A tool box for a climate neutral housing sector: Analysis

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 Previous Results

We import the dataset to save the cleaned data and teh cliamte goals:

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

1.3 Input for Renovation Rate and EC Goal

In this section we define two key (policy) parameters, namely the renovation rate(given as the share of renovated units in %-points) and the goal for the energy requirements (in kWh/m_2) of the renovated buildings.

Furthermore, we need information on the average number of apartments per building to account for the fact that datasets typically incorporate apartments whereas the relevant unit of observation is buildings.

*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 policy parameters
rate_renovation <- 3
ec_goal <- 60

# basic technical parameters
apartment_per_building <- 6.85
```

2 Main analysis: Renovating the residential building sector

2.1 Define additional technical/scientific variables

In this section, we define a series of variables that are required to map the information contained in our dataset on ghg-emissions and aggregate costs. These variables are taken from the related literature or simply assumed as parameters. If the former applies for the *German case* a source is given as a weblink.

- `share_hp` defines the share of observations which are suitable to equip with a heating pump
- `share_dh` defines the share of observations which are suitable to equip with district heating
- `cop_factor` defines the annual coefficient of performance. It measures the ratio of energy supplied (electricity) to energy generated (heat dissipated). Source
- `share_natural_gas` The share of heating which is based on natural gas Source
- `share_oil` The share of building, which employs heating oil Source
- `factor_natural_gas` The greenhouse gas emissions in gigatons per kwh of energy demanded for heating with natural gas. Source
- `factor_oil` The greenhouse gas emissions in giga tons per kwh for heating with heating oil.
- `fossil_factor` The greenhouse gas emissions for fossil heating (which is calculated by employing the factors and shares for oil and gas as given above.
- `factor_dh_direct` The greenhouse gas emissions in gram per kwh for heating with district heating, translated to giga tons.
- `electricity_factor` The greenhouse gas emissions in gram per kwh for heating with power (heating pump), translated to giga tons. Source
- `dh_decarb_factor` is parameter defining to what extent district heating can be effectively decarbonized.
- `years_til_2030` defines how many years are left until 2030.

The emission factors considered here abstract from emissions relating to the pre-processing of relevant energy sources as these are, strictly speaking, not part of the housing sector. However, to make it easier to change this assumption we included estimates for emission factors for the *German case*, that take pre-processing into account (see this source for details.)

```r
share_hp <- 0.75
share_dh <- 1 - share_hp
cop_factor <- 3.1
share_natural_gas <- 0.482
share_oil <- 0.256

factor_natural_gas <- 2.02 * 10 ** -13
# factor_natural_gas <- 2.51 * 10 ** -13 incl. upstream chain
factor_oil <- 2.66 * 10 ** -13
# factor_oil <- 3.19 * 10 ** -13 incl. upstream chain
factor_dh_direct <- 2.00 * 10 ** -13
# factor_dh_direct <- 2.29 * 10 ** -13 incl. upstream chain
```

```r
fossil_factor <- (factor_natural_gas * share_natural_gas + factor_oil * share_oil) /
  (share_oil + share_natural_gas)
electricity_factor <- 4.42 * 10 ** -13
#electricity_factor <- 4.98 * 10 ** incl. upstream chain

dh_decarb_factor <- 1
years_til_2030 <- 7
```

2.2 Reorganize data on residential units

This section creates the variable `partial_renovation` for assessing whether and to what extent the insulation of some residential unit is required to reach the desired standard. We thereby assume that no renovation is needed in case the residential units already complies with or overperforms the desired standard, while a full renovation is needed for all buildings, which have an energy use surpassing the average. Between these values we employ a linear extrapolation to numerically describe the expected renovation requirements. In addition, this section simulates the replacement of heating facilities according to the parameters given in in the section above.

```r
# Calculate partial_renovation
immo_clean <- mutate(immo_rep,
                     partial_renovation = case_when(
                       ec_index >= weighted.mean(ec_index, renovation_weight) ~ 1,
                       ec_index <= ec_goal ~ 0,
                       TRUE ~ (ec_index - ec_goal) / (weighted.mean(ec_index,
                                          renovation_weight) - ec_goal)))

# Simplify heating_system into firing_simple
immo_clean <- mutate(immo_clean,
                     firing_simple = case_when(
                       heating_system == "DH" ~ "DH",
                       heating_system == "HP & other renewables" ~ "HP",
                       heating_system == "electricity"  ~ "electricity",
                       TRUE ~ "FOSSIL"),
                     energy_use_direct = case_when(
                       heating_system == "DH" ~ 0,
                       heating_system == "HP & other renewables" ~ 0,
                       heating_system == "electricity" ~ 0,
                       TRUE ~ energy_use)
                     )

# Calculate firing_final
immo_clean <- mutate(immo_clean,
                     firing_final =
                     ifelse(firing_simple == "DH", "DH(old)",
                       ifelse(firing_simple == "HP", "HP(old)",
                         ifelse(rbinom(n = nrow(immo_clean),1 , share_dh) == 1,
                                "DH(new)",
                                "HP(new)")))
                     )

# Assign cop
immo_clean <- immo_clean %>%
```

```
  mutate(cop = case_when(
           firing_final == "HP(old)" ~ cop_factor,
           firing_final == "HP(new)" ~ cop_factor,
           TRUE ~ NA_real_ ))
```

2.3 Calculate current ghg-emissions

```
immo_clean <- mutate(immo_clean,
                     ghg_original = case_when(
                       firing_simple == "DH" ~ energy_use * factor_dh_direct,
                       firing_simple == "HP" ~ energy_use / cop *
                         electricity_factor,
                       firing_simple == "electricity" ~ energy_use *
                         electricity_factor,
                       TRUE ~ energy_use * fossil_factor
                       ),
                     ghg_direct = case_when(
                       firing_simple == "DH" ~ 0,
                       firing_simple == "HP" ~ 0,
                       firing_simple == "electricity" ~ 0,
                       TRUE ~ energy_use_direct * fossil_factor
                       )
                     )
```

2.4 Calculate potential savings of energy required

```
immo_clean <- mutate(immo_clean,
                     potential_saving_kW =
                       ifelse(partial_renovation == 0, 0,
                              pmax(energy_use - (ec_goal * living_area), 0)),
                     potential_saving_kW_direct =
                       ifelse(partial_renovation == 0, 0,
                              pmax(energy_use_direct - (ec_goal * living_area),
                                   0)))
```

2.5 Savings in terms of ghg-emissions

In this section we translate the savings potential in final energy into reductions in ghg-emissions. The following general considerations inform our implementation:

1. If the existing heating system is district heating potential savings arise from:

(a) insulation

(b) changes in the provisioning of district heating (less fossil sources used)

2. If the existing heating system is a fossil heating system and it is replaced by district heating, savings arise from:

(a) insulation
(b) difference int he Emission factor of fossil heating and the energy mix behind district heating
(c) changes in the provisioning of district heating (less fossil sources used)

3. If the existing heating system is a heating pump, potential savings arise from:

(a) insulation
(b) decarbonization of electricity

4. If the existing heating system is a fossil heating system and it is replaced by heating pump, savings arise from:

(a) additional insulation
(b) replacing fossil energy sources by electricity
(c) decarbonization of electricity

On this basis we calculate the potential ghg emission savings for four different scenarios regarding the power composition. (1) `ghg_direct`: this estimate reproduces the logic of conventional measurements building on direct emissions only. (1) `ghg_old`: we assume that the composition of the origin of power remains unchanged. This leads to relatively high emissions for switching to heating pumps as the current energy mix in Germany remains dependent on fossil fuels. (2) `ghg_neutral`: we assume the German power production is emission neutral. (3) `ghg_decarb`: we assume district heating as well as electricity will be provided free of emissions.

