Reproducing the visualisations using R programming language

Email: ravanasamuthram_krishnan.pooja@stud.hs-fresenius.de

Name: Pooja Ravanasamuthram Krishnan ID: 400376705

Last compiled on 03 February, 2025

Contents

Abstract **2**

Project Overview **2**

Significance of using R **3**

Initial steps for replication **3**

Learning with R language **21**

Learning about Git and Github **22**

Challenges Encountered **22**

Conclusion and future work **23**

Affidavit **23**

References **25**

Abstract

This paper analyzes the origin and trends of economic sanctions (in international law also referred to as restrictive measures), 1950–2022, drawing from the Global Sanctions Database. We reproduce the graphs in the study using the R programming language (original code provided in Stata). The dataset needed for the graphs is however not provided, which creates a problem for replication. The consequences of sanctions on political goals are central to this paper. The types of graphs are line graph, bar graph, scatter plot, stacked area chart and composite graph showing different trend in data. As observed, despite the overall complexity and variety of visual trends present in the data, all graphs have similarities in the high customisation in order to further make the data easily understandable, the use of the same colour schemes to distinguish different categories, and the concentration of time as a common factor as the X-axis, illustrating trends and shifts over decades. This project demonstrates the utility of R in replicating complex visualizations and offers insights into the process of analyzing economic sanctions data.

Project Overview

The project explores converting raw data into meaningful visualizations through R, a statistical programming language, in terms of how well they reflect findings in the paper. This project is also quite useful in understanding how the methodologies were done in the study while it has practical lessons on the challenges and fine points of data analysis and visualization in R.This project focuses on replicating visualizations from the paper “Economic Sanctions: Evolution, Consequences, and Challenges.” The paper discusses the history, objectives, and effectiveness of economic sanctions, with an emphasis on their evolution over time and their political and economic goals.

Significance of using R

R is widely used in the data science job market and can be run on any operating system, such as Windows, macOS, Linux, and many other environments. It can be integrated with various programming languages, including C, C#, and Python. The number of R users and those interested in learning it is significantly increasing, largely because it is free to use, eliminating the need for financial investment and licensing. As an open-source programming language, R is accessible to anyone. With the growing number of users, many skilled professionals use R as part of their profession, while others learn it out of personal interest.

Initial steps for replication

The capabilities of this language are extended by the Comprehensive R Archive Network (CRAN), which contains numerous libraries and packages, such as ggplot2 and dplyr. These resources can be applied in various fields, including machine learning and bioinformatics, further enhancing the versatility and utility of R. Necessary packages are installed to manipulate the data for analysis and visualization.

```r
options(repos = c(CRAN = "https://cran.rstudio.com/"))


install.packages("dplyr")
```

```
## Installing package into 'C:/Users/pooja/AppData/Local/R/win-library/4.3'
## (as 'lib' is unspecified)

## package 'dplyr' successfully unpacked and MD5 sums checked

## Warning: cannot remove prior installation of package 'dplyr'

## Warning in file.copy(savedcopy, lib, recursive = TRUE): problem copying
## C:\Users\pooja\AppData\Local\R\win-library\4.3\00LOCK\dplyr\libs\x64\dplyr.dll
## to C:\Users\pooja\AppData\Local\R\win-library\4.3\dplyr\libs\x64\dplyr.dll:
## Permission denied

## Warning: restored 'dplyr'

##
## The downloaded binary packages are in
##  C:\Users\pooja\AppData\Local\Temp\RtmpEd9ocY\downloaded_packages
```

```r
install.packages("haven")
```

```
## Installing package into 'C:/Users/pooja/AppData/Local/R/win-library/4.3'
## (as 'lib' is unspecified)

## package 'haven' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\pooja\AppData\Local\Temp\RtmpEd9ocY\downloaded_packages
```

```r
install.packages("ggplot2")
```

```
## Installing package into 'C:/Users/pooja/AppData/Local/R/win-library/4.3'
## (as 'lib' is unspecified)

## package 'ggplot2' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\pooja\AppData\Local\Temp\RtmpEd9ocY\downloaded_packages
```

```r
install.packages("reshape2")
```

```
## Installing package into 'C:/Users/pooja/AppData/Local/R/win-library/4.3'
## (as 'lib' is unspecified)
## package 'reshape2' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\pooja\AppData\Local\Temp\RtmpEd9ocY\downloaded_packages
```

