A tool box for a climate neutral housing sector: Calculating costs and assessing the distributional impact

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

2024-10-26

1 Basic Setup

1.1 Load Packages

1.2 Defining Key Parameters

In this section we define key policy parameters – the renovation rate to be analyzed as well as upper and lower bounds to take into account subsidies. Specifically, below the lower bound all costs will we carried by the public sector, while above the upper bound all costs have to be borne by private households. In between, we use linear extrapolation to determine the subsidized share of total investment costs.

As a renovation subsidy is basically a subsidy on private wealth, we use total net wealth as a key criterion for assessing the distributional status of some household.

```
# policy parameters
rate_to_be_analyzed=3 # choosing a scenario to analyze
first_percentile_to_pay <- 65/100 # policy design parameter
last_percentile_to_get_reduction <- 89/100 # policy design parameter

# supplementary parameters
perc_steps <- 100
start_year <- 2023
years_till_2050 <- 2050 - start_year
gdp_growth <- 1.01
```

1.3 Importing Results From Past Steps

Here we import the results on the impact of renovating and refitting residential buildings. The numbers reported at the end of the chunk refer to the share of renovated buildings (first two lines) as well as the share of fully renovated building relative to all 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
building_outputs <-  fread(here::here(
  paste0("Intermediate_Results/immo_export_rate_", rate_to_be_analyzed, ".csv")))
immo_hp <- fread(here::here(
  paste0("Intermediate_Results/hp_share_", rate_to_be_analyzed, ".csv")))
building_outputs <- building_outputs %>% left_join(immo_hp, by = "year")
```

2 Calculating Costs

2.1 Some Key Parameters for Calculating Costs

Defining key technical parameters for calculating renovation costs. In part, these parameters can be obtained from data on residential buildings. Other parameters have to be reconstructed from external sources. In our case we use total renovation costs from 2014 (as given by the DIW), standard inflation data from Destatis.

- `share of privately hold buildings` represents the share of buildings owned by private households Source
- `average_size_mfh` The average size of apartment buildings (multi-family houses). Taken from complete code
- `average_size_stfh` The average size of single and two family buildings. Taken from complete code
- `n_buildings` Number of residential buildings in Germany Source
- `n_mfc` Number of apartment buildings in Germany Source
- `n_buildings` Number of single- and two family buildings in Germany Source
- `rate_mfh_2014` Current full renovation equivalents (energetic renovations) of apartment buildings. We implicitly assume that this rate did not change over the last years. Source
- `rate_stfh_2014` Current full renovation equivalents (energetic renovations) of single and two family buildings. We implicitly assume that this rate did not change over the last years. Source
- `cost_2014` The amount spent on energetic renovations in 2014 Source
- `value_factor_2023` Gives information about the development of prices in the sector since 2014 Source
- `gdp` German GDP in 2023 Source
- `cost_hp` The average cost of a heat-pump Source
- `weighted_rate_n_buildings` The number of buildings that undergo a full renovation. This number will be used in our cost calculations.
- `weighted_rate_sqm` The share of living area that undergos a full renovation. This is equivalent to the standard definition of the full renovation equivalent.

```r
share_of_privately_owned_buildings = 0.78 # share of residential buildings
                                          # owned by private households
average_size_mfh = 492.0391
average_size_stfh = 173.0257

n_mfh = 3270414
n_stfh = 16105498
```

```
n_buildings = n_mfh + n_stfh
rate_mfh_2014 = 0.014
rate_stfh_2014 = 0.01

cost_2014 = 34780000000
cost_index_2023 = 162.9
cost_index_2014 = 97.8
value_factor_2023 = cost_index_2023 / cost_index_2014
gdp = 4122.21e9
cost_hp = 35083.6
rate = rate_to_be_analyzed

share_mfh = n_mfh / n_buildings
share_stfh = 1 - share_mfh
weighted_rate_n_buildings = rate_mfh_2014 * share_mfh +
                            rate_stfh_2014 * share_stfh
rate_io = weighted_rate_n_buildings

share_mfh2 = n_mfh*average_size_mfh /
  (n_mfh*average_size_mfh+n_stfh*average_size_stfh)
share_stfh2 = 1 - share_mfh2
weighted_rate_sqm = rate_mfh_2014 * share_mfh2 + rate_stfh_2014 * share_stfh2
full_renovation_equivalent = weighted_rate_sqm
```

2.2 Compute yearly investment

The function `get_yearly_costs` upscales renovation costs per building to costs to be expected per year all relevant scenarios. - prio denotes the scenario where worst performing buildings are prioritized - rand denotes the scenario where buildings are renovated in a random order

We consider the overall costs of renovations as calculated above and additional costs, which might arise from the exchange of heating systems in buildings, which do not require renovation otherwise.