```
immo_clean <- mutate(immo_clean,
  potential_saving_ghg_direct =
    ifelse(firing_final %in% c("DH(old)", "HP(old)"), 0,
    ifelse(firing_simple == "electricity", 0,
           energy_use_direct * fossil_factor)),

  potential_saving_ghg_old =
    ifelse(firing_final == "DH(old)", potential_saving_kW * factor_dh_direct,
    ifelse(firing_final == "DH(new)" & firing_simple != "electricity",
      energy_use * fossil_factor - (energy_use - potential_saving_kW) *
                                   factor_dh_direct,
    ifelse(firing_final == "DH(new)" & firing_simple == "electricity",
        energy_use * electricity_factor - (energy_use - potential_saving_kW)
                                          * factor_dh_direct,
    ifelse(firing_final == "HP(old)",
          potential_saving_kW * electricity_factor / cop,
    ifelse(firing_final == "HP(new)" & firing_simple != "electricity",
            energy_use * fossil_factor - (energy_use - potential_saving_kW) /
                                         cop * electricity_factor,
    ifelse(firing_final == "HP(new)" & firing_simple == "electricity",
              energy_use * electricity_factor - (energy_use - potential_saving_kW)
                                                / cop * electricity_factor,
              NA
            )
          )
        )
      )
    )
  ),
```

```
  potential_saving_ghg_neutral =
    ifelse(firing_final == "DH(old)", potential_saving_kW * factor_dh_direct,
    ifelse(firing_final == "DH(new)" & firing_simple != "electricity",
      energy_use * fossil_factor - (energy_use - potential_saving_kW)
      * factor_dh_direct,
    ifelse(firing_final == "DH(new)" & firing_simple == "electricity",
        energy_use * electricity_factor - (energy_use - potential_saving_kW)
        * factor_dh_direct,
    ifelse(firing_final == "HP(old)",
          energy_use / cop * electricity_factor,
    ifelse(firing_final == "HP(new)" & firing_simple != "electricity",
            energy_use * fossil_factor,
    ifelse(firing_final == "HP(new)" & firing_simple == "electricity",
              energy_use * electricity_factor, NA
            )
          )
        )
      )
    )
  ),

  potential_saving_ghg_decarbDH =
    ifelse(firing_final == "DH(old)",
    energy_use * factor_dh_direct - (energy_use - potential_saving_kW)
    * factor_dh_direct * (1 - dh_decarb_factor),
    ifelse(firing_final == "DH(new)" & firing_simple != "electricity",
      energy_use * fossil_factor - (energy_use - potential_saving_kW)
      * factor_dh_direct * (1 - dh_decarb_factor),
    ifelse(firing_final == "DH(new)" & firing_simple == "electricity",
        energy_use * electricity_factor - (energy_use - potential_saving_kW)
        * factor_dh_direct * (1 - dh_decarb_factor),
    ifelse(firing_final == "HP(old)",
          energy_use / cop * electricity_factor,
    ifelse(firing_final == "HP(new)" & firing_simple != "electricity",
            energy_use * fossil_factor,
    ifelse(firing_final == "HP(new)" & firing_simple == "electricity",
              energy_use * electricity_factor, NA
            )
          )
        )
      )
    )
  )
)
```

We calculate the relative savings of greenhouse gas emissions, based on the three different scenarios.

```
immo_clean <- mutate(immo_clean,
              ghg_savings_direct   = potential_saving_ghg_direct / ghg_direct,
              ghg_savings_old      = potential_saving_ghg_old / ghg_original,
              ghg_savings_neutral  = potential_saving_ghg_neutral / ghg_original,
              ghg_savings_decarbDH = potential_saving_ghg_decarbDH / ghg_original
              )
```

3 Order of renovations

3.1 Priorization of renovations

This section allows for introducing a prioritization of renovations by sorting observations in the suggested order. At this point different variables can be employed to provide an exact specification of *how exactly* this sorting should be operationalized, which not only impacts on mitigation speed, but also on expected costs.

In our setup we prioritize the order of renovations, based on expected emissions savings adjusted for size and renovation weight. We choose this setup, because it allows for a fast transmission and, at the same, gives estimates for costs and required capacities that seem economically feasible.

```
immo_prio <- immo_clean %>%
  arrange(desc(potential_saving_ghg_old * living_area / renovation_weight)) %>%
  mutate(kum_partial_renovation = cumsum(partial_renovation)) %>%
  mutate(kum_renovation_weight = cumsum(renovation_weight))
```

We create a variable which defines the number of renovations using the cumulative sums of all renovation weights and the renovation rate. In other words, we calculate the number of buildings that are renovated in any given year. Finally, we add for every observation an indicator indicating the decade, in which the building will be renovated.

```
binsize <- max(immo_prio$kum_renovation_weight) / (100/rate_renovation)
cutz <- binsize*(1:(100 / rate_renovation))
#number of intervals till everything is renovated

immo_prio <-  mutate(immo_prio,
               year_renovation_prio = findInterval(kum_renovation_weight, cutz,
                                                   rightmost.closed = TRUE) + 2024,
               timing_prio = ifelse(year_renovation_prio <= 2030, 2030,
                             ifelse(year_renovation_prio <= 2040, 2040,
                               ifelse(year_renovation_prio <= 2050, 2050,
                                 ifelse(year_renovation_prio <= 2060, 2060,
                                                                 NA))))
                  )
```

We can now generate a timing table for the renovations. The code calculates how many weighted full renovations have to be done per year based on the political goal of decarbonation.

Then the code assigns scheduled dates of renovation to every data point based on the ghg saving potential and the necessary renovations per year.

```
timingtable_prio <- data.table(
  "what" = c("all", "flats", "houses", "weighted"),
  "till_2030_prio" = c(
    count(immo_prio, timing_prio == 2030)[2, 2, 1],
    count(filter(immo_prio, type == "Apartment" & timing_prio == 2030))[1, 1, 1],
    count(filter(immo_prio, type == "House" & timing_prio == 2030))[1, 1, 1],
    count(filter(immo_prio, type == "House" & timing_prio == 2030))[1, 1, 1] +
    count(filter(immo_prio, type == "Apartment" & timing_prio == 2030))[1, 1, 1]
    / apartment_per_building
  ),
  "till_2040_prio" = c(
    count(immo_prio, timing_prio == 2040)[2, 2, 1],
```

```
    count(filter(immo_prio, type == "Apartment" & timing_prio == 2040))[1, 1, 1],
    count(filter(immo_prio, type == "House" & timing_prio == 2040))[1, 1, 1],
    count(filter(immo_prio, type == "House" & timing_prio == 2040))[1, 1, 1] +
      count(filter(immo_prio, type == "Apartment" & timing_prio == 2040))[1, 1, 1]
    / apartment_per_building
  ),
  "till_2050_prio" = c(
    count(immo_prio, timing_prio == 2050)[2, 2, 1],
    count(filter(immo_prio, type == "Apartment" & timing_prio == 2050))[1, 1, 1],
    count(filter(immo_prio, type == "House" & timing_prio == 2050))[1, 1, 1],
    count(filter(immo_prio, type == "House" & timing_prio == 2050))[1, 1, 1] +
     count(filter(immo_prio, type == "Apartment" & timing_prio == 2050))[1, 1, 1]
    / apartment_per_building
  ),
  "till_2060_prio" = c(
    count(immo_prio, timing_prio == 2060)[2, 2, 1],
    count(filter(immo_prio, type == "Apartment" & timing_prio == 2060))[1, 1, 1],
    count(filter(immo_prio, type == "House" & timing_prio == 2060))[1, 1, 1],
    count(filter(immo_prio, type == "House" & timing_prio == 2060))[1, 1, 1] +
     count(filter(immo_prio, type == "Apartment" & timing_prio == 2060))[1, 1, 1]
    / apartment_per_building
  )
)
timingtable_prio
```

```
##        what till_2030_prio till_2040_prio till_2050_prio till_2060_prio
## 1:      all       37099.00       56897.00       61577.00       45376.00
## 2:    flats       16121.00       27594.00       33074.00       28414.00
## 3:   houses       20978.00       29303.00       28503.00       16962.00
## 4: weighted       23331.43       33331.32       33331.32       21110.03
```

3.2 Randomized Renovation

To illustrate the impact of a prioritization of renovations, we can alternatively assume that renovations are randomly distributed. We can therefore generate a random year to simulate such a random distribution of renovations.