To reproduce the graphs, which are stacked area graph and bar charts, necessary libraries such as 'dplyr', 'tidyr', 'ggplot2', and 'readxl' are loaded. Since the dataset is in Excel format, the 'readxl' library is used as it supports older Excel file formats like '.xls' and '.xlsx'. The Excel file is loaded by mentioning the path of the file, and the sheet = 1 argument is specified to indicate that the first sheet of the Excel file should be loaded. And I have shown how the original dataset looks like. Later, for the purpose of reproducing the graphs, I have manipulated the dataset.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)
library(ggplot2)
library(readxl)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats   1.0.0     v readr     2.1.5
## v lubridate 1.9.3     v stringr   1.5.1
## v purrr     1.0.2     v tibble    3.2.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts
```

```
gsdb_data <- read_excel("C:/Users/pooja/OneDrive/Data Science- Economic Sanctions/GSDB_V


head(gsdb_data)
```

```
## # A tibble: 6 x 16
##   case_id sanctioned_state sanctioning_state begin   end trade descr_trade  arms
##     <dbl> <chr>            <chr>             <dbl> <dbl> <dbl> <chr>       <dbl>
## 1       1 German Democrat~ Germany            1949  1973     0 <NA>            0
## 2       2 Pakistan         India              1949  1951     1 exp_compl,~     0
## 3       3 Bulgaria         United States      1950  1966     0 <NA>            0
## 4       4 Bulgaria         United States      1950  1959     0 <NA>            0
## 5       5 Bulgaria         United States      1950  1963     0 <NA>            0
## 6       6 China            CoCom              1950  1985     1 exp_part        0
## # i 8 more variables: military <dbl>, financial <dbl>, travel <dbl>,
## #   other <dbl>, target_mult <dbl>, sender_mult <dbl>, objective <chr>,
## #   success <chr>
```

Replicating graph 1:

In the dataset, the column 'other' is renamed to 'other_sanct' for clarity and to avoid potential conflicts, as 'other' is too generic. The 'objective' column is divided into four separate columns: ‘objective1’, ‘objective2’, ‘objective3’, and ‘objective4’.This is because the dataset uses commas to store up to four values in a single row. A for loop is used which gives an output value of 1 if a specific objective exists in any of the four split columns and 0 if it does not. Similarly, the ‘success’ column is split into four columns, and the same procedure is applied to the ‘success’ column.

```
gsdb_data <- gsdb_data %>% rename(other_sanct = other)

gsdb_data <- gsdb_data %>%
  separate(objective, into = c("objective1", "objective2", "objective3",
  "objective4"), sep = ",", fill = "right")

objectives <- c("democracy", "destab_regime", "end_war", "human_rights",
                "terrorism", "territorial_conflict", "prevent_war",
                "policy_change", "other")

for (obj in objectives) {
  gsdb_data[[obj]] <- ifelse(gsdb_data$objective1 == obj | gsdb_data$objective2
                             == obj | gsdb_data$objective3 == obj |
                               gsdb_data$objective4 == obj, 1, 0)
}

gsdb_data <- gsdb_data %>%
  separate(success, into = c("success1", "success2", "success3", "success4"),
           sep = ",", fill = "right")

success_types <- c("success_total", "success_part", "nego_settlement", "failed",
                   "ongoing")
```

