
LONGITUDINAL DATA CLEANING - A CASE OF THE NLSY79 DATA

A PREPRINT

Dewi Amaliah *

Department of Econometric and Business Statistics
Monash University
Clayton, VIC 3168
dama0007@student.monash.edu

Dianne Cook

Department of Econometric and Business Statistics
Monash University
Clayton, VIC 3168
dicook@monash.edu

February 22, 2021

Abstract

Enter the text of your abstract here.

Keywords longitudinal data · data cleaning · data tidying · robust linear model

1 Introduction

2 The NLSY79

3 The NLSY79 Data Cleaning

3.1 Getting and Tyding the Data

The NLYS data is stored in a database and could be downloaded by variables. Several variables are available to be downloaded and could be browsed by index. For the wages data set, we only extracted these variables:

- Education, Training & Achievement Scores
 - Education -> Summary measures -> All schools -> By year -> Highest grade completed
 - * Downloaded all of the 78 variables in Highest grade completed.
- Employment
 - Summary measures -> By job -> Hours worked and Hourly wages
 - * Downloaded all of the 427 variables in Hours worked
 - * Downloaded all of the 151 variables in Hourly wages
 - Both hours worked and hourly wages are recorded by the job, up to five jobs for each id/subject.
- Household, Geography & Contextual Variables

*Use footnote for providing further information about author (webpage, alternative address)—*not* for acknowledging funding agencies. Optional.

- Context -> Summary measures -> Basic demographics
 - * Downloaded year and month of birth, race, and sex variables.

There are two versions of the year and month of birth, i.e. 1979 and 1981 data. We downloaded these two versions.

The downloaded data set came in a csv (NLSY79.csv) and dat (NLSY79.dat) format. We only used the .dat format. It also came along with these files:

- NLSY79.NLSY79: This is the tagset of variables that can be uploaded to the web site to recreate the data set.
- NLSY79.R: This is an R script provided automatically by the database for reading the data into R and convert the variables' name and its label into something more sensible. We utilized this code to do an initial data tidying. It produced two data set, `categories_qnames` (the observations are stored in categorical/interval values) and `new_data_qnames` (the observations are stored in integer form). In this case, we only used the latter.

```
source(here::here("data-raw/NLSY79/NLSY79.R"))
```

`new_data_qnames` is still untidy, where all of the variables for each year and each job being stored in one column, hence the data contains a huge number of columns (686 columns). Thus, the data should be tidied and wrangled first to extract the demographic and employment variables that we want to put in the final data set. We mainly used `tidyr` (Wickham 2020), `dplyr` (Wickham et al. 2020), and `stringr` (Wickham 2019) to do this job.

3.1.1 Tidying demographic variables

Age in 1979, gender, race, highest grade completed (factor and integer), and the year when the highest grade completed are the variables that we want to use in the cleaned data set.

Age in 1979 are derived from year of birth and month of birth variables in the raw data set. The variables have two versions, which are the 1979 version and the 1981 version. We only used the 1979 data and did a consistency check of those years and flag the inconsistent observations.

```
## tidy the date of birth data
dob_tidy <- new_data_qnames %>%
  select(CASEID_1979,
         starts_with("Q1-3_A~")) %>%
  mutate(dob_year = case_when(
    # if the years recorded in both sets match, take 79 data
    'Q1-3_A~Y_1979' == 'Q1-3_A~Y_1981' ~ 'Q1-3_A~Y_1979',
    # if the year in the 81 set is missing, take the 79 data
    is.na('Q1-3_A~Y_1981') ~ 'Q1-3_A~Y_1979',
    # if the sets don't match for dob year, take the 79 data
    'Q1-3_A~Y_1979' != 'Q1-3_A~Y_1981' ~ 'Q1-3_A~Y_1979'),
  dob_month = case_when(
    # if months recorded in both sets match, take 79 data
    'Q1-3_A~M_1979' == 'Q1-3_A~M_1981' ~ 'Q1-3_A~M_1979',
    # if month in 81 set is missing, take the 79 data
    is.na('Q1-3_A~M_1981') ~ 'Q1-3_A~M_1979',
    # if sets don't match for dob month, take the 79 data
    'Q1-3_A~M_1979' != 'Q1-3_A~M_1981' ~ 'Q1-3_A~M_1979'),
  # flag if there is a conflict between dob recorded in 79 and 81
  dob_conflict = case_when(
    ('Q1-3_A~M_1979' != 'Q1-3_A~M_1981') & !is.na('Q1-3_A~M_1981')
    ~ TRUE,
    ('Q1-3_A~Y_1979' != 'Q1-3_A~Y_1981') & !is.na('Q1-3_A~Y_1981')
    ~ TRUE,
    ('Q1-3_A~Y_1979' == 'Q1-3_A~Y_1981') &
    ('Q1-3_A~M_1979' == 'Q1-3_A~M_1981') ~ FALSE,
    is.na('Q1-3_A~M_1981') | is.na('Q1-3_A~Y_1981') ~ FALSE)) %>%
```

```
select(CASEID_1979,
       dob_month,
       dob_year,
       dob_conflict)
```