```
set.seed(1301)
immo_rand <- mutate(immo_prio,
                    year_renovation_rand =
                      ifelse(ec_index >=60,
                             floor(runif(nrow(immo_prio),min=0, max=99)/
                             (rate_renovation*(1+
                             (nrow(filter(immo_prio,ec_index<60))/nrow(immo_prio))))
                      +2024),NA),
                    timing_rand          =
                      ifelse(year_renovation_rand <= 2030, 2030,
                      ifelse(year_renovation_rand <= 2040, 2040,
                      ifelse(year_renovation_rand <= 2050, 2050,
                      ifelse(year_renovation_rand <= 2060, 2060,
                                                           NA))))
                    )
```

We can now generate a timing table for the renovations.

```
timingtable_rand <- data.table(
  "what" = c("all", "flats", "houses", "weighted"),
  "till_2030" = c(
    count(immo_rand, timing_rand == 2030)[2, 2, 1],
    count(filter(immo_rand, type == "Apartment" & timing_rand == 2030))[1, 1, 1],
    count(filter(immo_rand, type == "House" & timing_rand == 2030))[1, 1, 1],
    count(filter(immo_rand, type == "House" & timing_rand == 2030))[1, 1, 1] +
    count(filter(immo_rand, type == "Apartment" & timing_rand == 2030))[1, 1, 1]
    / apartment_per_building
  ),
  "till_2040" = c(
    count(immo_rand, timing_rand == 2040)[2, 2, 1],
    count(filter(immo_rand, type == "Apartment" & timing_rand == 2040))[1, 1, 1],
    count(filter(immo_rand, type == "House" & timing_rand == 2040))[1, 1, 1],
    count(filter(immo_rand, type == "House" & timing_rand == 2040))[1, 1, 1] +
    count(filter(immo_rand, type == "Apartment" & timing_rand == 2040))[1, 1, 1]
    / apartment_per_building
  ),
  "till_2050" = c(
    count(immo_rand, timing_rand == 2050)[2, 2, 1],
    count(filter(immo_rand, type == "Apartment" & timing_rand == 2050))[1, 1, 1],
    count(filter(immo_rand, type == "House" & timing_rand == 2050))[1, 1, 1],
    count(filter(immo_rand, type == "House" & timing_rand == 2050))[1, 1, 1] +
    count(filter(immo_rand, type == "Apartment" & timing_rand == 2050))[1, 1, 1]
    / apartment_per_building
  ),
  "till_2060" = c(
    count(immo_rand, timing_rand == 2060)[2, 2, 1],
    count(filter(immo_rand, type == "Apartment" & timing_rand == 2060))[1, 1, 1],
    count(filter(immo_rand, type == "House" & timing_rand == 2060))[1, 1, 1],
    count(filter(immo_rand, type == "House" & timing_rand == 2060))[1, 1, 1] +
    count(filter(immo_rand, type == "Apartment" & timing_rand == 2060))[1, 1, 1]
    / apartment_per_building
  )
)

timingtable_rand
```

```
##        what till_2030 till_2040 till_2050 till_2060
## 1:      all  42308.00  60491.00  60095.00  19311.00
## 2:    flats  22200.00  31622.00  31751.00  10098.00
## 3:   houses  20108.00  28869.00  28344.00   9213.00
## 4: weighted  23348.88  33485.35  32979.18  10687.16
```

3.3 Save and export the data

```
immo_small <- select(immo_rand,
                     energy_use, energy_use_direct, ghg_original, ghg_direct,
                     ec_index, potential_saving_kW, potential_saving_ghg_old,
```

```
                        potential_saving_kW_direct, potential_saving_ghg_direct,
                        potential_saving_ghg_neutral, potential_saving_ghg_decarbDH,
                        year_renovation_prio, year_renovation_rand, firing_final,
                        renovation_weight, partial_renovation)

write.csv(immo_small,
          here::here("Results/immo_reduced_with_renovation.csv"),
          row.names = FALSE)
```

4 Assessing Carbon Neutrality

4.1 Calculating progress in emission reductions

This section calculates relative emissions reductions over the whole time-span of interest. We first calculate sums for each year and accumulate these sums in a second step.

```
progress_fct <- function(year) {
  # Filter rows based on the specified year for renovation_prio and renovation_rand
  subdata_prio <- immo_small[immo_small$year_renovation_prio == year, ]
  subdata_rand <- immo_small[immo_small$year_renovation_rand == year, ]

  # Calculate sums of relevant variables for each year
  sum_saved_kw_direct_prio <- sum(subdata_prio$potential_saving_kW_direct,
                                  na.rm=TRUE)
  sum_saved_ghg_direct_prio <- sum(subdata_prio$potential_saving_ghg_direct,
                                   na.rm=TRUE)
  sum_saved_kw_prio <- sum(subdata_prio$potential_saving_kW, na.rm=TRUE)
  sum_saved_ghg_old_prio <- sum(subdata_prio$potential_saving_ghg_old,
                                na.rm=TRUE)
  sum_saved_ghg_neutral_prio <- sum(subdata_prio$potential_saving_ghg_neutral,
                                    na.rm=TRUE)
  sum_saved_ghg_decarb_prio <- sum(subdata_prio$potential_saving_ghg_decarbDH,
                                   na.rm=TRUE)

  # Calculate sums of relevant variables for each year
  sum_saved_kw_direct_rand <- sum(subdata_rand$potential_saving_kW_direct,
                                  na.rm=TRUE)
  sum_saved_ghg_direct_rand <- sum(subdata_rand$potential_saving_ghg_direct,
                                   na.rm=TRUE)
  sum_saved_kw_rand <- sum(subdata_rand$potential_saving_kW, na.rm=TRUE)
  sum_saved_ghg_old_rand <- sum(subdata_rand$potential_saving_ghg_old,
                                na.rm=TRUE)
  sum_saved_ghg_neutral_rand <- sum(subdata_rand$potential_saving_ghg_neutral,
                                    na.rm=TRUE)
  sum_saved_ghg_decarb_rand <- sum(subdata_rand$potential_saving_ghg_decarbDH,
                                   na.rm=TRUE)

  # Create a DataFrame with the results
  progress_df <- data.frame(
    "year" = year,
    "sum_saved_kw_direct_prio" = sum_saved_kw_direct_prio,
    "sum_saved_ghg_direct_prio" = sum_saved_ghg_direct_prio,
```

```
    "sum_saved_kw_prio" = sum_saved_kw_prio,
    "sum_saved_ghg_old_prio" = sum_saved_ghg_old_prio,
    "sum_saved_ghg_neutral_prio" = sum_saved_ghg_neutral_prio,
    "sum_saved_ghg_decarb_prio" = sum_saved_ghg_decarb_prio,
    "sum_saved_kw_direct_rand" = sum_saved_kw_direct_rand,
    "sum_saved_ghg_direct_rand" = sum_saved_ghg_direct_rand,
    "sum_saved_kw_rand" = sum_saved_kw_rand,
    "sum_saved_ghg_old_rand" = sum_saved_ghg_old_rand,
    "sum_saved_ghg_neutral_rand" = sum_saved_ghg_neutral_rand,
    "sum_saved_ghg_decarb_rand" = sum_saved_ghg_decarb_rand
  )

  return(progress_df)
}

# Create an empty list to store the results for each year
result_list <- list()

# Loop through each year and calculate the DataFrame for each year
start_year <- 2023
end_year <- 2060
for (year in start_year:end_year){
  result_list[[as.numeric(year)]] <- progress_fct(year)
}

