Data Cleaning Script

2023-09-06

Authors 2023

Diego Perez Ruiz has written this code in 2021/2022.

A few minor amendments by Wendy Olsen help us access data for different age groups, years, or sex.

Furthermore, the advisor on all the materials is Arkadiusz Wisniowski, and the research assistant in 2021/22 was Madhu Chauhan.

We thank the funder, University of Manchester - School of Social Sciences.

Data Cleaning - Script 1

```
# Clear the R environment
rm(list = ls())

# Load necessary packages
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

In this section, the code starts by clearing the R environment to remove any existing variables or objects. Then, it loads the `tidyverse` package, which is a collection of popular packages in the R ecosystem used for data manipulation, visualization, and analysis.

```
# Define a custom '%notin%' operator
`%notin%` <- Negate(`%in%`)
```

Here, a custom `%notin%` operator is defined using the `Negate` function from base R. This operator is used to check if an element is not in a vector or list, as opposed to the `%in%` operator which checks if an element is in a vector or list. The `setwd` function is used to set the working directory to a specific path. In this case, it sets the working directory to the specified path.

```r
# Set the working directory to the specified path
setwd("/Users/user/Dropbox/Mac (2)/Desktop/Workshop Files/Data Files")

# Load data from files
load("CensusLinking.rda")
Indiaraw <- readRDS("IndiaPLFS201718.rds")
sample_sizes <- readRDS("Sample_Sizes.rds")
```

Here, data is loaded from three files. The `load` function loads data from an RDA file named "CensusLinking.rda," and the `readRDS` function reads data from two RDS (R Data Serialization) files, "IndiaPLFS201718.rds" and "Sample_Sizes.rds."

```r
set.seed(123456)
N <- 35000
Indiaraw <- Indiaraw[sample(nrow(Indiaraw), N), ]
```

This code sets a random seed for reproducibility using `set.seed`, then it randomly samples 35,000 rows from the `Indiaraw` dataset. This random sample is taken from the `Indiaraw` dataset, presumably to reduce the dataset size for further analysis.

```r
# Data preprocessing
India_Employment <- Indiaraw %>%
  filter(sex != "3" & ( age >= "14" & age <= "24"))  %>%
  mutate(
    distnew = str_pad(district, 2, pad = "0"),
    statenew = str_pad(state, 2, pad = "0"),
    distcode = paste0(statenew, '', distnew)
  )
```

This section of code performs data preprocessing on the `Indiaraw` dataset. It filters the data to include only rows where the "sex" column is not equal to "3" and the "age" column is less than or equal to "24." It also creates three new columns (`distnew`, `statenew`, and `distcode`) based on the existing "district" and "state" columns.

- `distnew` is created by padding the "district" column with leading zeros so that it has two digits.
- `statenew` is created by padding the "state" column with leading zeros so that it has two digits.
- `distcode` is created by concatenating `statenew` and `distnew`.

These transformations are often done for data standardization or aggregation purposes.

```r
# Aggregate data by various criteria
India_Employment_State <- India_Employment %>%
  group_by(statenew) %>%
  summarise(Freq_State = sum(medwork))
```

This code aggregates data at the state level. It groups the `India_Employment` dataset by the `statenew` column and calculates the sum of the "medwork" column within each group. The results are stored in a new data frame called `India_Employment_State` with a summary column named `Freq_State`.

```r
India_Employment_District <- India_Employment %>%
  group_by(distcode) %>%
  summarise(Freq_Medwork = sum(medwork))
```

This section performs a similar aggregation, but at the district code (`distcode`) level. It groups the `India_Employment` dataset by the `distcode` column and calculates the sum of the "medwork" column within each group. The results are stored in a new data frame called `India_Employment_District` with a summary column named `Freq_Medwork`.

```
India_Employment_Males <- India_Employment %>%
  group_by(distcode) %>%
  summarise(Prop_Males = sum(female))
```

This code aggregates data at the district code (`distcode`) level as well, but this time it calculates the sum of the "female" column within each group. The results are stored in a new data frame called `India_Employment_Males` with a summary column named `Prop_Males`.

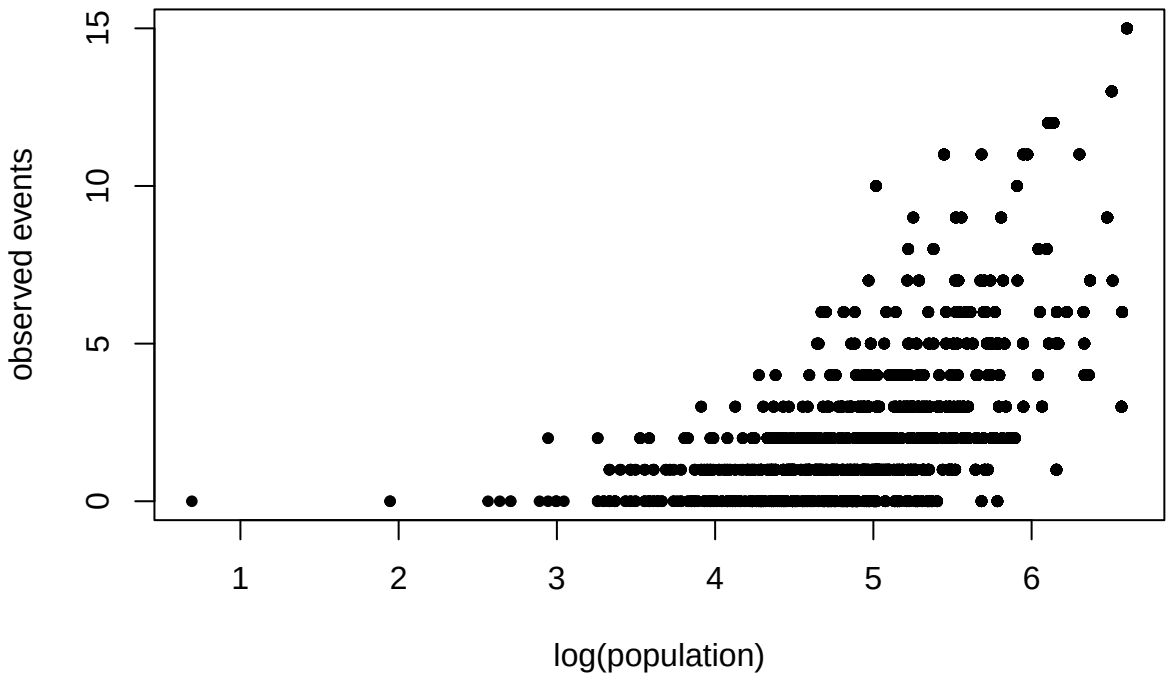
```
India_Employment_Rural <- India_Employment %>%
  group_by(distcode) %>%
  summarise(Prop_Rural = sum(rural))
```

This section also aggregates data at the district code (`distcode`) level, calculating the sum of the "rural" column within each group. The results are stored in a new data frame called `India_Employment_Rural` with a summary column named `Prop_Rural`.