For gender and race, we only renamed these variables.

```
## tidy the gender and race variables
demog_tidy <- categories_qnames %>%
  select(CASEID_1979,
         SAMPLE_RACE_78SCRN,
         SAMPLE_SEX_1979) %>%
  rename(gender = SAMPLE_SEX_1979,
         race = SAMPLE_RACE_78SCRN)
```

The highest grade completed came with several version in each year. We chose the revised May data because the May data seemed to have less missing and presumably the revised data has been checked. However, there was no revised May data for 2012, 2014, 2016, and 2018 so we just used the ordinary May data.

```
# tidy the grade
demog_education <- new_data_qnames %>%
  as_tibble() %>%
  # in 2018, the variable's name is Q3-4_2018, instead of HGC_2018
  rename(HGC_2018 = 'Q3-4_2018') %>%
  select(CASEID_1979,
         starts_with("HGCREV"),
         "HGC_2012",
         "HGC_2014",
         "HGC_2016",
         "HGC_2018") %>%
  pivot_longer(!CASEID_1979,
               names_to = "var",
               values_to = "grade") %>%
  separate("var", c("var", "year"), sep = -4) %>%
  filter(!is.na(grade)) %>%
  select(-var)
```

In the final data, we only used the highest grade completed ever and derived the year of when the highest grade completed its categorical value. Therefore, we wrangled the highest grade completed in each year to mutate these variables.

```
## getting the highest year of completed education ever
highest_year <- demog_education %>%
  group_by(CASEID_1979) %>%
  mutate(hgc_i = max(grade)) %>%
  filter(hgc_i == grade) %>%
  filter(year == first(year)) %>%
  rename(yr_hgc = year) %>%
  select(CASEID_1979, yr_hgc, hgc_i) %>%
  ungroup() %>%
  mutate('hgc' = ifelse(hgc_i == 0, "NONE", ifelse(hgc_i == 1, "1ST GRADE",
         ifelse(hgc_i == 2, "2ND GRADE", ifelse(hgc_i == 3, "3RD GRADE",
         ifelse(hgc_i >= 4 & hgc_i <= 12, paste0(hgc_i, "TH GRADE"),
         ifelse(hgc_i == 13, "1ST YEAR COL",
         ifelse(hgc_i == 14, "2ND YEAR COL",
         ifelse(hgc_i == 15, "3RD YEAR COL",
         ifelse(hgc_i == 95, "UNGRADED",
         paste0((hgc_i - 12), "TH YEAR COL"))))))))))))
```

Finally, we join all the tidy variables in a data set called `full_demographics`.

```
full_demographics <- full_join(dob_tidy, demog_tidy, by = "CASEID_1979") %>%
  full_join(highest_year, by = "CASEID_1979") %>%
  rename("id" = "CASEID_1979")

head(full_demographics)
```

```
##   id dob_month dob_year dob_conflict      race gender yr_hgc
## 1 1         9      58      FALSE NON-BLACK, NON-HISPANIC FEMALE 1979
## 2 2         1      59      FALSE NON-BLACK, NON-HISPANIC FEMALE 1985
## 3 3         8      61      FALSE NON-BLACK, NON-HISPANIC FEMALE 1993
## 4 4         8      62      FALSE NON-BLACK, NON-HISPANIC FEMALE 1986
## 5 5         7      59      FALSE NON-BLACK, NON-HISPANIC  MALE 1984
## 6 6        10      60      FALSE NON-BLACK, NON-HISPANIC  MALE 1983
##   hgc_i      hgc
## 1 12 12TH GRADE
## 2 12 12TH GRADE
## 3 12 12TH GRADE
## 4 14 2ND YEAR COL
## 5 18 6TH YEAR COL
## 6 16 4TH YEAR COL
```