# Combine the DataFrames into a single DataFrame
immo_progress <- do.call(rbind, result_list)
```

```
immo_progress <- mutate(immo_progress,
               cum_saved_kw_direct_prio   = cumsum(sum_saved_kw_direct_prio),
               cum_saved_ghg_direct_prio  = cumsum(sum_saved_ghg_direct_prio),
               cum_saved_kw_prio      = cumsum(sum_saved_kw_prio),
               cum_saved_ghg_old_prio     = cumsum(sum_saved_ghg_old_prio),
               cum_saved_ghg_neutral_prio = cumsum(sum_saved_ghg_neutral_prio),
               cum_saved_ghg_decarb_prio  = cumsum(sum_saved_ghg_decarb_prio),
               cum_saved_kw_direct_rand   = cumsum(sum_saved_kw_direct_rand),
               cum_saved_ghg_direct_rand  = cumsum(sum_saved_ghg_direct_rand),
               cum_saved_kw_rand      = cumsum(sum_saved_kw_rand),
               cum_saved_ghg_old_rand     = cumsum(sum_saved_ghg_old_rand),
               cum_saved_ghg_neutral_rand = cumsum(sum_saved_ghg_neutral_rand),
               cum_saved_ghg_decarb_rand  = cumsum(sum_saved_ghg_decarb_rand)
                      )
```

Add values of the remaining ghg and kw:

```
immo_progress <- mutate(immo_progress,
               remain_kw_direct_prio   = sum(immo_small$energy_use_direct)-
                 cum_saved_kw_direct_prio,
               remain_ghg_direct_prio  = sum(immo_small$ghg_direct)-
                 cum_saved_ghg_direct_prio,
               remain_kw_prio          = sum(immo_small$energy_use)-
                 cum_saved_kw_prio,
               remain_ghg_old_prio     = sum(immo_small$ghg_original)-
```

```
    cum_saved_ghg_old_prio,
  remain_ghg_neutral_prio = sum(immo_small$ghg_original)-
    cum_saved_ghg_neutral_prio,
  remain_ghg_decarb_prio  = sum(immo_small$ghg_original)-
    cum_saved_ghg_decarb_prio,
  remain_kw_direct_rand   = sum(immo_small$energy_use_direct)-
    cum_saved_kw_direct_rand,
  remain_ghg_direct_rand  = sum(immo_small$ghg_direct)-
    cum_saved_ghg_direct_rand,
  remain_kw_rand          = sum(immo_small$energy_use)-
    cum_saved_kw_rand,
  remain_ghg_old_rand     = sum(immo_small$ghg_original)-
    cum_saved_ghg_old_rand,
  remain_ghg_neutral_rand = sum(immo_small$ghg_original)-
    cum_saved_ghg_neutral_rand,
  remain_ghg_decarb_rand  = sum(immo_small$ghg_original)-
    cum_saved_ghg_decarb_rand
        )
```

Transform the data to the relative savings of ghg and kW:

```
immo_progress <- mutate(immo_progress,
              relative_kw_direct_prio   = (sum(immo_small$energy_use_direct)-
                                             cum_saved_kw_direct_prio)/
                                             sum(immo_small$energy_use_direct),
              relative_ghg_direct_prio  = (sum(immo_small$ghg_direct)-
                                             cum_saved_ghg_direct_prio)/
                                             sum(immo_small$ghg_direct),
              relative_kw_prio          = (sum(immo_small$energy_use)-
                                             cum_saved_kw_prio)/
                                             sum(immo_small$energy_use),
              relative_ghg_old_prio     = (sum(immo_small$ghg_original)-
                                             cum_saved_ghg_old_prio)/
                                             sum(immo_small$ghg_original),
              relative_ghg_neutral_prio = (sum(immo_small$ghg_original)-
                                             cum_saved_ghg_neutral_prio)/
                                             sum(immo_small$ghg_original),
              relative_ghg_decarb_prio  = (sum(immo_small$ghg_original)-
                                             cum_saved_ghg_decarb_prio)/
                                             sum(immo_small$ghg_original),
              relative_kw_direct_rand   = (sum(immo_small$energy_use_direct)-
                                             cum_saved_kw_direct_rand)/
                                             sum(immo_small$energy_use_direct),
              relative_ghg_direct_rand  = (sum(immo_small$ghg_direct)-
                                             cum_saved_ghg_direct_rand)/
                                             sum(immo_small$ghg_direct),
              relative_kw_rand          = (sum(immo_small$energy_use)-
                                             cum_saved_kw_rand)/
                                             sum(immo_small$energy_use),
              relative_ghg_old_rand     = (sum(immo_small$ghg_original)-
                                             cum_saved_ghg_old_rand)/
                                             sum(immo_small$ghg_original),
              relative_ghg_neutral_rand = (sum(immo_small$ghg_original)-
```

```
                                        cum_saved_ghg_neutral_rand)
                                        /sum(immo_small$ghg_original),
            relative_ghg_decarb_rand  = (sum(immo_small$ghg_original)-
                                        cum_saved_ghg_decarb_rand)/
                                        sum(immo_small$ghg_original)
                    )
```

Export `immo_progress` to save the results:

```
write.csv(immo_progress, here::here('Intermediate_Results', paste("immo_progress_",
                                    rate_renovation, ".csv", sep = "")),
                                    row.names=FALSE)
```

Set colour schemes and data for plots

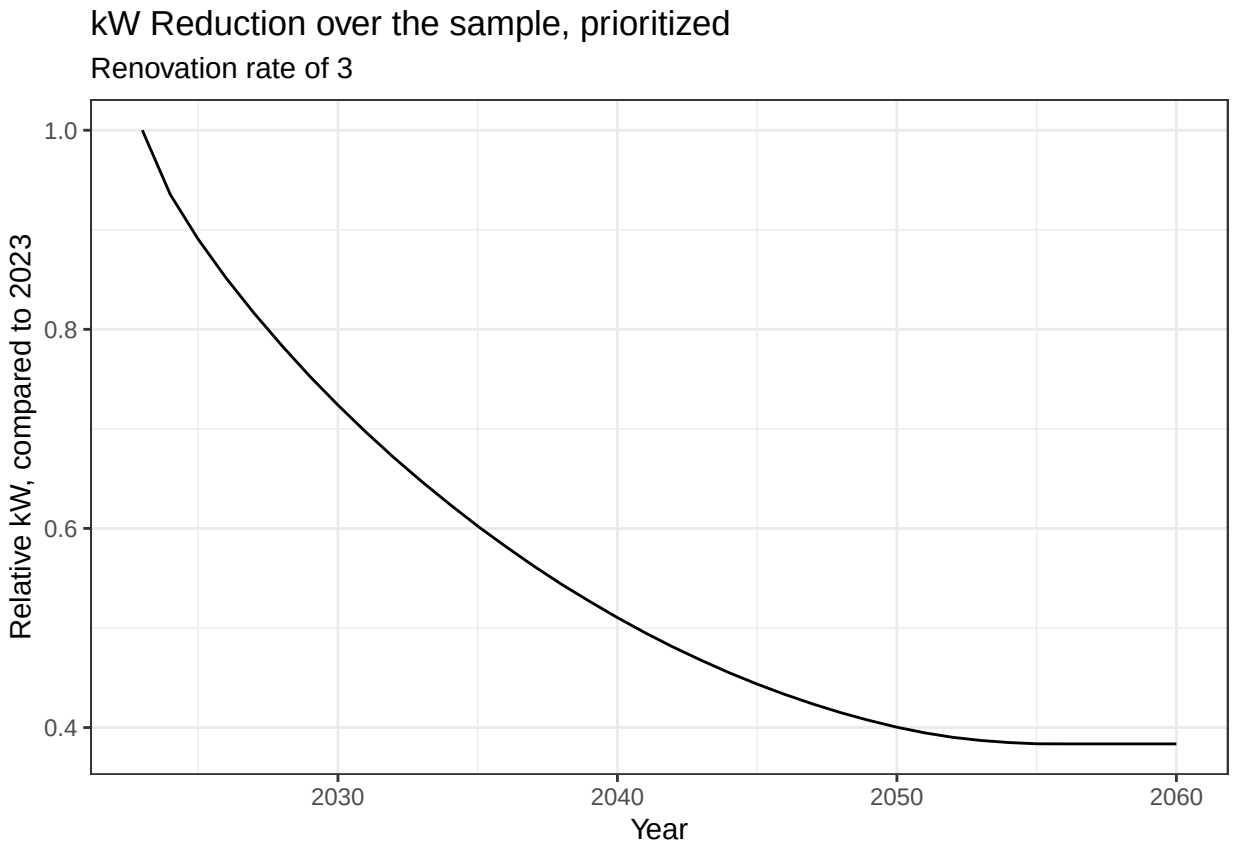
```
immo_progress_truncated <-  subset(immo_progress,
                           year >= min(climate_goals$year) &
                           year <= max(climate_goals$year))

