Quasi-long analysis with HES

Welcome to readabs microdata.

This package was made for you because the ABS's surveys can be difficult to analyse across years. Sometimes variable names change, sometimes their values change and it can be difficult to keep track of what's what. This package will help you combine and compare many years of the HES and SIH so that analysis is a little bit quicker.

This vignette will show you how to install the package in rstudio and run a quasi-longitudinal expenditure analysis.

Installation

You can install `readabsmicrodata` from Github as follows:

```
# If the `remotes` package is not installed, install it
if(!require(remotes)) {
  install.packages("remotes")
}

# Install `readabsmicrodata` from GitHub using remotes like this:
install.packages("kableExtra")
remotes::install_github("grattan/grattandata",
                        dependencies = TRUE,
                        upgrade = "always",
                        build_vignettes = TRUE)
remotes::install_github("jonathananolan/readabsmicrodata",
                        dependencies = TRUE,
                        upgrade = "always",
                        build_vignettes = TRUE)
```

Occasionally the package won't install because it's dependencies are not installed. R should install dependencies, but sometimes it fails. Try first restarting R, and if that doesn't work try installing the required package first. In the example above we installed "kableExtra" and "grattandata" before installing the package we're interested in.

Get started

To run the package you will need to first download the SAS version of the HES or SIH surveys that you are interested in. Unzip them all into a folder, with sub-folders for each year of the survey e.g. data/2015/sih15bh.sas7bdat. Keep the format files associated with each year in the same folder as the sas7bdat folder.

Now you can import many years of the dataset.

```
library(readabsmicrodata)

data<- read_abs_microdata(survey = "hes",
                          file   = "household",
                          data_dir="C:/data")
```

If you work at grattan and have access to the grattan data warehouse, you can instead run:

```
library(readabsmicrodata)

data<- read_abs_microdata(survey  = "hes",
                          file    = "household",
                          grattan = TRUE)
```

if you are only interested in a couple of years you can specify that too:

```
library(readabsmicrodata)

data<- read_abs_microdata(survey = "hes",
                          file   = "household",
                          data_dir="C:/data",
                          years = c(2009,2015))
```

This function will bind together every year of your dataset into one big dataframe that contains every year of the HES or the SIH. Easy to remember names have been created for the most commonly used variables.

For less common variables, the original name has been kept, with the addition of a suffix for the year, e.g. famcomp_2017. Where variables are used in multiple years, and the variable label is very similar in both years, the most recent year is assigned to variables for both years. This reduces the risk of merging two variables with the same name that different meanings across different years.

When you run read_abs_microdata a data dictionary will be created to help you compare each variable. There's no subsitute for reading the ABS microdata userguide and checking for changes, but this is a good start.

For our analysis we want to compare people with similar dates of birth across many years of the HES. To do this, we need to clean up this dataset a little bit. There's a few other functions that might help you analyse the HES and SIH too:

```
library(readabsmicrodata)
library(grattan) # used for "weighted_ntile" function
library(tidyverse)
library(Hmisc) # used for "wtd.median" function

data<- read_abs_microdata(survey = "hes",
                          file   = "household",
                          grattan = TRUE,
# I've said grattan = TRUE to point to the grattan data directory here but you can replace that with gr
                          create_html_dictionary = FALSE) %>%
  # Data dictionary is nice the first time, but there's no need to keep remaking it.

  inflate_survey(to_year = "2015-16") %>% # Inflates all the values to $2015-16
  equivalise_survey() %>% # Uses the OECD-modified equivalisation factor to adjust for household size
  fix_old_vars() %>% #adjusts old variables denomiated in cents to dollars.
  add_ages() %>%
  #Adds ages in 5, 10 and 20 year groupings.
  add_cohorts()
  #Adds 5, 10 and 20 year cohorts for dates of birth.
  #Where an age is a top group (e.g. 75 plus, 85 plus) it is excluded from cohort analysis
```