```
get_yearly_costs <- function(building_results){
  building_results <- building_results %>%
    mutate(renovation_cost_prio = n_buildings * share_flats_prio *
                                  full_renovations_prio_partial *
                                  cost_renovation_mfh +
                                  n_buildings * (1 - share_flats_prio) *
                                  full_renovations_prio_partial *
                                  cost_renovation_stfh,
      hp_cost_prio = n_buildings * excess_hp_prio * cost_hp,
      yearly_cost_prio = renovation_cost_prio + hp_cost_prio,
      renovation_cost_rand = n_buildings * share_flats_rand *
                             full_renovations_rand_partial * cost_renovation_mfh
                             + n_buildings * (1-share_flats_rand) *
                             full_renovations_rand_partial * cost_renovation_stfh,
      hp_cost_rand = n_buildings * excess_hp_rand * cost_hp,
      yearly_cost_rand = renovation_cost_rand + hp_cost_rand,
      io_status_quo = rate_mfh_2014 * share_mfh * n_buildings *
                      cost_renovation_mfh +
                      rate_stfh_2014 * share_stfh * n_buildings *
                      cost_renovation_stfh
```

```
    )
    building_results <- building_results %>%
    mutate(io_cost_prio = ifelse(full_renovations_prio_partial > rate_io,
                                 pmax(yearly_cost_prio - io_status_quo,
                                     more_hps_prio * n_buildings * cost_hp),
                                   more_hps_prio * n_buildings * cost_hp),
           io_cost_rand = ifelse(full_renovations_rand_partial > rate_io,
                                 pmax(yearly_cost_rand - io_status_quo,
                                      more_hps_rand * n_buildings * cost_hp),
                                  more_hps_rand * n_buildings * cost_hp),
           share_gdp_prio = io_cost_prio / (gdp * gdp_growth ^ (year - start_year)),
           share_gdp_rand = io_cost_rand / (gdp * gdp_growth ^ (year - start_year)),
           yearly_io_hp_cost = n_buildings * more_hps_prio * cost_hp)

  selected_columns <-
    building_results[, c("year", "yearly_cost_rand", "yearly_cost_prio",
                         "share_gdp_prio", "share_gdp_rand", "io_cost_prio",
                         "io_cost_rand", "io_status_quo", "yearly_io_hp_cost",
                         "more_hps_prio", "renovation_cost_prio", "hp_cost_prio")]
  return(selected_columns)
}
```

2.3 Calculating Renovation Costs per House

In this section we calculate the average costs per renovated builing by extrapolating costs from a specified base year (in our case: 2014). We thereby take into account that different types of buildings differ in terms of size and consider this in the extrapolation process.

```
n_renovated_mfh_2014 = rate_mfh_2014 * n_mfh
n_renovated_stfh_2014 = rate_stfh_2014 * n_stfh

share_fixed = 0.25
share_variable = 1 - share_fixed
fixed_costs_2014 = cost_2014 * share_fixed /
                   (n_renovated_stfh_2014 + n_renovated_mfh_2014)
variable_costs_stfh_2014 = cost_2014 * share_variable /
                           (n_renovated_stfh_2014 + n_renovated_mfh_2014 *
                            average_size_mfh / average_size_stfh )

cost_renovation_stfh = (fixed_costs_2014 + variable_costs_stfh_2014) *
                        value_factor_2023
cost_renovation_mfh = (fixed_costs_2014 + variable_costs_stfh_2014 *
                       average_size_mfh / average_size_stfh) * value_factor_2023
average_renovation_cost = (cost_renovation_stfh * n_renovated_stfh_2014 +
                           cost_renovation_mfh * n_renovated_mfh_2014) /
                          (n_renovated_stfh_2014 + n_renovated_mfh_2014)

cost_of_full_renovation = average_renovation_cost * value_factor_2023
```

2.4 Calculating Renovation Costs per Year

In this section we upscale renovation costs using the function `get_yearly_costs`, introduced in section **??** (although for the *German case* the costs associated with refitting only are minimal).