colors4 <-  scales::hue_pal()(8)[1:4]
colors8 <-  scales::hue_pal()(8)[1:8]
line_colors4 <- c("black", colors4)
line_colors8 <- c("black", colors8)
```

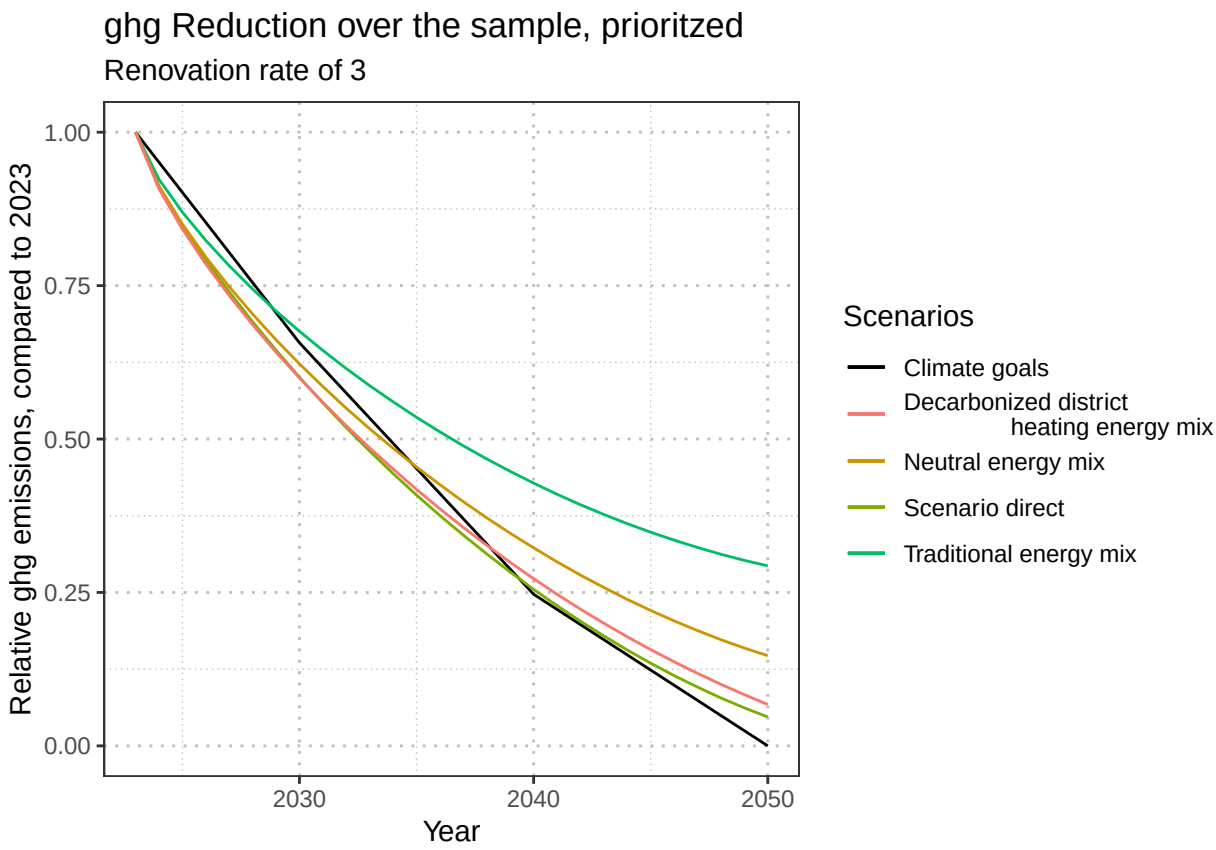
4.2 Plot Progress

Create plot to illustrate progress for the prioritized renovations in different scenarios.

```
plot_prio_kw <- ggplot(immo_progress, aes(x=year))+
  geom_line(aes(y=relative_kw_prio))+
  ggtitle(label="kW Reduction over the sample, prioritized",
          subtitle=paste("Renovation rate of",rate_renovation))+
  labs(x="Year", y="Relative kW, compared to 2023") +
  theme_bw()
plot_prio_kw
```

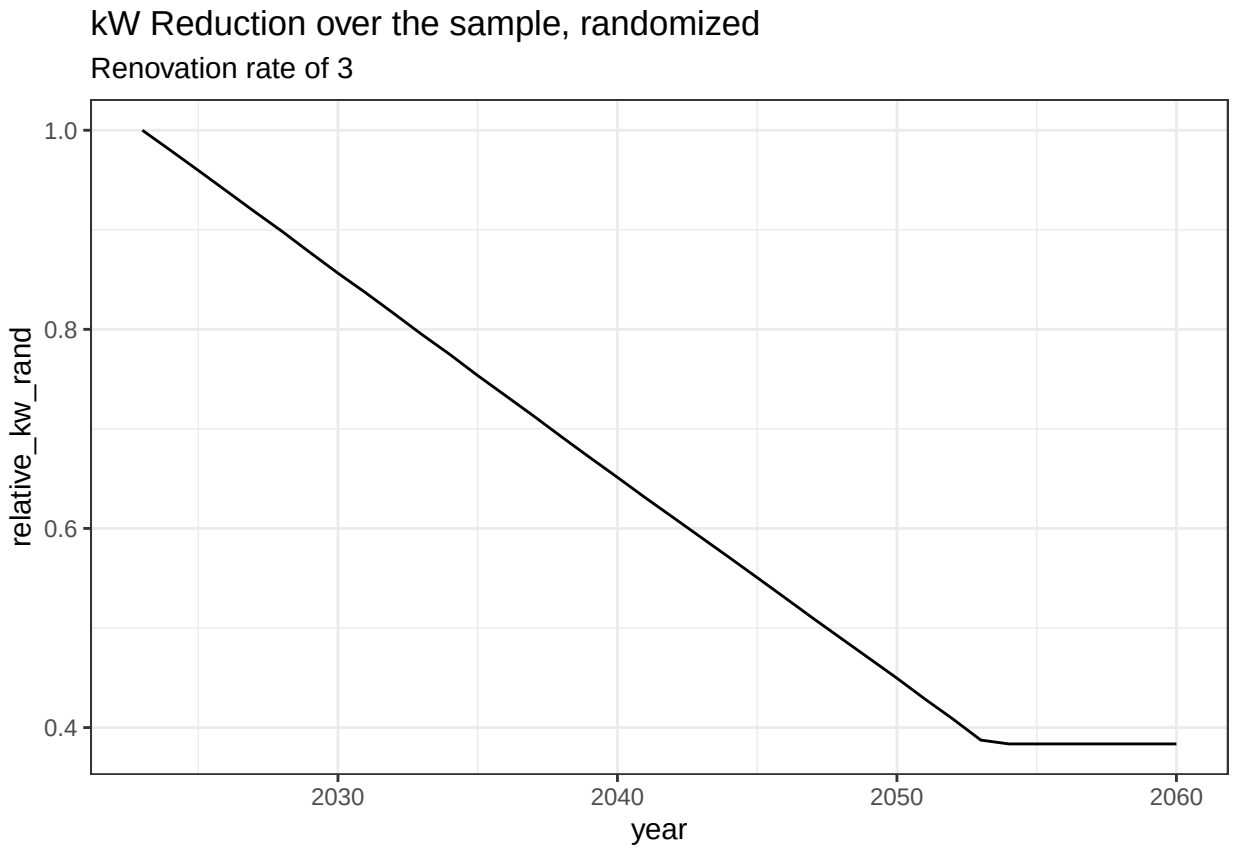