```r
for (succ in success_types) {
  gsdb_data[[succ]] <- ifelse(gsdb_data$success1 == succ | gsdb_data$success2 ==
                                succ | gsdb_data$success3 == succ |
                                gsdb_data$success4 == succ, 1, 0)
}
```

Two datasets are created: sanct_begin and sanct_end and two columns are selected for each of the dataset. For the sanct_begin dataset, the columns case_id and begin are selected, with the begin column renamed to year. Similarly, for the sanct_end dataset, the columns case_id and end are selected, with the end column renamed to year. Both datasets are then integrated (bound together), duplicates are removed, and only cases from the year 1950 and onward are retained. For each case_id, the begin and end years are marked, and all the years between the begin and end years are filled. For example, if the begin year for a case_id is 1950 and the end year is 1954, it is expanded to include 1950, 1951, 1952, 1953, and 1954. Additional columns, such as objective and success, are incorporated into the expanded dataset using the left_join function. A mutate function is applied to the expanded dataset to add a new column, id_new. This column outputs 1 if the year corresponds to the starting year of a sanction case and 0 otherwise. Finally, for each year, the total number of cases (total_cases) and new cases (new_cases) are calculated, excluding rows with missing values in the id_new column.

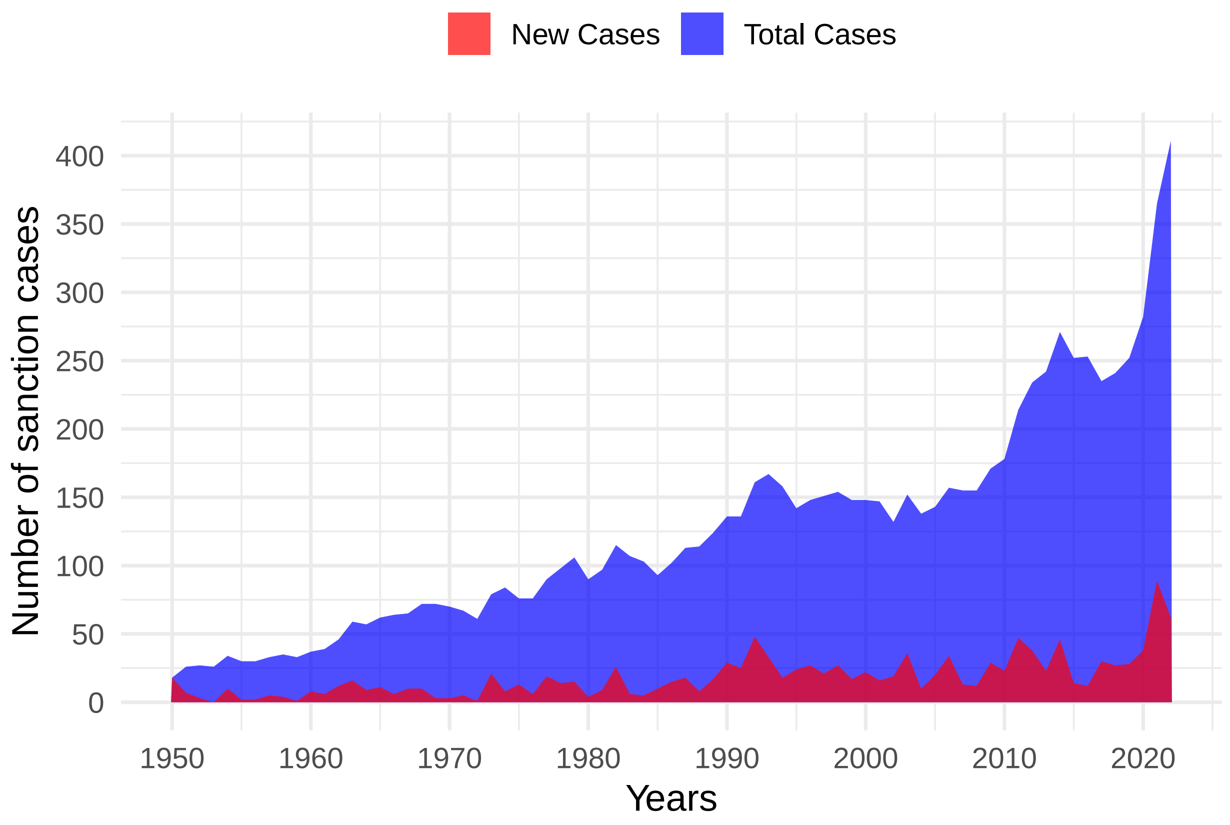
```r
sanct_begin <- gsdb_data %>%
  select(case_id, begin) %>%
  rename(year = begin)


