Nominal house prices data in Luxembourg - Data cleaning

Eric Odongo

2025-04-24

```r
library(dplyr)
library(ggplot2)
library(janitor)
library(purrr)
library(readxl)
library(rvest)
library(stringr)
```

Downloading the data

This data is downloaded from the Luxembourguish Open Data Portal (the data set called *Série rétrospective desprix annoncés des maisons par commune, de 2010 à 2021*), and the original data is from the "Observatoire de l'habitat". This data contains prices for houses sold since 2010 for each luxembourguish commune.

The function below uses the permanent URL from the Open Data Portal to access the data, but I have also rehosted the data, and use my link to download the data (for archival purposes):

```r
get_raw_data <- function(url = "https://www.is.gd/1vvBAc"){
raw_data <- tempfile(fileext = ".xlsx")
download.file(url,
raw_data,
mode = "wb")
sheets <- excel_sheets(raw_data)
read_clean <- function(..., sheet){
read_excel(..., sheet = sheet) %>%
mutate(year = sheet)
}

raw_data <- map_dfr(
sheets, ~read_clean(raw_data,
skip = 10,
sheet = .)) %>%
clean_names()
raw_data %>%
rename(
locality = commune,
n_offers = nombre_doffres,
average_price_nominal_euros    = prix_moyen_annonce_en_courant,
average_price_m2_nominal_euros = prix_moyen_annonce_au_m2_en_courant,
average_price_m2_nominal_euros = prix_moyen_annonce_au_m2_en_courant
) %>%
mutate(locality = str_trim(locality)) %>%
select(year, locality, n_offers, starts_with("average"))
}
```

```
raw_data <- get_raw_data(url = "https://www.is.gd/1vvBAc")
```

```
## New names:
## * ‘*‘ -> ‘*...3‘
## * ‘*‘ -> ‘*...4‘
```

We need clean the data: “Luxembourg” is “Luxembourg-ville” in 2010 and 2011, then “Luxembourg”. “Pétange” is also spelt non-consistently, and we also need to convert columns to the right type. We also directly remove rows where the locality contains information on the “Source”:

```
clean_raw_data <- function(raw_data){

raw_data %>%
mutate(locality = ifelse(grepl("Luxembourg-Ville", locality), "Luxembourg",
                         locality),
       locality = ifelse(grepl("P.tange", locality), "Pétange", locality)
) %>%
filter(!grepl("Source", locality)) %>%
mutate(across(starts_with("average"), as.numeric))
}
```

```
flat_data <- clean_raw_data(raw_data)
```

```
## Warning: There were 2 warnings in ‘mutate()‘.
## The first warning was:
## i In argument: ‘across(starts_with("average"), as.numeric)‘.
## Caused by warning:
## ! NAs introduced by coercion
## i Run ‘dplyr::last_dplyr_warnings()‘ to see the 1 remaining warning.
```

We now need to make sure that we got all the communes/localities in there. There were mergers in 2011, 2015 and 2018. So we need to account for these localities. We’re now scraping data from Wikipedia of former Luxembourguish communes:

```
get_former_communes <- function(
url = "https://www.is.gd/lux_former_communes",
min_year = 2009,
table_position = 3
){
read_html(url) %>%
html_table() %>%
    pluck(table_position) %>%
clean_names() %>%
filter(year_dissolved > min_year)
}
```

```
former_communes <- get_former_communes()
```

We can scrape current communes:

```
get_current_communes <- function(
url = "https://www.is.gd/lux_communes",
table_position = 2
){
read_html(url) |>
html_table() |>
pluck(table_position) |>
clean_names() |>
filter(name_2 != "Name") |>
rename(commune = name_2) |>
mutate(commune = str_remove(commune, " .$"))
}
```

```
current_communes <- get_current_communes()
```

Let's now create a list of all communes:

```
get_test_communes <- function(former_communes, current_communes){
communes <- unique(c(former_communes$name, current_communes$commune))

# we need to rename some communes
# Different spelling of these communes between wikipedia and the data
communes[which(communes == "Clemency")] <- "Clémency"
communes[which(communes == "Redange")] <- "Redange-sur-Attert"
communes[which(communes == "Erpeldange-sur-Sûre")] <- "Erpeldange"
communes[which(communes == "Luxembourg City")] <- "Luxembourg"
communes[which(communes == "Käerjeng")] <- "Kaerjeng"
communes[which(communes == "Petange")] <- "Pétange"
communes
}
```

```
former_communes <- get_former_communes()
current_communes <- get_current_communes()
communes <- get_test_communes(former_communes, current_communes)
```

Let's test to see if all the communes from our dataset are represented.

```
setdiff(flat_data$locality, communes)
```

```
## [1] NA                   "Moyenne nationale" "Total d'offres"
```

If the above code doesn't show any communes, then this means that we are accounting for every commune.

Let's keep the national average in another dataset:

```r
make_country_level_data <- function(flat_data){
country_level <- flat_data %>%
filter(grepl("nationale", locality)) %>%
select(-n_offers)
offers_country <- flat_data %>%
filter(grepl("Total d.offres", locality)) %>%
select(year, n_offers)
full_join(country_level, offers_country) %>%
select(year, locality, n_offers, everything()) %>%
mutate(locality = "Grand-Duchy of Luxembourg")
}
```

```r
country_level_data <- make_country_level_data(flat_data)
```

```
## Joining with ‘by = join_by(year)‘
```

We can finish cleaning the commune data:

```r
make_commune_level_data <- function(flat_data){
flat_data %>%
filter(!grepl("nationale|offres", locality),
!is.na(locality))
}
```

```r
commune_level_data <- make_commune_level_data(flat_data)
```

We now save the dataset in a folder for further analysis (keep chunk option to `eval = FALSE` to avoid running it when knitting):

```r
write.csv(commune_level_data,
          "../datasets/house_prices_commune_level_data.csv",
          row.names = FALSE)

write.csv(country_level_data,
          "../datasets/house_prices_country_level_data.csv",
          row.names = FALSE)
```