```
plot_prio_ghg <- ggplot(immo_progress_truncated, aes(x=year))+
  geom_line(aes(y = climate_goals$rel_reduction, color = "Climate goals")) +
  geom_line(aes(y=relative_ghg_direct_prio, color = "Scenario direct"))+
  geom_line(aes(y=relative_ghg_old_prio, color = "Traditional energy mix"))+
  geom_line(aes(y=relative_ghg_neutral_prio, color = "Neutral energy mix"))+
  geom_line(aes(y=relative_ghg_decarb_prio, color = "Decarbonized district
                heating energy mix"))+
  scale_color_manual(name = "Scenarios", values = line_colors4)+
  ggtitle(label="ghg Reduction over the sample, prioritzed",
          subtitle=paste("Renovation rate of",rate_renovation))+
  labs(x = "Year", y = "Relative ghg emissions, compared to 2023") +
  theme_bw()+
  theme(panel.grid = element_line(color = "gray", linetype = "dotted"))
plot_prio_ghg
```

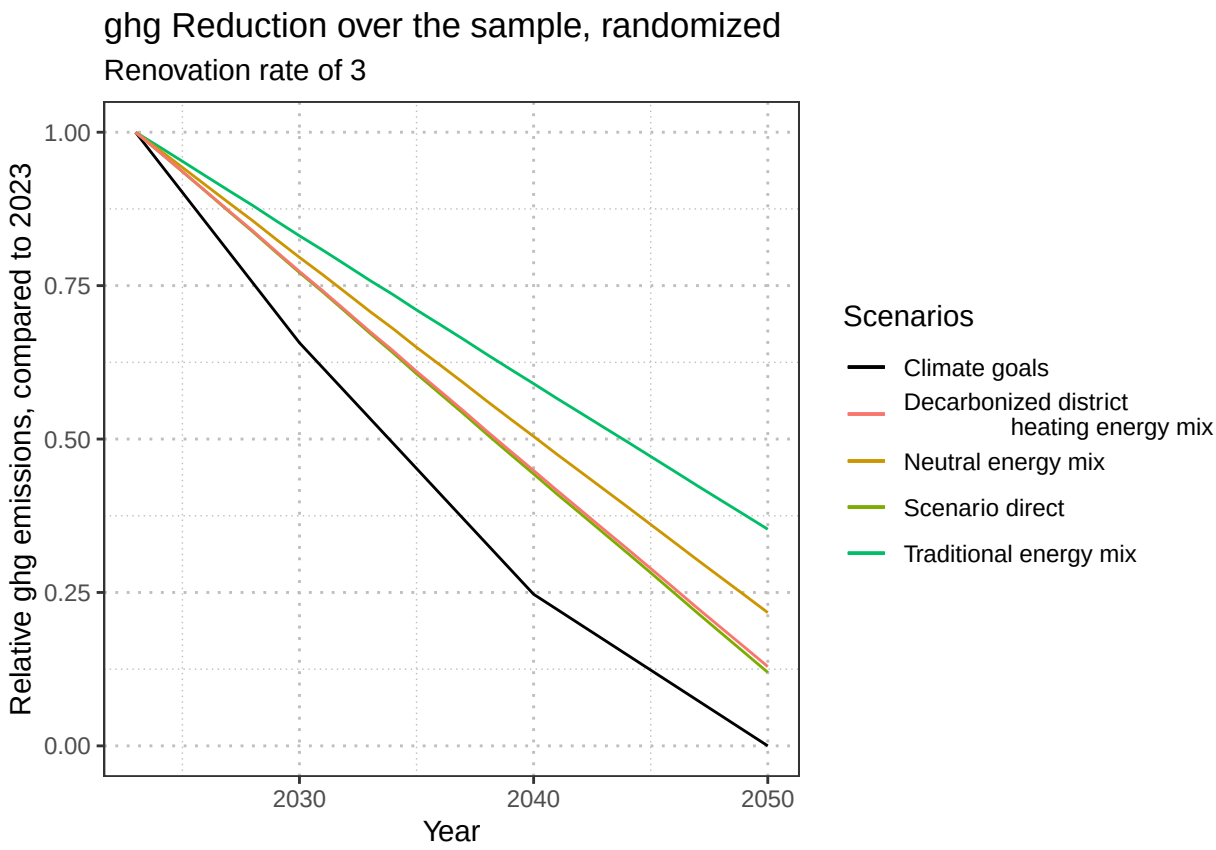


Create plot to illustrate progress for the randomized renovations in different scenarios.

```
plot_rand_kw <- ggplot(immo_progress, aes(x=year))+
  geom_line(aes(y=relative_kw_rand))+
  ggtitle(label="kW Reduction over the sample, randomized",
          subtitle=paste("Renovation rate of",rate_renovation))+
  theme_bw()
plot_rand_kw
```

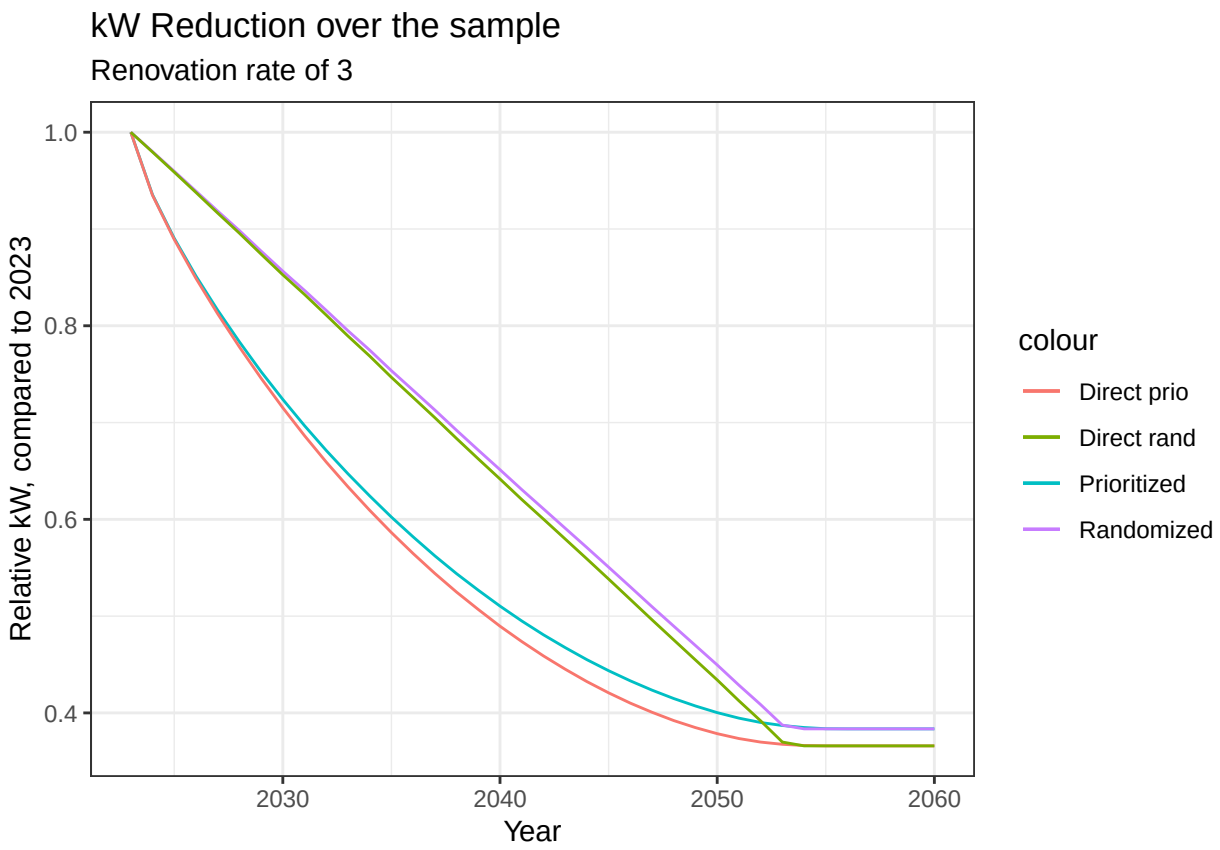



