A tool box for a climate neutral housing sector: Data Cleaning

Anna Hornykewycz* Jakob Kapeller† Jan David Weber‡ Bernhard Schütz§
Lukas Cserjan¶

2024-10-26

1 Preparations

1.1 Load packages and set seed

This chunk gathers all relevant packages and sets one seed for all the random processes that follow.

```
library(here)
library(tidyverse)
library(data.table)
library(ggplot2)
library(dplyr)
library(viridis)

set.seed(1301)
```

1.2 Import Climate Goals

Here, we import the climate goals.

```
climate_goals <- read.csv(here::here(
                 "Intermediate_Results/global_buildings_climate_goals2040.csv"))
```

1.3 Setting basic parameters

```
# basic policy parameters
rate_renovation <- 3
ec_goal <- 60
```

*Institute for Comprehensive Analysis of the Economy (ICAE), Johannes Kepler University, Linz, Austria
†Institute for Socio-Economics, University of Duisburg-Essen, Germany and Institute for Comprehensive Analysis of the Economy (ICAE), Johannes Kepler University, Linz, Austria
‡Institute for Socio-Economics, University of Duisburg-Essen, Germany
§Vienna Institute for International Economic Comparisions (wiiw) and Institute for Socio-Economics, University of Duisburg-Essen, Germany
¶Institute for Comprehensive Analysis of the Economy (ICAE), Johannes Kepler University, Linz, Austria

```r
# basic technical parameters
apartment_per_building <- 6.85
```

1.4 Importing a dataset

In this section micro-data on available residential units is imported. In addition, we identify all variables of interest, create a new variable for the type of building (flat vs. house), trim the dataset in several basic dimensions and translate key variables into english.

When applying the code make sure that the files containing the raw data are in a folder "data" within the repository of this code file.

For *Germany*, there is ample data available, so we restrict the units observed to those observed in housing ads on immoscout24.de published in 2020 or later.[1]

```r
v1 <- 7
v2 <- 14

path_str <- c(paste0("HKSUF",v1:v2), paste0("HMSUF",v1:v2),
              paste0("WKSUF",v1:v2), paste0("WMSUF",v1:v2))

vars_of_interest <- c("ev_kennwert", "bef1", "spell", "letzte_modernisierung",
                      "baujahr", "wohnflaeche", "immobilientyp", "zimmeranzahl",
                      "kategorie_Haus", "kategorie_Wohnung",
                      "energieeffizienzklasse", "energieausweistyp", "anbieter",
                      "ejahr")

for (p in path_str){
  path <- here::here(paste0("Data/", p, ".csv"))
  raw <- data.table::fread(path,
                           select=vars_of_interest,
                           colClasses=c(baujahr="character",
                                        wohnflaeche="character"))
  mod <- raw %>%
    filter(spell == 1) %>% # this eliminates double entries
    filter(ejahr>2019) %>%
    filter(kategorie_Haus!="Block of flats") %>% # eliminates unusable building type
    mutate(type = ifelse(grepl("H", p) == TRUE, "House", "Apartment"))
    # assign(paste0(p), mod)
    # the line above imports individual files as separate data frames
  if(p == path_str[1]){
    immo_raw <- mod
  } else {
    immo_raw <- dplyr::bind_rows(immo_raw, mod)
  }
  print(p)
}

immo_raw <- immo_raw %>%
    mutate(living_area = as.double(wohnflaeche),
           ec_index = as.double(ev_kennwert),
```

[1] In terms of concrete data vintages this implies using data files 7 to 14 for all relevant data groups (housing vs. flats, lease vs. sell).

```
            construction_year = as.double(baujahr),
            last_modernization = as.double(letzte_modernisierung)) %>%
    rename(
    efficiency_class_raw = energieeffizienzklasse,
    energy_certificate_type = energieausweistyp,
    provider = anbieter #,
    #living_area = wohnflaeche
  )
```

2 Data processing

2.1 Data cleaning

In a first step we eliminate entries where crucial data is missing and exclude implausible values for `living_area`.

```
na_strings <- c("Other missing", "Implausible value", "NO_INFORMATION",
                "Not specified", NA, "NA")

immo_raw <- immo_raw %>%
  filter(!ec_index %in% na_strings) %>%
  filter(!bef1 %in% na_strings) %>%
  filter(!living_area %in% na_strings)

immo_raw <- immo_raw %>%
  filter(living_area > 20) %>%
  filter(living_area < 500)
```

2.2 Creation of new variables

We create a new variable for the `efficiency_class` based on `ec_index` (which focuses more on final energy use than the information of efficiency classes as contained in our dataset for *Germany*), as well as a variable for final energy use (`energy_use`). In doing we also transform some variables to numeric information.