```
building_outputs <- building_outputs %>%
  rowwise() %>%
  mutate(excess_hp_prio = max(prio_hp_share - full_renovations_prio_partial, 0),
         excess_hp_rand = max(rand_hp_share - full_renovations_rand_partial, 0)) %>%
  mutate(
    more_hps_prio = max(prio_hp_share - rate_io, 0),
    more_hps_rand = max(rand_hp_share - rate_io, 0)
  ) %>%
  ungroup()

yearly_financing <- get_yearly_costs(building_outputs)
yearly_financing <- yearly_financing %>%
    filter(year > min(year),
           year < 2051)

cost_of_renovation_prio <- sum(yearly_financing$yearly_cost_prio)
cost_of_renovation_random <- sum(yearly_financing$yearly_cost_rand)
io_cost_of_renovation <- sum(yearly_financing$io_cost_prio)
```

3 Distributional Analysis

3.1 Introducing Key Functions

This first set of functions helps formatting the respective data set by introducing percentile limits (in `ceiling_dec` and `get_percentile`).

The function `datamanipulation` extracts the relevant variables from the Household Finance and Consumption Survey. If another data source has to be used, this function has to be adapted accordingly.In detail, this function mainly reformats the data, e.g. by recoding the absence of housing wealth from `NA` to 0. It also sums up all residential wealth associated with the respective household, calculates residential wealth as share of net and gross wealth. It also adds the income of respective households (which is not needed, but can be employed to design alternative policy implementation strategies) and prepares fundamentals for plotting Lorenz curves.

```
ceiling_dec <- function(x, level=1) round(x + 5*10^(-level-1), level)

get_percentile <- function(df, quant_steps, order_by, var_name, dec_name){
  perc_interval <- 1 / quant_steps
  dec_interval = 0.1
  df <- df %>%
    arrange(get(order_by)) %>%
    mutate(
      cum_weight = cumsum(weight),
      cum_weight_share = cumsum(weight) / sum(weight),
      percentile_exact = frollmean(c(0,cum_weight_share),2)[-1],
      percentile = ceiling_dec(percentile_exact,2),
      decile = ceiling_dec(percentile_exact,1)) %>%
```

```r
    rename(!!var_name :=percentile,
           !!dec_name :=decile)
  return(df)
}

datamanipulation <- function(df, perc_steps){
   df <- df %>%
     rename(grosswealth = da3001) %>%
     mutate(main_residence = replace(da1110, is.na(da1110), 0)) %>%
     mutate(immo_val1 = ifelse(hb2501 %in% c(1,2), hb2801*hb2701/100, 0),
        immo_val2 = ifelse(hb2502 %in% c(1,2), hb2802*hb2702/100, 0),
        immo_val3 = ifelse(hb2503 %in% c(1,2), hb2803*hb2703/100, 0)) %>%
    mutate(immo_val = immo_val1 + immo_val2 + immo_val3 + main_residence) %>%
    mutate(immo_in_net = ifelse(nwealth <= 0, NA,
                                       immo_val/nwealth)) %>%
    mutate(immo_in_gross = ifelse(grosswealth==0, 0, immo_val/grosswealth)) %>%
    mutate(home_in_net = ifelse(nwealth <= 0, NA, main_residence/nwealth)) %>%
    mutate(home_in_gross =
             ifelse(grosswealth==0, 0, main_residence/grosswealth)) %>%
    rename(income_eq = di2000eq) %>%
    rename(income = di2000)

    df <- get_percentile(df, perc_steps,
                         "nwealth", "wealth_percentile", "wealth_decile")
    df <- get_percentile(df, perc_steps,
                         "grosswealth", "gross_percentile", "gross_decile")
    df <- get_percentile(df, perc_steps,
                         "immo_val", "immo_percentile", "immo_decile")
    df <- get_percentile(df, perc_steps,
                         "income_eq", "income_percentile", "income_decile")

   df_gross <- df %>%
      arrange(gross_percentile) %>%
      mutate(lorenz = cumsum(grosswealth*weight) / sum(grosswealth*weight) * 100,
             perc = cumsum(weight) / sum(weight) * 100)

   df_nwealth <- df %>%
     arrange(wealth_percentile) %>%
      mutate(lorenz = cumsum(nwealth*weight) / sum(nwealth*weight) * 100,
             perc = cumsum(weight) / sum(weight) * 100)

   df_immoval <- df %>%
        arrange(immo_percentile) %>%
        mutate(lorenz = cumsum(immo_val*weight) / sum(immo_val*weight) * 100,
               perc = cumsum(weight) / sum(weight) * 100)

   df_income <- df %>%
        arrange(income_percentile) %>%
        mutate(lorenz = cumsum(income_eq*weight) / sum(income_eq*weight) * 100,
               perc = cumsum(weight) / sum(weight) * 100)

  df_lorenz <- bind_rows(df_gross %>% select(lorenz, perc) %>%
                           mutate(variable = "Gross Wealth"),
```