3.1.2 Tidying employment variables

The employment data comprises of three variables, i.e. total hours of work per week, number of jobs that an individual has, and mean hourly wage. For hours worked per week, initially only one version per job, no choice from 1979 to 1987 (QES-52A). From 1988 onward, when we had more options, we chose the variable for total hours including time spent working from home (QES-52D). However, 1993 did not have all the five D variables (the first one and the last one were missing), so we used QES-52A variable instead. In addition, 2008 only had jobs 1-4 for the QES-52D variable (whereas the other years had 1-5), so we just used these.

```
# make a list for years where we used the "QES-52A"
year_A <- c(1979:1987, 1993)
#function to get the hour of work
get_hour <- function(year){
  if(year %in% year_A){
    temp <- new_data_qnames %>%
      select(CASEID_1979,
              starts_with("QES-52A") &
              ends_with(as.character(year)))}
  else{
    temp <- new_data_qnames %>%
      select(CASEID_1979,
              starts_with("QES-52D") &
              ends_with(as.character(year)))}
  temp %>%
    pivot_longer(!CASEID_1979,
                  names_to = "job",
                  values_to = "hours_work") %>%
    separate("job", c("job", "year"), sep = -4) %>%
    mutate(job = paste0("job_", substr(job, 9, 10))) %>%
    rename(id = CASEID_1979)
}

# list to save the iteration result
hours <- list()
# getting the hours of work of all observations
for(ayear in c(1979:1994, 1996, 1998, 2000, 2002, 2004, 2006, 2008, 2010,
```

```

      2012, 2014, 2016, 2018)) {
    hours[[ayear]] <- get_hour(ayear)
  }
# unlist the hours of work
hours_all <- bind_rows(!!!hours)

```

The same algorithm was also deployed to tidy the rate of wage by year and by ID. The difference is that the hourly rate had only one version of each year. The hours of work and the hourly rate were then joined to calculate the number of jobs that an ID has and their mean hourly wage. Some observations had 0 in their hourly rate, which is considered as invalid value. Thus, their hourly rate set to be N.A.

```

get_rate <- function(year) {
  new_data_qnames %>%
    select(CASEID_1979,
           starts_with("HRP") &
             ends_with(as.character(year))) %>%
    pivot_longer(!CASEID_1979, names_to = "job", values_to = "rate_per_hour") %>%
    separate("job", c("job", "year"), sep = -4) %>%
    mutate(job = paste0("job_0", substr(job, 4, 4))) %>%
    rename(id = CASEID_1979)
}
rates <- list()
for(ayear in c(1979:1994, 1996, 1998, 2000, 2002, 2004, 2006, 2008, 2010,
              2012, 2014, 2016, 2018)) {
  rates[[ayear]] <- get_rate(ayear)
}
rates_all <- bind_rows(!!!rates)
# join hours and rates variable
hours_wages <- left_join(rates_all,
                        hours_all,
                        by = c("id", "year", "job")) %>%
  # set the 0 value in rate_per_hour as NA
  mutate(rate_per_hour = ifelse(rate_per_hour == 0, NA,
                                rate_per_hour))
head(hours_wages)

```

```

## # A tibble: 6 x 5
##   id job   year rate_per_hour hours_work
##   <int> <chr> <chr>         <int>         <int>
## 1     1 job_01 1979             328             38
## 2     1 job_02 1979              NA             15
## 3     1 job_03 1979              NA             NA
## 4     1 job_04 1979              NA             NA
## 5     1 job_05 1979              NA             NA
## 6     2 job_01 1979             385             35

```

Since our ultimate goal is to calculate the mean hourly wage, the number of jobs is calculate based on the availability of the `rate_per_hour` information. For example, the number of jobs of ID 1, based on `hours_work`, is two. However, since the information of hourly rate of `job_02` is not available, the number of job is considered as 1.