Furthermore, we create a variable called `renovation weight`. This variable allows for taking the differences between houses and apartments into account and requires the average number of apartments per multi-unit building as an input. Finally, we collect different descriptions for available heating systems to arrive at a more consistently worded variable.

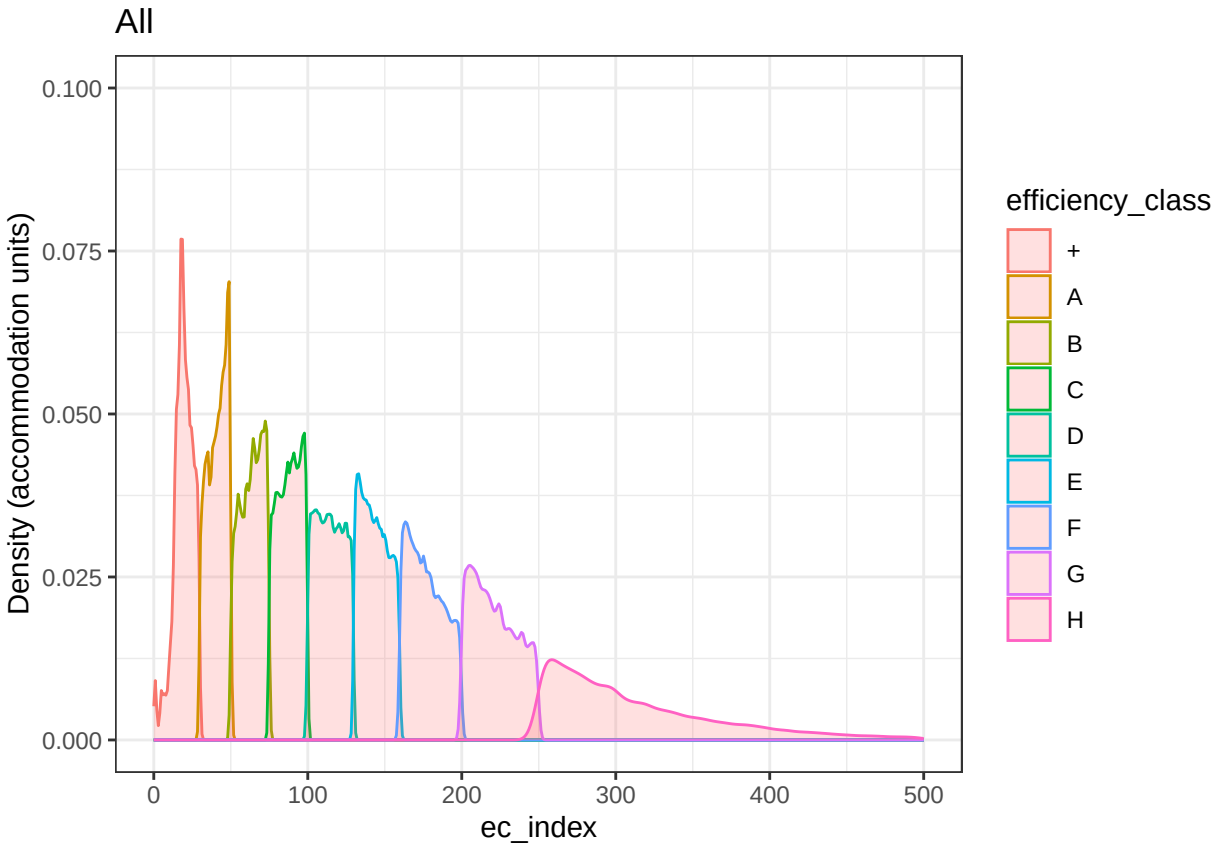
```
immo_raw <- immo_raw %>%
  mutate(
    #living_area = as.double(living_area),
    efficiency_class = case_when(
      ec_index < 30 ~ "+",
      ec_index < 50 ~ "A",
      ec_index < 75 ~ "B",
      ec_index < 100 ~ "C",
      ec_index < 130 ~ "D",
      ec_index < 160 ~ "E",
      ec_index < 200 ~ "F",
      ec_index < 250 ~ "G",
```

```
      ec_index >= 250 ~ "H"
    ),
    energy_use = living_area * ec_index
  )

immo_raw <- immo_raw %>%
  mutate(
    renovation_weight = if_else(type == "Apartment",
                                1 / apartment_per_building, 1))

other <- c("COMBINED_HEAT_AND_POWER_FOSSIL_FUELS" )
unclear <- c("HEAT_SUPPLY", "LOCAL_HEATING")
biomass <- c("WOOD","PELLET_HEATING","WOOD_CHIPS","BIO_ENERGY",
             "COMBINED_HEAT_AND_POWER_BIO_ENERGY",
             "COMBINED_HEAT_AND_POWER_REGENERATIVE_ENERGY")
heatingpump <- c("GEOTHERMAL", "ENVIRONMENTAL_THERMAL_ENERGY",
                 "SOLAR_HEATING", "HYDRO_ENERGY", "WIND_ENERGY",
                 "COMBINED_HEAT_AND_POWER_RENEWABLE_ENERGY")
gas <- c("GAS", "LIQUID GAS", "NATURAL_GAS_LIGHT", "NATURAL_GAS_HEAVY")
district_heating <- c("DISTRICT_HEATING", "STEAM_DISTRICT_HEATING")
coal <- c("COAL", "COAL_COKE")
oil <- c("OIL")
electricity <-c("ELECTRICITY")

immo_raw <- immo_raw %>%
  mutate(
    heating_system = case_when(befl %in% electricity ~ "electricity",
                               befl %in% oil ~ "oil",
                               befl %in% coal ~ "coal",
                               befl %in% district_heating ~ "DH",
                               befl %in% gas ~ "gas",
                               befl %in% heatingpump ~ "HP & other renewables",
                               befl %in% biomass ~ "biomass",
                               befl %in% unclear ~ "unclear",
                               befl %in% other ~ "other"))
```