```
# Perform left joins to merge data frames
India_Employment <- India_Employment %>%
  left_join(India_Employment_State, by = "statenew") %>%
  left_join(India_Employment_District, by = "distcode") %>%
  left_join(sample_sizes, by = "distcode") %>%
  left_join(India_Employment_Males, by = "distcode") %>%
  left_join(India_Employment_Rural, by = "distcode")
```

In this section, left joins are performed to merge the data frames created in the previous aggregation steps. The common key for joining is the "distcode" column. The resulting merged data frame is stored back in `India_Employment`.

```
# Create a plot
plot(log(India_Employment$sample_size), India_Employment$Freq_Medwork,
     xlab = "log(population)", ylab = "observed events", pch = 20)
```



This code generates a scatter plot. It plots the logarithm of the "sample_size" column on the x-axis and the "Freq_Medwork" column on the y-axis. Labels for the x-axis and y-axis are provided using `xlab` and `ylab`, and the `pch` argument specifies the point character used in the plot.

```
# Create a new data frame for further analysis
India_Employment_By_District <- India_Employment %>%
  dplyr::select(distcode, Freq_State, Prop_Males, Prop_Rural, sample_size) %>%
  distinct()
```

This code creates a new data frame called `India_Employment_By_District`. It selects specific columns from the `India_Employment` data frame (including

"distcode," "Freq_State," "Prop_Males," "Prop_Rural," and "sample_size") and retains only distinct rows.

```
# Data transformation
India_Employment_By_District <- India_Employment_By_District %>%
  mutate(
    Prop_Males = Prop_Males / sample_size,
    Prop_Females = 1 - Prop_Males,
    Prop_Rural = Prop_Rural / sample_size,
    Prop_Urban = 1 - Prop_Rural
  )
```

This section of code performs data transformation on the `India_Employment_By_District` data frame. It calculates new columns: - `Prop_Males`: It is calculated as the ratio of "Prop_Males" to "sample_size." - `Prop_Females`: It represents the complementary proportion of males (1 - Prop_Males). - `Prop_Rural`: It is

calculated as the ratio of "Prop_Rural" to "sample_size." - `Prop_Urban`: It represents the complementary proportion of rural (1 - Prop_Rural).

These transformations likely involve normalizing proportions relative to the sample size.

```
# Join data frames
India_Sample_Employment <- India_Employment_By_District %>%
  left_join(Key_Censsu2011, by = "distcode")

head(India_Sample_Employment)
```

```
##   distcode Freq_State Prop_Males Prop_Rural sample_size Prop_Females Prop_Urban
## 1     2023         32 0.02666667 0.06000000         150    0.9733333  0.9400000
## 2     3319         68 0.07638889 0.09027778         144    0.9236111  0.9097222
## 3     1404         24 0.05223881 0.07462687         268    0.9477612  0.9253731
## 4     0602         45 0.06930693 0.08910891         101    0.9306931  0.9108911
## 5     0701         16 0.03886398 0.01046338         669    0.9611360  0.9895366
## 6     0830         71 0.01162791 0.02325581          86    0.9883721  0.9767442
##   statecode districtcode        districtname censuscode            district
## 1        20           23 Pashchimi Singhbhum        368 Pashchimi Singhbhum
## 2        33           19           Thanjavur        620           Thanjavur
## 3        14           04           Bishnupur        275           Bishnupur
## 4        06           02              Ambala         70              Ambala
## 5        07           01    North West Delhi         90          North West
## 6        08           30               Baran        128               Baran
##      statename
## 1    Jharkhand
## 2   Tamil Nadu
## 3      Manipur
## 4      Haryana
## 5 NCT of Delhi
## 6    Rajasthan
```

```
dim(India_Sample_Employment)
```

```
## [1] 638  13
```

Here, a left join is performed between the `India_Employment_By_District` data frame and another data frame called `Key_Censsu2011` based on the "distcode" column. The result is stored in a new data frame called `India_Sample_Employment`. The `head` function is then used to display the first few rows of the merged data frame, and `dim` is used to show the dimensions (number of rows and columns) of the resulting data frame.

```
# Save the resulting data frame to an RDA file
save(India_Sample_Employment, file = "India_Employment_withCensus2011_SampleSize.rda")
```

Finally, this code saves the `India_Sample_Employment` data frame to an RDA file named "India_Employment_withCensus2011_SampleSize.rda." This allows the data to be stored and accessed for future analysis without the need to re-run the entire data processing pipeline.