Further, we calculated the mean hourly wage for each ID in each year using a weighted mean with the hours of work as the weight. However, there are a lot of missing value in `hours_work` variable. In that case, we only calculated the mean hourly wage based on arithmetic/regular mean method. Hence, we created a new variable to flag whether the mean hourly wage is a weighted or a regular mean. Additionally, if an ID only had one job, we directly used their hourly wages information and flagged it as an arithmetic mean.

```

# calculate number of jobs that a person has in one year
no_job <- hours_wages %>%

```

```

filter(!is.na(rate_per_hour)) %>%
group_by(id, year) %>%
summarise(no_jobs = length(rate_per_hour))

# filter the observations with available rate per hour
eligible_wages <- hours_wages %>%
  filter(!is.na(rate_per_hour)) %>%
  left_join(no_job, by = c("id", "year"))

# calculate the mean_hourly_wage
# flag1 = code 1 for weighted mean
# code 0 for arithmetic mean
mean_hourly_wage <-
  eligible_wages %>%
  group_by(id, year) %>%
  #calculate the weighted mean if the number of jobs > 1
  mutate(wages = ifelse(no_jobs == 1, rate_per_hour/100,
                        weighted.mean(rate_per_hour, hours_work, na.rm = TRUE)/100)) %>%
  #give the flag if it the weighted mean
  mutate(flag1 = ifelse(!is.na(wages) & no_jobs != 1, 1,
                        0)) %>%
  #calculate the arithmetic mean for the na
  mutate(wages = ifelse(is.na(wages), mean(rate_per_hour)/100,
                        wages)) %>%
  group_by(id, year) %>%
  summarise(wages = mean(wages),
            total_hours = sum(hours_work),
            number_of_jobs = mean(no_jobs),
            flag1 = mean(flag1)) %>%
  mutate(year = as.numeric(year)) %>%
  ungroup() %>%
  rename(mean_hourly_wage = wages) %>%
  mutate(is_wm = ifelse(flag1 == 1, TRUE,
                        FALSE)) %>%
  dplyr::select(-flag1)

head(mean_hourly_wage, n = 10)

```

```

## # A tibble: 10 x 6
##       id year mean_hourly_wage total_hours number_of_jobs is_wm
##   <int> <dbl>         <dbl>         <int>         <dbl> <lgl>
## 1     1   1979           3.28             38           1 FALSE
## 2     1   1981           3.61             NA           1 FALSE
## 3     2   1979           3.85             35           1 FALSE
## 4     2   1980           4.57             NA           1 FALSE
## 5     2   1981           5.14             NA           1 FALSE
## 6     2   1982           5.71             35           1 FALSE
## 7     2   1983           5.71             NA           1 FALSE
## 8     2   1984           5.14             NA           1 FALSE
## 9     2   1985           7.71             NA           1 FALSE
## 10    2   1986           7.69             NA           1 FALSE

```

The mean_hourly_wage and full_demographic data are then joined. We also filtered the data to only have the cohort who completed the education up to 12th grade and participated at least five rounds in the survey.

```

# join the wages information and the demographic information by case id.
wages_demog <- left_join(mean_hourly_wage, full_demographics, by="id")
# calculate the years in work force and the age of the subjects in 1979
wages_demog <- wages_demog %>%

```

```

mutate(yr_hgc = as.numeric(yr_hgc)) %>%
mutate(years_in_workforce = year - yr_hgc) %>%
mutate(age_1979 = 1979 - (dob_year + 1900))
# filter only the id with high school education
wages_demog_hs <- wages_demog %>% filter(grepl("GRADE", hgc))
# calculate the number of observation
keep_me <- wages_demog_hs %>%
  count(id) %>%
  filter(n > 4)
wages_demog_hs <- wages_demog_hs %>%
  filter(id %in% keep_me$id)

```

3.2 Initial Data Analysis

3.3 Robust Linear Model for Influential Values Treatment

Conclusion

Wickham, Hadley. 2019. *Stringr: Simple, Consistent Wrappers for Common String Operations*. <https://CRAN.R-project.org/package=stringr>.

———. 2020. *Tidyr: Tidy Messy Data*. <https://CRAN.R-project.org/package=tidyr>.

Wickham, Hadley, Romain François, Lionel Henry, and Kirill Müller. 2020. *Dplyr: A Grammar of Data Manipulation*. <https://CRAN.R-project.org/package=dplyr>.