labor_70_95_database

Dor Meir

1/21/2020

70-95 database building

This program builds the 70-95 databaste from all different dta files. This markdown file (The program script) contains 3 chunks, to be run in their order (or all atonce): 1. The first chunk contains all specific pre-defined hyperparameter and values that the program uses, such as the files locations, the name of the id variable, etc. *if something changes in those values - you need to change things in this chunk!* This chunk also loads the packages which are being used in the program. 2. The second chunk contains all the helper functions who are called in by each other and in the third chunck, in order to build the database. *unless the files structures changes dramatically, you shouldn't change this chunk.* 3. The final chunk runs everything, using the hyperparamaters of the first chunk and the helper function of the second chunk, and saves the results to both csv and Rdata workspace files. *If you want to do extra things, you might write them here or add another chunk*.

The running of this file takes LONG time, so… *Goodluck!*

1. Define hyperparameters and Load pacakges

```
print("Please verify all the hyperparameters below are correct:")
```

```
## [1] "Please verify all the hyperparameters below are correct:"
```

```
dta_files_location <- "D:/Research/NeelieBT_Research_13800207/Data/Original Data/1.12.2019/test/"
#dta_files_location <- "D:/Research/NeelieBT_Research_13800207/Data/Original Data/1.12.2019/"
id_variable_name <- "MisparZehut_fic"
memory_limit <- 10000000
extra_packages_path <- "D:/Research/3.6"
min_year_possible_of_all_survey <- 1940
max_year_possible_of_all_survey <- 2030
mitzav_file_contains_the_word <- "mitzav"
dataframe_name <- "df"
final_database_file_name <- "70_95_df"
final_database_file_Rdata<-paste0(dta_files_location,final_database_file_name,".Rdata")
final_database_file_csv<-paste0(dta_files_location,final_database_file_name,".csv")
print("We're assuming two assumptions:")
```

```
## [1] "We're assuming two assumptions:"
```

```
print("1. All files are dta stata files")
```

```
## [1] "1. All files are dta stata files"
```

```
print("2. mitzav file contain two tests: on 5th grade and on 8th grade")
```

```
## [1] "2. mitzav file contain two tests: on 5th grade and on 8th grade"
```

```
# Set the extra packages path:
.libPaths(c(.libPaths(),extra_packages_path))
# Load packages:
library(readstata13)
```

```
## Warning: package 'readstata13' was built under R version 3.6.2
```

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.6.2
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(data.table)
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:data.table':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
```

```
## The following object is masked from 'package:base':
##
##     date
```

```
library(stringr)
```

```
## Warning: package 'stringr' was built under R version 3.6.2
```

```
library(rio)
#readline(prompt="Check that all parameters are correct and that the packages were loaded correctly, and than press [enter]
 to continue")
# set hyperparametes:
print("Switching to the origional dta file folder so we can import them to R:")
```

```
## [1] "Switching to the origional dta file folder so we can import them to R:"
```

```
knitr::opts_chunk$set(root.dir = dta_files_location)
setwd(dta_files_location)
getwd()
```

```
## [1] "D:/Research/NeelieBT_Research_13800207/Data/Original Data/1.12.2019/test"
```

```
# we need all the memory we can get for this:
memory.limit(size=memory_limit)
```

```
## [1] 1e+07
```

```
# our list of files:
list_of_files <- list.files(pattern = ".dta")
cat("The 70-95 file in the directory are:")
```

```
## The 70-95 file in the directory are:
```

```
cat(list_of_files)
```

```
## mechinot.dta ramat_haredim.dta seder_leda.dta
```

2. Define Helper functions

```
setwd(dta_files_location)

# find id_variable_column_number
id_variable_column_number_file_name <- function(file_name){
     return(which(colnames(get(file_name))==id_variable_name))
  }
id_variable_column_number_file <- function(file){
     return(which(colnames(file)==id_variable_name))
  }
any_variable_column_number <- function(file_name,variable){
     return(which(colnames(get(file_name))==variable))
  }

move_to_last <- function(file,column){
    file[c(setdiff(names(file),column),column)]
  }

# move any column to first
move_column_to_first <- function(file,column){
      file[c(column,setdiff(names(file),column))]
  }

# move id column to first column
move_id_to_first <- function(file){
      file[c(id_variable_name,setdiff(names(file),id_variable_name))]
   }


# Import a file's id column only, and sort by id number
file_id_column_to_variable <- function(file_name){
    assign(paste0(file_name),move_to_first(import((file_name),id_variable_name), select.cols = id_variable_name),envir = .Gl
obalEnv)
    id_column_number <- id_variable_column_number_file_name(file_name)
    setorderv(get(file_name),id_variable_name)
  }


# Find the Modes (most common value) function. for the first mode, use max().
Mode <- function(x){
  ux <- unique(x)
  ux[which.max(tabulate(match(x,ux)))]
  }

# max function that ignore NA
```

```
my.max <- function(x){
  x <- as.numeric(unlist(x))
  ifelse (!all(is.na(x)),max(x,na.rm=T),NA)
}