```
                          df_nwealth %>% select(lorenz, perc) %>%
                           mutate(variable = "Net wealth"),
                          df_immoval %>% select(lorenz, perc) %>%
                           mutate(variable = "Residential wealth"),
                          df_income %>% select(lorenz, perc) %>%
                           mutate(variable = "Annual income"))

  return(list(df=df, df_lorenz=df_lorenz))
 }
```

This final function, `get_financing`, assesses the private and public costs associated with the renovation efforts. Since we make different assumption about the financing needs of private and institutional owners, this table focuses on privately owned buildings.

```
get_financing <- function(df){
df_finance <- df %>%
    filter(immo_val != 0) %>%
    mutate(total_immo_property = immo_val * weight,
           share_immo_property = total_immo_property /
                                 sum(total_immo_property)) %>%
    rowwise() %>%
    mutate(share_paid_household = max(0, min(1, (wealth_percentile -
                                  first_percentile_to_pay) /
                                  (last_percentile_to_get_reduction -
                                  first_percentile_to_pay))),
           share_paid_private = share_immo_property * share_paid_household,
           share_paid_state = (1 - share_paid_household) *
                              share_immo_property) %>%
    ungroup()

  financing_table <- data.table(
                           "total_cost_prio"=cost_of_renovation_prio,
                           "yearly_prio"=cost_of_renovation_prio /
                                         years_till_2050,
                           "yearly_io_prio"=io_cost_of_renovation /
                                             years_till_2050,
                           "total_cost_random"=cost_of_renovation_random,
                           "yearly_random"= cost_of_renovation_random /
                                            years_till_2050,
                           "yearly_io_random"=(cost_of_renovation_random -
                                               io_cost_of_renovation) /
                                               years_till_2050,
                           "share_private"=sum(df_finance$share_paid_private),
                           "share_state"=sum(df_finance$share_paid_state),
                           "total_public_prio"=sum(df_finance$share_paid_state)*
                            cost_of_renovation_prio*
                            share_of_privately_owned_buildings,
                           "yearly_public_prio"=sum(df_finance$share_paid_state)*
                            cost_of_renovation_prio /
                            years_till_2050*share_of_privately_owned_buildings,
                           "total_public_random"=
                            sum(df_finance$share_paid_state) *
                            cost_of_renovation_random *
```

```
                              share_of_privately_owned_buildings,
                             "yearly_public_random"=
                              sum(df_finance$share_paid_state) *
                              cost_of_renovation_random/years_till_2050 *
                              share_of_privately_owned_buildings)

  return(list(table = financing_table, df = df_finance))
}
```

3.2 Mapping Costs to Households

Finally, we create a loop to run all relevant functions for all implicates in the underlying dataset, thereby considering the fact that most datasets on private wealth are based on multiple imputation. Multiple imputation requires to consider a small number of separate datasets (in case of the HFCS there are five such datasets) and to produce final estimates by meaning across across implicates.

In a first step we import and manipulate available data on private wealth and apply the function that assigns cost to households.

```
# Initialize data
imps <- c(1, 2, 3, 4, 5)

# Loop over imps
for (i in imps) {
  # Read CSV files
  df1 <- read_csv(here::here(paste0("Data/HFCS_countryDE_imp", i, ".csv")),
                  show_col_types = FALSE)
  df2 <- read_csv(here::here(paste0("Data/HFCS_countryDE_imp", i, "_income.csv")),
                  show_col_types = FALSE)

  # Join data frames and manipulate data
  HFCS <- full_join(df1, df2) %>%
    mutate(imp = i)

  manipulated_data <- datamanipulation(HFCS, perc_steps)

  # Collect data implicates
  if (i == 1) {
    collect_data_implicates <- manipulated_data$df
  } else {
    collect_data_implicates <- full_join(collect_data_implicates,
                                         manipulated_data$df)
  }

  # Get financing results and manipulate financing table
  finance_results <- get_financing(manipulated_data$df)
  financing_table <- finance_results$table
  financing_table$implicate <- i

  # Collect results in result_table
  if (i == 1) {
    result_table <- financing_table
  } else {
```

```
    result_table <- merge(result_table, financing_table, all = TRUE)
  }
}
```

In a second step we collect results across all implicates to calculate final estimates.