```
plot_rand_ghg <- ggplot(immo_progress_truncated, aes(x=year))+
  geom_line(aes(y = climate_goals$rel_reduction, color = "Climate goals")) +
  geom_line(aes(y=relative_ghg_direct_rand, color = "Scenario direct"))+
  geom_line(aes(y=relative_ghg_old_rand, color = "Traditional energy mix"))+
  geom_line(aes(y=relative_ghg_neutral_rand, color = "Neutral energy mix"))+
  geom_line(aes(y=relative_ghg_decarb_rand, color = "Decarbonized district
                heating energy mix"))+
  scale_color_manual(name = "Scenarios", values = line_colors4)+
  ggtitle(label="ghg Reduction over the sample, randomized",
          subtitle=paste("Renovation rate of",rate_renovation))+
  labs(x="Year", y="Relative ghg emissions, compared to 2023") +
  theme_bw()+
  theme(panel.grid = element_line(color = "gray", linetype = "dotted"))
plot_rand_ghg
```

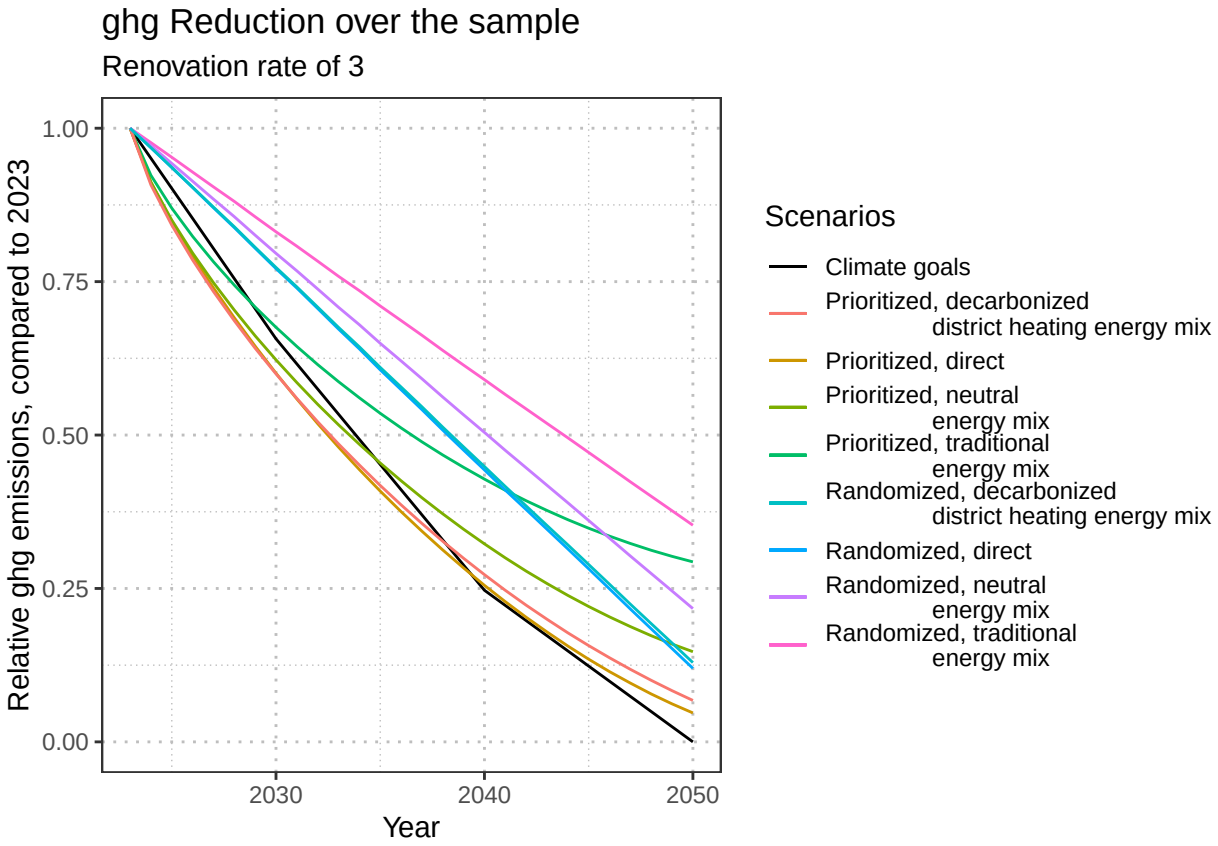


Create plot to compare progress for the prioritized and randomized renovations.

```
plot_compare_kw <- ggplot(immo_progress, aes(x=year))+
  geom_line(aes(y=relative_kw_prio, color = "Prioritized"))+
  geom_line(aes(y=relative_kw_rand, color = "Randomized"))+
  geom_line(aes(y=relative_kw_direct_prio, color = "Direct prio"))+
  geom_line(aes(y=relative_kw_direct_rand, color = "Direct rand"))+
  ggtitle(label="kW Reduction over the sample",
          subtitle=paste("Renovation rate of",rate_renovation))+
  labs(x="Year", y="Relative kW, compared to 2023") +
  theme_bw()
plot_compare_kw
```




```
plot_compare_ghg <- ggplot(immo_progress_truncated, aes(x=year))+
  geom_line(aes(y = climate_goals$rel_reduction, color = "Climate goals")) +
  geom_line(aes(y=relative_ghg_direct_prio, color = "Prioritized, direct"))+
  geom_line(aes(y=relative_ghg_old_prio, color = "Prioritized, traditional
                energy mix"))+
  geom_line(aes(y=relative_ghg_neutral_prio, color = "Prioritized, neutral
                energy mix"))+
  geom_line(aes(y=relative_ghg_decarb_prio, color = "Prioritized, decarbonized
                district heating energy mix"))+
  geom_line(aes(y=relative_ghg_direct_rand, color = "Randomized, direct"))+
  geom_line(aes(y=relative_ghg_old_rand, color = "Randomized, traditional
                energy mix"))+
  geom_line(aes(y=relative_ghg_neutral_rand, color = "Randomized, neutral
                energy mix"))+
  geom_line(aes(y=relative_ghg_decarb_rand, color = "Randomized, decarbonized
                district heating energy mix"))+
  scale_color_manual(name = "Scenarios",
                     values = line_colors8)+
  ggtitle(label="ghg Reduction over the sample",
          subtitle=paste("Renovation rate of",rate_renovation))+
  labs(x="Year", y="Relative ghg emissions, compared to 2023") +
  theme_bw()+
  theme(panel.grid = element_line(color = "gray", linetype = "dotted"))
plot_compare_ghg
```



Exporting graphs and intermediary data-sets #

4.3 Exporting data for distributional analysis

We export the relevant variables for the merging our findings with survey data on wealth (for the *German case* we use data provided by the Household Finance and Consumption Survey (HFCS) administered by EZB.