# Find the year column function (4 characters, mode is between 1850 and 2100) using the Mode function from above
find_year_column <- function(file){
  number_of_chars <- nchar(lapply(file, my.max))
  potential_number_of_year_columns <- list()
  i <-1
  for (num in number_of_chars){
     if  (num==4){
       potential_number_of_year_columns[length(potential_number_of_year_columns)+1] <- i
     }
    i=i+1
  }
potential_number_of_year_columns <- unlist(potential_number_of_year_columns)
  if (length(potential_number_of_year_columns)>1){
    for (column in potential_number_of_year_columns){
      cat("\nColumn",column,"has 4 characters, checking the most common value:")
      mode_file_column <- my.max(Mode(file[column]))
      cat("\nThe most common value is ",mode_file_column)
      if ((mode_file_column>min_year_possible_of_all_survey)&(mode_file_column<max_year_possible_of_all_survey)){
        cat("\n",mode_file_column, "is between", min_year_possible_of_all_survey, "and",max_year_possible_of_all_survey,", s
o we'll set column no.", column, "as the year column.")
        year_column_name <- colnames(file)[column]
        cat("The year column name is:",year_column_name)
        break
      }
    }
    potential_number_of_year_columns <- year_column_name
  }
  return(unlist(colnames(file[potential_number_of_year_columns])))
  }

# Copy df to new column of variable with proper year
copy_df_to_new_column_by_year <- function(get_file,rows_to_copy,columns_to_copy,years_to_copy){
  # copy the values:
  for (i in seq_along(rows_to_copy)){
    for (j in seq_along(columns_to_copy)){
    column_to_paste <- paste0(columns_to_copy[j],years_to_copy[i])
    get_file[rows_to_copy[i],column_to_paste]=get_file[rows_to_copy[i],columns_to_copy[j]]
    }
    }
  return(get_file)
```

```
  }

# expand a non unique id df (with several rows for some ids) to unique id (one row for each id), by adding columns for diffe
rent years
convert_non_unique_ids_file_to_unique <- function(file_name){
  get_file <- get(file_name)
  year_column <- find_year_column(get_file)
  print("We've got the year column, shaping the file to unique id, this will take a while...")
  year_column_number <- any_variable_column_number(file_name,year_column)
  list_of_years <- sort(unique(get_file[[year_column]]))
  number_of_years <- length(sort(unique(get_file[[year_column]])))
  origional_columns <- colnames(get_file)
  id_column_number <- id_variable_column_number_file_name(file_name)
  columns_to_add <- unlist(lapply(colnames(get_file[-c(id_column_number,year_column_number)]),paste0,y=list_of_years))
  # Add new empty columns
  get_file[,columns_to_add] <- NA
  # copy the non NA rows to the new empty columns
  years_to_copy <- get_file[[year_column]][which(!is.na(get_file[[year_column]]))]
  rows_to_copy <- which(!is.na(get_file[[year_column]]))
  columns_to_copy <- origional_columns[-c(id_column_number,year_column_number)]
  get_file <- copy_df_to_new_column_by_year(get_file,rows_to_copy,columns_to_copy,years_to_copy)
  # delete the year column and the origional columns who were coppied:
  get_file[c(year_column,columns_to_copy)] <- NULL
  # squeeze all non-unique id rows into one row
  get_file = aggregate(get_file[-id_column_number], by = list(get_file
  [[id_variable_name]]), FUN = mean, na.rm=TRUE)
  # Convert Nan's to NA
  get_file[is.na(get_file)]=NA
  # Rename the id_variable_name back to it's name:
  colnames(get_file)[1] <- id_variable_name
  # assign the new df to the old one:
  assign(paste0(file_name),get_file,envir = .GlobalEnv)
  }

# Return TRUE if non-unique id file
is_unique_id_file <- function(file_name){
  percent_of_non_uniques_lines <- 1-length(unique(get(file_name)[[id_variable_name]]))/length(get(file_name)[[id_variable_na
me]])
  if (percent_of_non_uniques_lines==0){
    return(TRUE)
  }
  return(FALSE)
  }

# Split the mitzav file in the two files containing two exams
```

```
split_mitzav_file <-function(file_name){
  file <- get(file_name)
  cat("\nThe file is a mitzav file, splitting it to 2 files:")
  mitzav_splitted_file_number_of_column = length(colnames(file)[-1])/2
  mitzav1_cols <- colnames(file[1:(1+mitzav_splitted_file_number_of_column)])
  mitzav1_name <- (paste0(file_name,1))
  assign(mitzav1_name,file[mitzav1_cols],envir = .GlobalEnv)
  mitzav2_cols <- c(id_variable_name,setdiff(colnames(file),mitzav1_cols))
  mitzav2_name <- (paste0(file_name,2))
  assign(mitzav2_name,file[mitzav2_cols],envir = .GlobalEnv)
  # add the files to list_of_files
  origional_mitzav_file_index <- match(file_name,list_of_files)
  ## file 1:
  list_of_files[origional_mitzav_file_index]=paste0(file_name,1)
  ## file 2:
  list_of_files <- append(list_of_files,paste0(file_name,2),origional_mitzav_file_index)
  # fix mitazv files if they have duplicate columns
    if (is_unique_id_file(mitzav1_name)==FALSE){
     cat("\n",mitzav1_name, " has non-unique ID, fixing it now: ")
     convert_non_unique_ids_file_to_unique(mitzav1_name)
    }
    if (is_unique_id_file(mitzav2_name)==FALSE){
       cat("\n",mitzav2_name, " has non-unique ID, fixing it now: ")
       convert_non_unique_ids_file_to_unique(mitzav2_name)
     }
  }

