesn't work, use Hmisc::wtd.quantile(var, probs=0.5).

Examples

```
data("acs22")
# calculate the weighted median household income by tenure
wtd_med(acs22, ten, hincp, wgtp)

#returns unweighted medians if no weight if provided
wtd_med(acs22, ten, hincp)
```

Index