2.3 Preliminary visualization of the dataset focusing on efficiency classes / energy use

Create a plot that shows the distribution of the different efficiency classes based on `ec_index`.

```
plot_all <- ggplot(immo_raw, aes(x = ec_index)) +
  geom_density(aes(colour = efficiency_class), alpha = .2, fill = "#FF6666",
               show.legend = TRUE) +
  labs(title = "All", x = "ec_index", y ="Density (accommodation units)") +
  xlim(0, 500) +
  ylim(0, 0.1) +
  theme_bw()

plot_all
```



Export the plot as a pdf file.

```
ggsave("../Graphs/plot_all.pdf", plot_all, width = 6, height = 4)
```

3 Creating a representative sample

We acknowledge that our sample is not representative across all efficiency classes. Especially low efficiency classes are underrepresented in the available data. As the real distribution of the efficiency classes, as well as the house vs flat distribution is known, we are able to create a dataset that is representative for these two key dimensions.[2]

3.1 Resampling the original data

To calibrate the sampling process information in the aggregate distribution of types (houses vs. apartments) and efficiency classes is required. Also the number of observations to be included in the final data set can be defined in this section.

```
# set size of final dataset
n_dataset <- 250000

# define the known distribution
type_share <- c("Apartment" = 0.537, "House" = 0.463)
type_distribution <- type_share * n_dataset
```