```
immo_export_small <- select(immo_rand, type, renovation_weight, partial_renovation,
                            year_renovation_prio, year_renovation_rand,
                            living_area)

export_fct <- function(year) {
  subdata_prio <- filter(immo_export_small,year_renovation_prio == year)
  subdata_rand <- filter(immo_export_small,year_renovation_rand == year)

  full_renovations_prio <- sum(subdata_prio$renovation_weight)/
                                  sum(immo_export_small$renovation_weight)
  full_renovations_rand <- sum(subdata_rand$renovation_weight)/
                                  sum(immo_export_small$renovation_weight)
  full_renovations_prio_cost <- sum(subdata_prio$renovation_weight*
                                       subdata_prio$partial_renovation)/
                                  sum(immo_export_small$renovation_weight)
  full_renovations_rand_cost <- sum(subdata_rand$renovation_weight*
                                       subdata_rand$partial_renovation)/
                                  sum(immo_export_small$renovation_weight)
  full_renovations_prio_m2 <- sum(subdata_prio$partial_renovation *
```

```
                                  subdata_prio$living_area, na.rm=TRUE)/
                                  sum(immo_export_small$living_area, na.rm=TRUE)
  full_renovations_rand_m2 <- sum(subdata_rand$partial_renovation *
                                  subdata_rand$living_area, na.rm=TRUE)/
                                  sum(immo_export_small$living_area, na.rm=TRUE)
  sqm_prio_partial <- sum(subdata_prio$living_area *
                            subdata_prio$partial_renovation) /
                            sum(subdata_prio$renovation_weight)
  sqm_rand_partial <- sum(subdata_rand$living_area *
                            subdata_rand$partial_renovation) /
                            sum(subdata_rand$renovation_weight)
  share_blocks_of_flats_prio <- sum(select(filter(subdata_prio,type=="Apartment")
                                  ,renovation_weight))/
                              sum(subdata_prio$renovation_weight)
  share_blocks_of_flats_rand <- sum(select(filter(subdata_rand,type=="Apartment")
                                  ,renovation_weight))/
                              sum(subdata_rand$renovation_weight)

  sqm_prio <- sum(subdata_prio$living_area) / sum(subdata_prio$renovation_weight)
  sqm_rand <- sum(subdata_rand$living_area) / sum(subdata_rand$renovation_weight)
  # Create a DataFrame with the results
  export_df <- data.frame(
    "year" = year,
    "full_renovations_prio" = full_renovations_prio,
    "full_renovations_rand" = full_renovations_rand,
    "full_renovations_prio_partial" = full_renovations_prio_cost,
    "full_renovations_rand_partial" = full_renovations_rand_cost,
    "full_renovations_prio_m2" = full_renovations_prio_m2,
    "full_renovations_rand_m2" = full_renovations_rand_m2,
    "sqm_prio" = sqm_prio,
    "sqm_rand" = sqm_rand,
    "sqm_prio_renovated" = sqm_prio_partial,
    "sqm_rand_renovated" = sqm_rand_partial,
    "share_flats_prio"=share_blocks_of_flats_prio,
    "share_flats_rand"=share_blocks_of_flats_rand
  )

  return(export_df)
}

# Create an empty list to store the results for each year
result_list <- list()

# Loop through each year and calculate the Dataframe for each year
for (year in 2023:2060) {
  result_list[[as.numeric(year)]] <- export_fct(year)
}

# Combine the DataFrames into a single DataFrame
immo_export <- do.call(rbind, result_list) %>%
  mutate(rate_renovation = rate_renovation)

filename <- paste(here::here(),"/Intermediate_Results/immo_export_rate_",
```

```
                  rate_renovation, ".csv", sep = "")
write.csv(immo_export, file = filename, row.names = FALSE)
```

4.4 Calculate different renovation rates

In this section we use the data compiled above to calculate different indicators for renovation intensity, where the indicators with suffix `n` counts renovated buildings relative to all buildings, the indicators with suffix `m2` count fully renovated square-meters relative to all square-meters (which takes into account that some building only require partial renovation), which is equivalent to the *full renovation equivalent*. Finally, the indicators with suffix `partial` present an intermediate version that counts renovated buildings relative to all buildings, but takes into account that some building only require partial renovations.

The latter is especially helpful for cost calculation is, at least for the *German case*, very closely aligned with full renovation equivalents.

```
fullrenequ <- summarise(filter(filter(
                        immo_export,year>2023),year<2051),
                        renrate_prio_n=mean(full_renovations_prio),
                        renrate_rand_n=mean(full_renovations_rand),
                        renrate_prio_m2=mean(full_renovations_prio_m2),
                        renrate_rand_m2=mean(full_renovations_rand_m2),
                        renrate_prio_partial=mean(full_renovations_prio_partial),
                        renrate_rand_partial=mean(full_renovations_rand_partial))

fullrenequ
```

```
##   renrate_prio_n renrate_rand_n renrate_prio_m2 renrate_rand_m2
## 1     0.02999992     0.02993969      0.02230958      0.02070035
##   renrate_prio_partial renrate_rand_partial
## 1            0.0237663           0.02233503
```

4.5 Exporting data for input output analysis

In this section we calculate the share of buildings, which will be refitted with a heating pump in the respective year, for both scenarios. These data in turn helps to adjust calculated costs for constructing/renovation insulation so that the latter actually reflect the correct number of heating pumps to be employed (otherwise these costs would be downscaled for those buildings that do not need full renovation, see section 2.2). All this is taken up in the section on costs and distribution.

```
HP_share_fct <- function(year){
  year <- year
  totals <- sum(immo_small$renovation_weight)
  prio_hp_share <- immo_small %>%
    filter(year_renovation_prio == year &
          immo_small$firing_final == "HP(new)") %>%
    summarise(share = sum(renovation_weight) / totals)
  rand_hp_share <- immo_small %>%
    filter(year_renovation_rand == year &
          immo_small$firing_final == "HP(new)") %>%
    summarise(share = sum(renovation_weight) / totals)
```

```
  return(list(year = year,
              prio_hp_share = prio_hp_share$share,
              rand_hp_share = rand_hp_share$share,
              rate_renovation = rate_renovation))
}

hp_share <- t(sapply(2023:2060, HP_share_fct))
```

Export the dataframe.

```
filename <- paste(here::here(), "/Intermediate_Results/hp_share_",
                  rate_renovation, ".csv", sep = "")
write.csv(hp_share, file = filename, row.names = FALSE)
```

4.6 Exporting graphs

```
filename <- paste(here::here(), "/Graphs/plot_prio_kw_", rate_renovation,
                  ".pdf", sep = "")
ggsave(filename, plot=plot_prio_kw, width = 10, height = 6, device = "pdf")
filename <- paste(here::here(), "/Graphs/plot_prio_ghg_", rate_renovation,
                  ".pdf", sep = "")
ggsave(filename, plot=plot_prio_ghg, width = 10, height = 6)
filename <- paste(here::here(), "/Graphs/plot_rand_kw_", rate_renovation,
                  ".pdf", sep = "")
ggsave(filename, plot=plot_rand_kw, width = 10, height = 6)
filename <- paste(here::here(), "/Graphs/plot_rand_ghg_", rate_renovation,
                  ".pdf", sep = "")
ggsave(filename, plot=plot_rand_ghg, width = 10, height = 6)
filename <- paste(here::here(), "/Graphs/plot_compare_kw_", rate_renovation,
                  ".pdf", sep = "")
ggsave(filename, plot=plot_compare_kw, width = 10, height = 6)
```