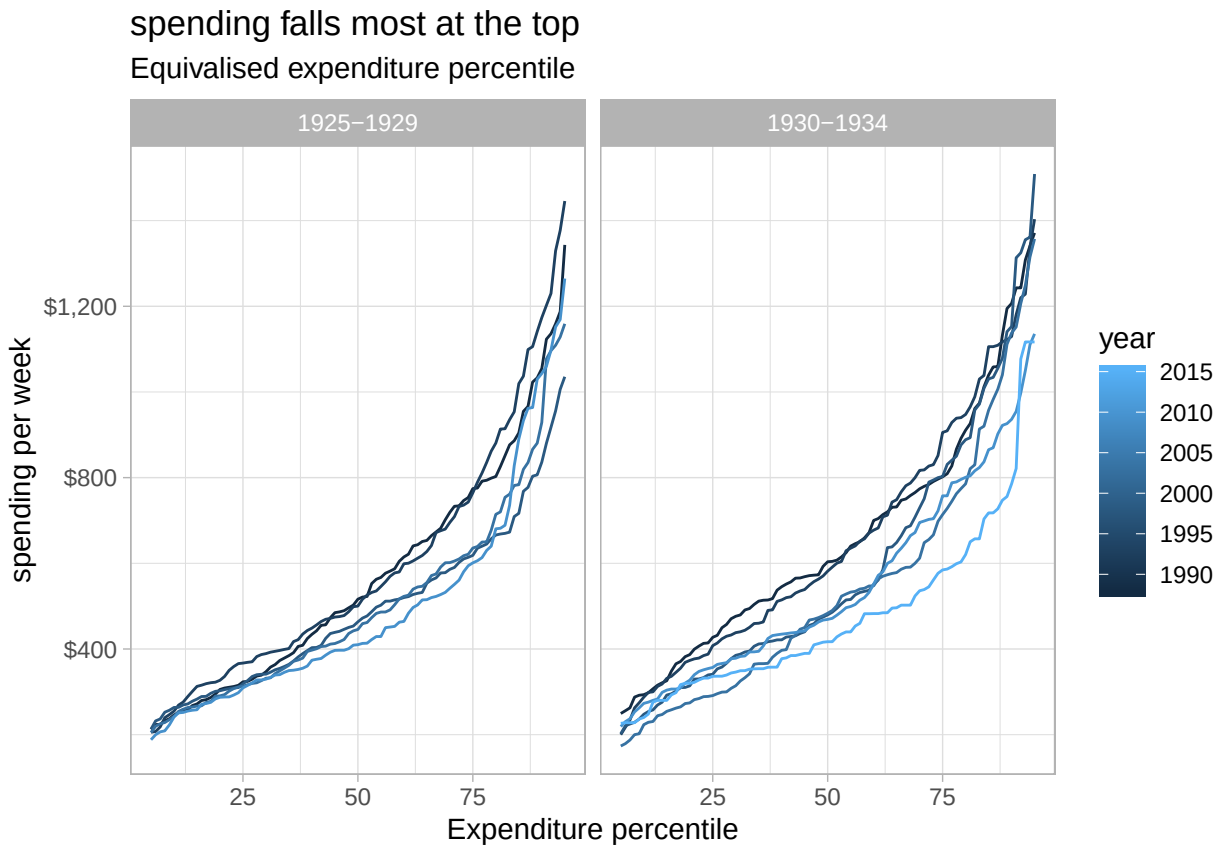
Now we have a nice clean dataset - analysis is relatively simple:

```
data %>%
  filter(!is.na(cohort_5y),
```

```
        cohort_5y!="",
        cohort_5y == c("1930-1934","1925-1929")) %>%
  group_by(cohort_5y,
           year) %>%
  summarise(mpg = list(enframe(wtd.quantile(exp_total_g_s,
                                            probs   = seq(.05,.95,.01),
                                            weights = weight)))) %>%
  #Calculates the value at each percentile, and returns the value as a nested list
  unnest() %>%
  # Unest into a tidy dataframe
  mutate(percentile = as.numeric(gsub("([0-9]+).*$", "\\1",name))) %>%
  #make percentiles numeric to graph them more easily
  ggplot(aes(x = percentile,
             y = value,
             colour = year,
             group = as.character(year))) +
  geom_line(stat = "identity")+
  facet_grid(.~cohort_5y)+
  labs(x = "Expenditure percentile",
       y = "spending per week",
       title = "spending falls most at the top",
       subtitle = "Equivalised expenditure percentile")+
  scale_y_continuous(labels = scales::dollar_format(accuracy = 1))+
  theme_light()
```

```
## Warning in cohort_5y == c("1930-1934", "1925-1929"): longer object length is not
## a multiple of shorter object length
```