```
imp_m <- result_table %>%
  dplyr::summarise(
    total_cost_prio = mean(total_cost_prio),
    yearly_prio = mean(yearly_prio),
    yearly_io_prio = mean(yearly_io_prio),
    total_cost_random = mean(total_cost_random),
    yearly_random = mean(yearly_random),
    yearly_io_random = mean(yearly_io_random),
    share_private = mean(share_private),
    share_state = mean(share_state),
    total_public_prio = mean(total_public_prio),
    yearly_public_prio = mean(yearly_public_prio),
    total_public_random = mean(total_public_random),
    yearly_public_random = mean(yearly_public_random)
  )
```

Finally, we export the results.

```
outpath_results <- here::here("Results/hfcs_results.csv")
fwrite(imp_m, outpath_results)
out_table_results <- xtable(imp_m)

outpath_io_results <- here("Intermediate_Results/results_for_io.csv")
fwrite(yearly_financing, outpath_io_results)
out_table_results <- xtable(imp_m)
```

4 Visualisations

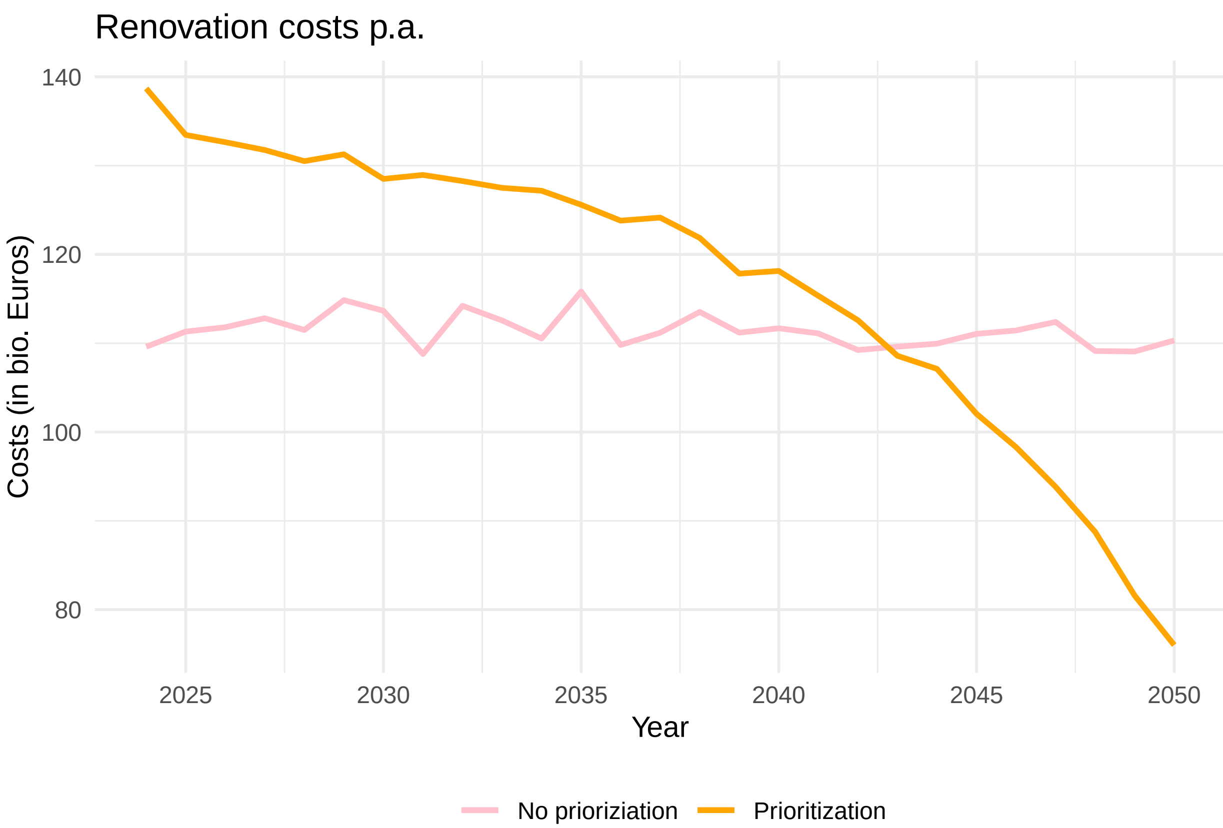
4.1 Costs Relative to GDP

The major visualisation plots expected costs relative to GDP.

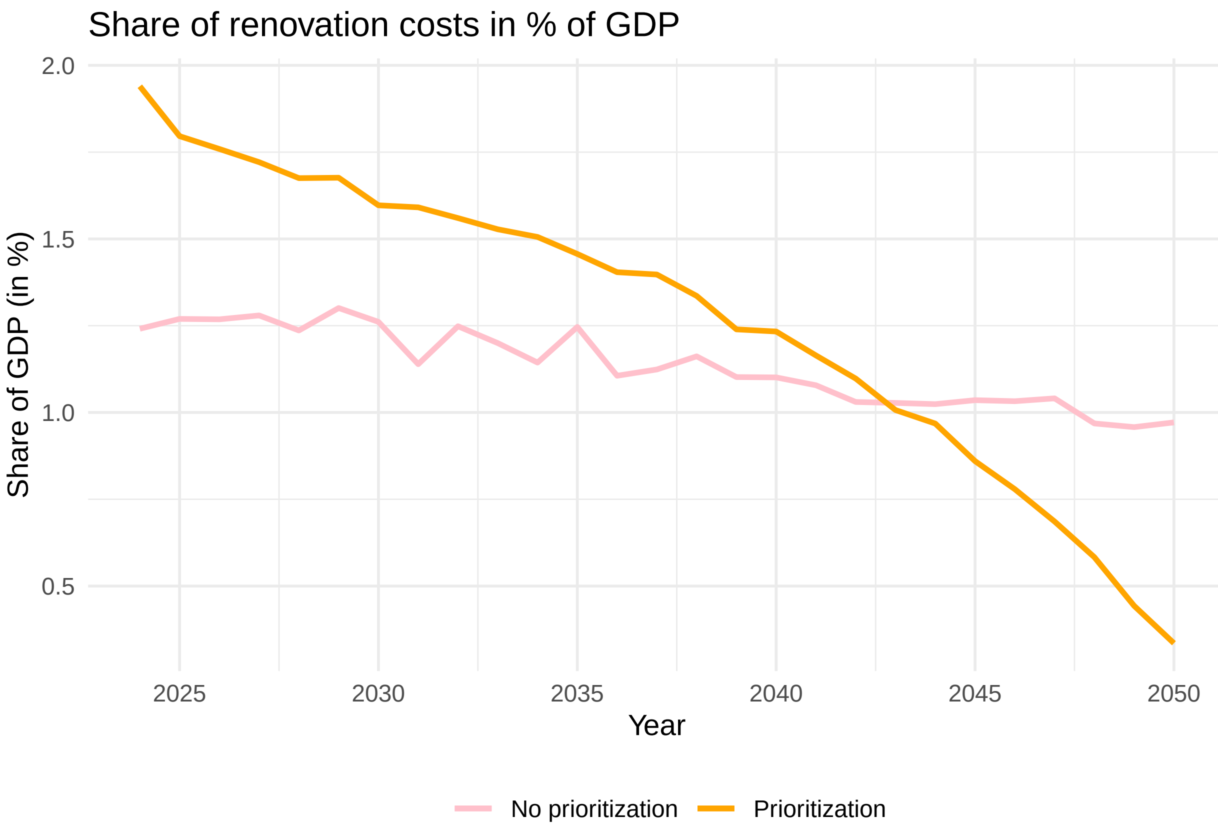
```
year_plot_costs <- ggplot(data = yearly_financing, aes(x = year)) +
  geom_line(aes(y = yearly_cost_rand / 1e9, color = "No prioriziation"),
            linewidth = 1, linetype = "solid") +
  geom_line(aes(y = yearly_cost_prio /1e9, color = "Prioritization"),
            linewidth = 1, linetype = "solid") +
  labs(title = "Renovation costs p.a.",
       x = "Year",
       y = "Costs (in bio. Euros)",
       color = " ") +
  scale_color_manual(values = c("No prioriziation" = "pink",
                                "Prioritization" = "orange")) +
  scale_y_continuous(
    labels = scales::comma,
```