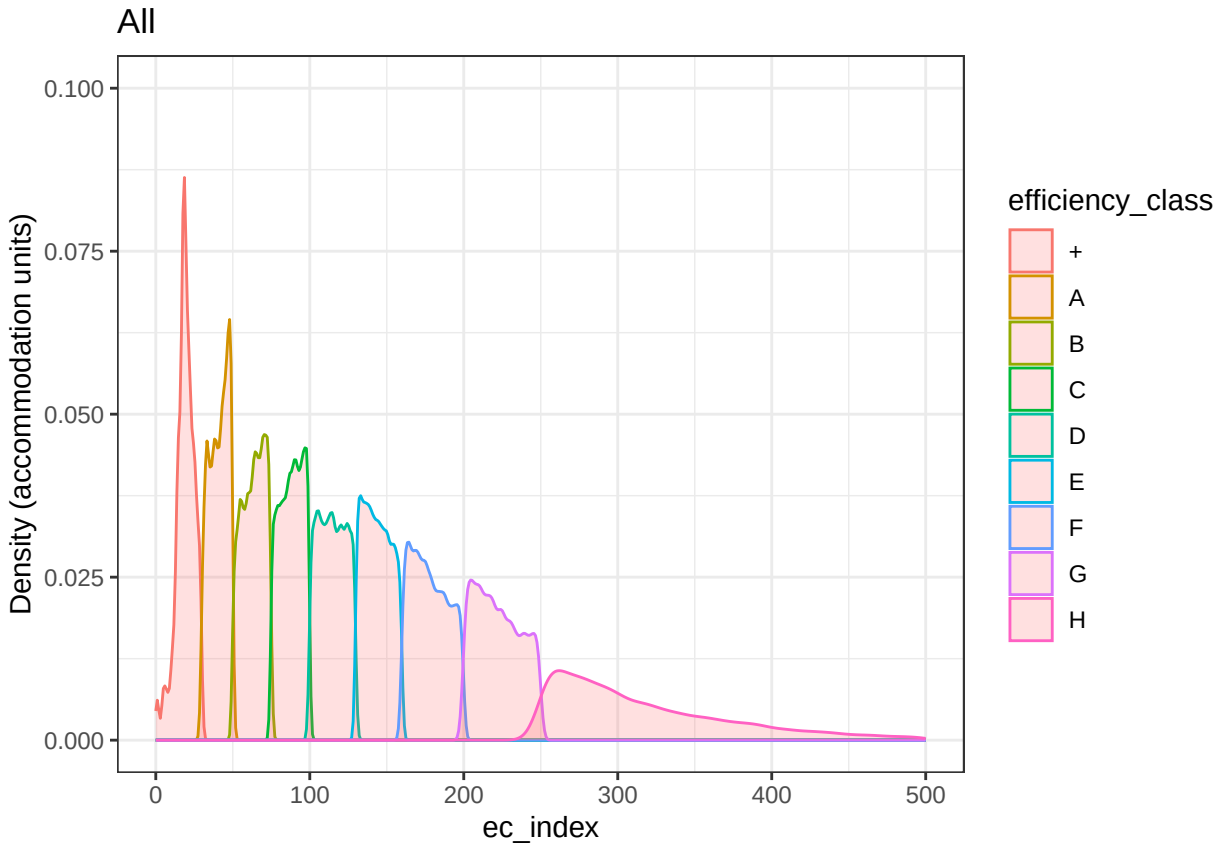
[2]By doing so, we implicitly assume that the observations within an efficiency class are representative.

```r
efficiency_share <- data.frame(
          "efficiency_class" = c("+", "A", "B", "C", "D", "E", "F", "G", "H"),
          "realdist"=c(4, 3, 7, 12, 15, 14, 15, 13, 17)/100)

# create a representative sample
set.seed(1301)
immo_rep1 <- immo_raw %>%
  split(.$type) %>%
  imap_dfr(~slice_sample(.x, n = round(type_distribution[.y]), replace=TRUE)) %>%
  mutate(index=seq(1:n_dataset))

immo_rep1 <- immo_rep1[sample(nrow(immo_rep1)),]

immo_classes_rep1 <- immo_rep1 %>%
  group_by(efficiency_class) %>%
  summarise(buildings = sum(renovation_weight),
            observations = n(),
            raw_class_share = n()/nrow(immo_rep1),
            weighted_class_share = sum(renovation_weight)/
                                      sum(immo_rep1$renovation_weight),
            mean_weight = mean(renovation_weight)) %>%
            mutate(realdist = efficiency_share$realdist,
            absdistance = weighted_class_share - realdist,
            reldistance = (weighted_class_share - realdist) /
                                realdist) %>%
  ungroup()

least_represented <- immo_classes_rep1 %>%
  filter(reldistance==min(immo_classes_rep1$reldistance))

immo_classes_rep1 <- immo_classes_rep1 %>%
                     mutate(todraw = floor(pull(least_represented, buildings)/
                            pull(least_represented, realdist)*realdist),
                            todraw_weighted = floor(todraw / mean_weight))

immo_rep1 <- immo_rep1 %>%
  group_by(efficiency_class) %>%
  mutate(cum_class_weight = cumsum(renovation_weight)) %>%
  ungroup()

for (i in efficiency_share$efficiency_class){
sampled_data <- filter(immo_rep1, efficiency_class==i & cum_class_weight < pull(filter(immo_classes_rep

 if(i=="+"){
    immo_rep <- sampled_data
  } else {
    immo_rep <- merge(immo_rep, sampled_data, all=TRUE)
  }
}

for (i in efficiency_share$efficiency_class){
  print(i)
  print(pull(filter(immo_classes_rep1,efficiency_class==i),todraw))
```

```
  print(max(pull(filter(immo_rep,efficiency_class==i),cum_class_weight)))
  print(sum(select(filter(immo_rep,efficiency_class==i),renovation_weight)))
}
```