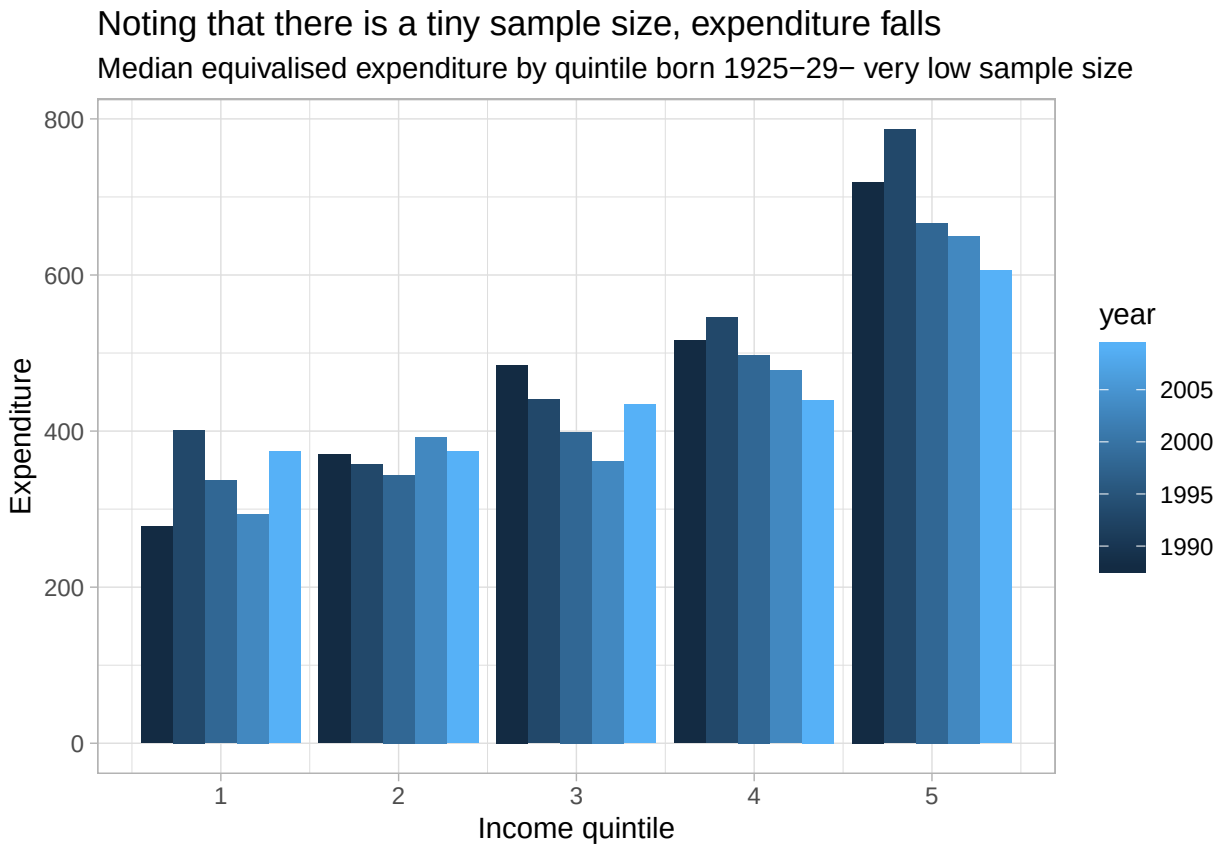
```
## Warning: `cols` is now required.
## Please use `cols = c(mpg)`
```



Grouping by expenditure percentile is simple and easy. We can do by quintile as well but our sample size starts to look vanishingly small.

```
data %>%
  filter(!is.na(cohort_5y),
         cohort_5y!="",
         cohort_5y == "1925-1929",
         equiv_hh %in% c(1,1.5)) %>%
#This will only include singles and ocuples. There are realtively few people >65 with kids and it's a
  group_by(cohort_5y,year) %>%
  mutate(income = if_else(is.na(income__disposable_0304),
  # income definitions changes in the mid 2000s. 0304 values are very close to 0506 values, and provi
                                income_disposable_0506,
                                income__disposable_0304),
       income_quintile = weighted_ntile(income,weight,5)) %>%
  group_by(income_quintile,
           cohort_5y,
           year) %>%
  summarise(exp = wtd.quantile(exp_total_g_s,weight,.5),
            n= n()) %>%
  ggplot(aes(x = income_quintile, y = exp, fill = year, group = year)) +
  geom_bar(stat = "identity",
           position = "dodge")+
  labs(title = "Noting that there is a tiny sample size, expenditure falls",
       subtitle = "Median equivalised expenditure by quintile born 1925-29- very low sample size",
       x = "Income quintile",
       y = "Expenditure")+
```

```
  theme_light()
```


Noting that there is a tiny sample size, expenditure falls
Median equivalised expenditure by quintile born 1925−29− very low sample size
Expenditure
800
600
400
200
0
1
2
3
4
5
Income quintile
year
2005
2000
1995
1990