```
    breaks = seq(0, 160, by = 20)
  ) +
  scale_x_continuous(
    breaks = seq(2025, 2055, by = 5)
  )  +
  theme_minimal() +
  theme(legend.position = "bottom",  # Place the legend at the bottom
        legend.box = "horizontal")

year_plot_gdp <- ggplot(data = yearly_financing, aes(x = year)) +
  geom_line (aes(y = share_gdp_rand * 100, color = "No prioritization"),
             linewidth = 1, linetype = "solid") +
  geom_line (aes(y = share_gdp_prio * 100, color = "Prioritization"),
             linewidth = 1, linetype = "solid") +
  labs(title = "Share of renovation costs in % of GDP",
       x = "Year",
       y = "Share of GDP (in %)",
       color = " ") +
  scale_color_manual(values = c("No prioritization" = "pink",
                                "Prioritization" = "orange")) +
  scale_y_continuous(
    labels = scales::comma,
    breaks = seq(0, 3, by = 0.5)
  ) +
  scale_x_continuous(
    breaks = seq(2025, 2055, by = 5)
  )  +
  theme_minimal() +
  theme(legend.position = "bottom",  # Place the legend at the bottom
        legend.box = "horizontal")

year_plot_gdp_de <- ggplot(data = yearly_financing, aes(x = year)) +
  geom_line (aes(y = share_gdp_rand * 100, color = "Ohne Priorisierung"),
             linewidth = 1, linetype = "solid") +
  geom_line (aes(y = share_gdp_prio * 100, color = "Mit Priorisierung"),
             linewidth = 1, linetype = "solid") +
  labs(title = "Anteil der Sanierungskosten am BIP",
       x = "Jahr",
       y = "Anteil am BIP (in %)",
       color = " ") +
  scale_color_manual(values = c("Ohne Priorisierung" = "pink",
                                "Mit Priorisierung" = "orange")) +
  scale_y_continuous(
    labels = scales::comma,
    breaks = seq(0, 3, by = 0.5)
  ) +
  scale_x_continuous(
    breaks = seq(2025, 2055, by = 5)
  )  +
  theme_minimal() +
  theme(legend.position = "bottom",  # Place the legend at the bottom
        legend.box = "horizontal")
```

```
year_plot_costs
```




```
year_plot_gdp
```



Exporting the major graph.

```
ggsave(here("Graphs/plot_gdp.jpeg"), plot = year_plot_gdp, width = 8, height = 6, dpi = 300)
ggsave(here("Graphs/plot_costs.jpeg"), plot = year_plot_costs, width = 8, height = 6, dpi = 300)
ggsave(here("Graphs/plot_gdp_de.jpeg"), plot = year_plot_gdp_de, width = 8, height = 6, dpi = 300)
```

4.2 Supplementary Visualisations

This code provides some basics for plotting the distribution of wealth.

```
visualize <- function(df, df_lorenz){
  coeff1 <- df %>%
    summarise(correlation = wtd.cor(immo_val, nwealth, weight)) %>%
    pull(correlation)

  cor1 <- ggplot(df, aes(x = immo_val, y = nwealth)) +
    geom_point() +
    geom_smooth(method = "lm", se = FALSE) +
    geom_text(x = Inf, y = 0,
            label = paste("Correlation:", round(coeff1[1], 2)),
            hjust = 1, vjust = 0) +
    xlab("Residential wealth") +
    ylab("Net wealth") +
    labs(title="Correlation of residential wealth and net wealth")

  lorenz <- ggplot(df_lorenz, aes(x = perc, y = lorenz, color = variable)) +
  geom_line() +
```

```r
  geom_segment(aes(x = 0, y = 0, xend = 100, yend = 100),
               linetype = "dashed", color = "gray") +
  labs(x = "Cumulative Percentage of Households",
       y = "Cumulative Percentage of Wealth",
       title = "Lorenz Curves",
       color = "Variable")

  return(list(cor_plot = cor1, lorenz = lorenz))
              #, bp = bp_home_in_assets))

}

plots <- visualize(manipulated_data$df, manipulated_data$df_lorenz)
cor_plot <- plots$cor_plot
lorenz_plot <- plots$lorenz
```