3.2 Plotting the representative dataset

```
plot_rep <- ggplot(immo_rep, aes(x=ec_index)) +
  geom_density(aes(colour= efficiency_class), alpha=.2, fill="#FF6666",
               show.legend = TRUE) +
  labs(title="All", x="ec_index", y="Density (accommodation units)") +
  xlim(0, 500) +
  ylim(0, 0.1) +
  theme_bw()

plot_rep
```



Export the plot as a pdf file.

```
ggsave("../Graphs/plot_rep.pdf", plot_rep, width = 6, height = 4)
```

3.3 Plausibility checks

In this section we make us of some offical data to assess the properties of the representative sample. In detail we perform the following checks:

1. We first reproduce the share associated with building types and efficiency classes used to calibrate the dataset.
2. On average, the thermal envelope of all private and commercial buildings in Germany corresponds to energy efficiency class F, with an annual demand or consumption of 170 kWh/m. With our data we can roughly reproduce this value.
3. We compare the share of squaremeters of houses and apartments in the worst efficiency classes; these should be 40% (for houses) and 16% (for apartments).

For the *German case*, the first two test give convincing results, close to aggregate estimates. For the third test, we find that we cannot fully account for the fact that more badly insulated buildings are, on average, larger. Hence, our estimates.

```
# Confirm correct shares (1)
print(count(filter(immo_rep,type == "Apartment")) / nrow(immo_rep))
```

```
##           n
## 1 0.5235308
```

```
print(count(filter(immo_rep,type == "House")) / nrow(immo_rep))
```

```
##           n
## 1 0.4764692
```

```
immo_rep %>%
  group_by(efficiency_class) %>%
  summarise(share_effclasses = sum(renovation_weight)/
              sum(immo_rep$renovation_weight))
```

```
## # A tibble: 9 x 2
##   efficiency_class share_effclasses
##   <chr>                       <dbl>
## 1 +                          0.0400
## 2 A                          0.0300
## 3 B                          0.0700
## 4 C                          0.120
## 5 D                          0.150
## 6 E                          0.140
## 7 F                          0.150
## 8 G                          0.130
## 9 H                          0.170
```

```
# confirm average insulation quality (2)
weighted.mean(immo_rep$ec_index, immo_rep$renovation_weight)
```

```
## [1] 165.37
```

```
# share of squaremeters in the worst efficiency-classes (3)
immo_rep %>%
  filter(type == "House") %>%
  group_by(efficiency_class) %>%
  summarise(share_effclasses = sum(living_area) /
                    sum(immo_rep$living_area[immo_rep$type == "House"]))
```

```
## # A tibble: 9 x 2
##   efficiency_class share_effclasses
##   <chr>                      <dbl>
## 1 +                         0.0419
## 2 A                         0.0302
## 3 B                         0.0648
## 4 C                         0.111
## 5 D                         0.142
## 6 E                         0.139
## 7 F                         0.159
## 8 G                         0.139
## 9 H                         0.173
```

```
immo_rep %>%
  filter(type == "Apartment") %>%
  group_by(efficiency_class) %>%
  summarise(share_effclasses = sum(living_area) /
                    sum(immo_rep$living_area[immo_rep$type == "Apartment"]))
```

```
## # A tibble: 9 x 2
##   efficiency_class share_effclasses
##   <chr>                      <dbl>
## 1 +                         0.0262
## 2 A                         0.0359
## 3 B                         0.127
## 4 C                         0.221
## 5 D                         0.247
## 6 E                         0.172
## 7 F                         0.107
## 8 G                         0.0456
## 9 H                         0.0188
```

3.4 Export the representative dataset

We export the dataset to save the cleaned data. This allows to remove temporary calculations in R, save calculation time, and import the cleaned data.

```
write.csv(immo_rep, here::here("Intermediate_Results/data_clean_representative.csv"),
          row.names = FALSE)
```