Documentation - Census data cleaning

Owen Forbes

11/05/2023

Documentation Manual for TableBuilder Data Cleaning Code

This code is a set of functions in R designed to clean census data from TableBuilder. The data cleaning process involves several steps, from data loading to formatting column names to saving the clean data. This guide explains each part of the code and how to use it.

Code Overview

The script includes two main functions: `TB_Census_cleaning_fn_individual` and `TB_Census_cleaning_fn_dwelling`. These functions process individual-level datasets and dwelling-level datasets, respectively.

Function Input Parameters

The functions have the following input parameters:

- `data_file_base`: stem for file name. (string)
- `data_item_name`: name for variable in cleaned datasheet. (string)
- `calendar_year`: one of 2006,2011,2016,2021. (numeric)
- `code_or_name`: if geographies are exported as name, then "name" will get them converted to codes. (string)
- `census_tag`: 4-letter code for census variable. (string)
- `census_filter_col_name`: desired formatting for column name of census variable. (string)
- `claire_redownload`: if FALSE, year after geography in file name, if TRUE, year before geography. (boolean)

Code Snippets and Functionality

Some selected key pieces are presented below with explanation. Comments throughout the cleaning code provide further information on other steps.

```
data_temp <- read.csv(data_temp_name, skip = 8, row.names = NULL,
    na.strings = c("", "NA"), check.names = FALSE)
```

Data Loading and Initial Cleaning This line reads the CSV file, skipping the first 8 rows which are considered as the header from the TableBuilder export. Empty cells are assigned NA so they can be filled later. The `check.names=FALSE` argument stops `read.csv` from editing the column names.

```
colnames(data_temp) <- colnames(data_temp)[2:ncol(data_temp)]
data_temp <- data_temp[, 1:(ncol(data_temp) - 1)]
```

Correcting Column Names These lines correct issues with header formatting by adjusting column names and removing the added NA column.

```
trailing_rows <- which(is.na(data_temp[, ncol(data_temp)]))
data_temp <- data_temp[-trailing_rows, ]
```

Deleting Trailing Rows These lines identify rows at the end of the data with NA in the final column and remove them.

```
data_temp <- data_temp %>%
    fill(everything(), .direction = "down")
```

Filling Down Variables This line uses the `fill()` function from `tidyr` to fill down variables. This is useful for dealing with NA values that are meant to take the value of the previous row.

```
names(data_temp)[grepl(geog_list[i], names(data_temp)[])] <- paste0(geog_list[i],
    "_CODE_", calendar_year)
```

Fixing Column Names This line changes the column name for geography up to the state level.

```
data_temp[, geo_column_index] <- relevant_codes[match(data_temp[,
    geo_column_index], relevant_names)]
```

Replacing Geography Names with Codes If the `code_or_name` parameter is set to "name", this line replaces geography names with corresponding codes.

```
write.csv(data_temp, file = paste0(interim_folder, data_file_base,
    "_", geog_list[i], "_", calendar_year, "_INTERIM.csv"), row.names = FALSE)
```

Saving Clean CSV The cleaned data is saved as a CSV file.

How to Use

1. Update the file paths in the script to the relevant local directory on your system.
2. Call the function with the appropriate arguments. For example:

```
setwd("/Users/Current/OneDrive - Queensland University of Technology/General - ACWA_QUT/Data_Collections

TB_Census_cleaning_fn_individual(data_file_base = "census_year12",
    data_item_name = "completed_year12", calendar_year = 2006,
    code_or_name = "code", census_tag = "HSCP", census_filter_col_name = "hscp_highest_year_of_school_c
```