add_file_name_to_columns <-function(file_name){
    get_file<-get(file_name)
    id_column_number <- which(colnames(get_file)==id_variable_name)
    colnames(get_file)<-paste(colnames(get_file),file_name,sep ="_")
    colnames(get_file)[id_column_number]<-id_variable_name
    assign(paste0(file_name),get_file,envir = .GlobalEnv)
}

# Import a file, sort by id number, move id column to first only if not already first, and
file_to_variable <- function(file_name){
  assign(paste0(file_name),import(file_name),envir = .GlobalEnv)
  # if id column is not first, make it first
  if  (id_variable_column_number_file(get(file_name))!=1){
    move_id_to_first(get(file_name))
  }
  # Order data by id number
  setorderv(get(file_name),id_variable_name)
  # If it's mitzav file, split it into the two mitav tests:
```

```
    if (str_extract_all(file_name,mitzav_file_contains_the_word,simplify = FALSE)==mitzav_file_contains_the_word) {
    split_mitzav_file(file_name)
  }
  else{
    # if the file has several lines for each id (with different years for each line), convert it into a unique id file using
the creation of new columns:
     if (is_unique_id_file(file_name)==FALSE){
       cat("\n",file_name, " has non-unique ID, fixing it now: ")
       convert_non_unique_ids_file_to_unique(file_name)
     }
  }
  add_file_name_to_columns(file_name)
}

merge_file_to_df <-function(file_name,df){
 cat("\nImportig the file:",file_name)
 file_to_variable(file_name)
 cat("\nMerging to df...")
 get_file<-get(file_name)
 df <- full_join(x = df, y = get_file, by = id_variable_name)
 assign(dataframe_name,df,envir = .GlobalEnv)
 #assign(paste0(df),df,envir = .GlobalEnv)
 cat("\nDone! The memory usage is:", memory.size())
 rm(list=ls(pattern=file_name),envir = .GlobalEnv)
 gc()
 cat("\nAfter cleanup, the memory usage is:", memory.size()," out of", memory.limit())
 Sys.sleep(10)
}
```

3. Build 70-95 Database

```
setwd(dta_files_location)

print("Using all the helper function above, let's import, prepare and merge all the files into one big database (while delet
ing each file):")
```

```
## [1] "Using all the helper function above, let's import, prepare and merge all the files into one big database (while dele
ting each file):"
```

```
# 1. Start the merging by defining the first file as df:

file1 <- list_of_files[1]
cat("\nMerging ", file1, " to df...")
```

```
##
## Merging  mechinot.dta  to df...
```

```
file_to_variable(file1)
df <- get(file1)
rm(list=ls(pattern=list_of_files[1]))
gc()
```

```
##            used  (Mb) gc trigger  (Mb) max used  (Mb)
## Ncells  4070042 217.4   10848035 579.4  4088396 218.4
## Vcells 19771949 150.9   34089392 260.1 33810204 258.0
```

```
cat("\nDone handling the first file. After cleanup, the memory usage is:", memory.size())
```

```
##
## Done handling the first file. After cleanup, the memory usage is: 362.83
```

```
# 2. Merge all other files to df, one-by-one using one-to-many merge, and delete the old files
lapply(list_of_files[-1], merge_file_to_df, df = df)
```

```
##
## Importig the file: ramat_haredim.dta
## Merging to df...
## Done! The memory usage is: 563.3
## After cleanup, the memory usage is: 538.33  out of 1e+07
## Importig the file: seder_leda.dta
## Merging to df...
## Done! The memory usage is: 1085.07
## After cleanup, the memory usage is: 613.41  out of 1e+07
```

```
## [[1]]
## NULL
##
## [[2]]
## NULL
```

```
print("We're Done! we're not allowd to print a sample of the unified df, but there are the df's columns:")
```

```
## [1] "We're Done! we're not allowd to print a sample of the unified df, but there are the df's columns:"
```

```
print(colnames(df))
```

```
## [1] "MisparZehut_fic"             "maslul_mechina_mechinot.dta"
## [3] "mech_mechinot.dta"           "seder_leda_seder_leda.dta"
```

```
print("Saving the final df into Rdata file (with functions, for later work on R), and csv file only the final df (for other
 frameworks:")
```

```
## [1] "Saving the final df into Rdata file (with functions, for later work on R), and csv file only the final df (for other
frameworks:"
```

```
save.image(file = final_database_file_Rdata)
write.csv(df,final_database_file_csv, row.names = FALSE)
```