sanct_end <- gsdb_data %>%
  select(case_id, end) %>%
  rename(year = end)

gsdb_case <- bind_rows(sanct_begin, sanct_end) %>%
  distinct() %>%
  arrange(case_id, year) %>%
  filter(year >= 1950)

expanded_data <- gsdb_case %>%
  group_by(case_id) %>%
  complete(year = full_seq(year, 1)) %>%
  left_join(gsdb_data, by = "case_id")

expanded_data <- expanded_data %>%
  filter(year >= 1950)

expanded_data <- expanded_data %>%
  mutate(id_new = ifelse(year %in% sanct_begin$year[sanct_begin$case_id ==
  case_id], 1, 0))
```

```
## Warning: There were 1027 warnings in `mutate()`.
## The first warning was:
## i In argument: `id_new = ifelse(...)`.
## i In group 3: `case_id = 3`.
## Caused by warning in `sanct_begin$case_id == case_id`:
## ! longer object length is not a multiple of shorter object length
## i Run `dplyr::last_dplyr_warnings()` to see the 1026 remaining warnings.
```

```r
expanded_data <- expanded_data %>%
  mutate(id = 1)


collapsed_data <- expanded_data %>%
  group_by(year) %>%
  summarise(total_cases = sum(id, na.rm = TRUE),
            new_cases = sum(id_new, na.rm = TRUE))
```

Finally, using ggplot and ggsave function, the graph is plotted and saved.

```r
ggplot(collapsed_data, aes(x = year)) +
  geom_area(aes(y = total_cases, fill = "Total Cases"), alpha = 0.7) +
  geom_area(aes(y = new_cases, fill = "New Cases"), alpha = 0.7) +
  scale_fill_manual(values = c("Total Cases" = "blue", "New Cases" = "red")) +
  labs(y = "Number of sanction cases", x = "Years") +
  scale_y_continuous(breaks = seq(0, 400, by = 50)) +
  scale_x_continuous(breaks = seq(1950, 2022, by = 10)) +
  theme_minimal(base_size = 14) +
  theme(legend.position = "top", legend.title = element_blank())
```



```r
ggsave("evol_sanct.png", width = 8, height = 6, dpi = 300)
ggsave("figure_1.pdf", width = 8, height = 6)
```

Replicating graph 2:

To replicate the 2nd and 3rd graphs, I used the same procedure of renaming the begin and end columns, integrating them and dedepulicating the redundant data, and considering the case_id for all the years starting from 1950. The new dataset with these changes is assigned to a different dataset named sanction_summary, and it is grouped by year. Then, the summarize function is used to sum the values of all the columns (arms, trade, travel, military, financial, other_sanct) by removing the 'NA' values. The mutate function is then used to create the columns sum2, sum3, sum4, sum5, and sum6.

```r
sanct_begin <- gsdb_data %>%
  select(case_id, begin) %>%
  rename(year = begin)

sanct_end <- gsdb_data %>%
  select(case_id, end) %>%
  rename(year = end)

gsdb_case <- bind_rows(sanct_begin, sanct_end) %>%
  distinct() %>%
  arrange(case_id, year) %>%
  filter(year >= 1950)

expanded_data <- gsdb_case %>%
  group_by(case_id) %>%
  complete(year = full_seq(year, 1)) %>%
  left_join(gsdb_data, by = "case_id") %>%
  filter(year >= 1950)

expanded_data <- expanded_data %>%
  mutate(
    id_new = ifelse(year %in% sanct_begin$year[sanct_begin$case_id == case_id], 1, 0),
    id = 1
  )
```

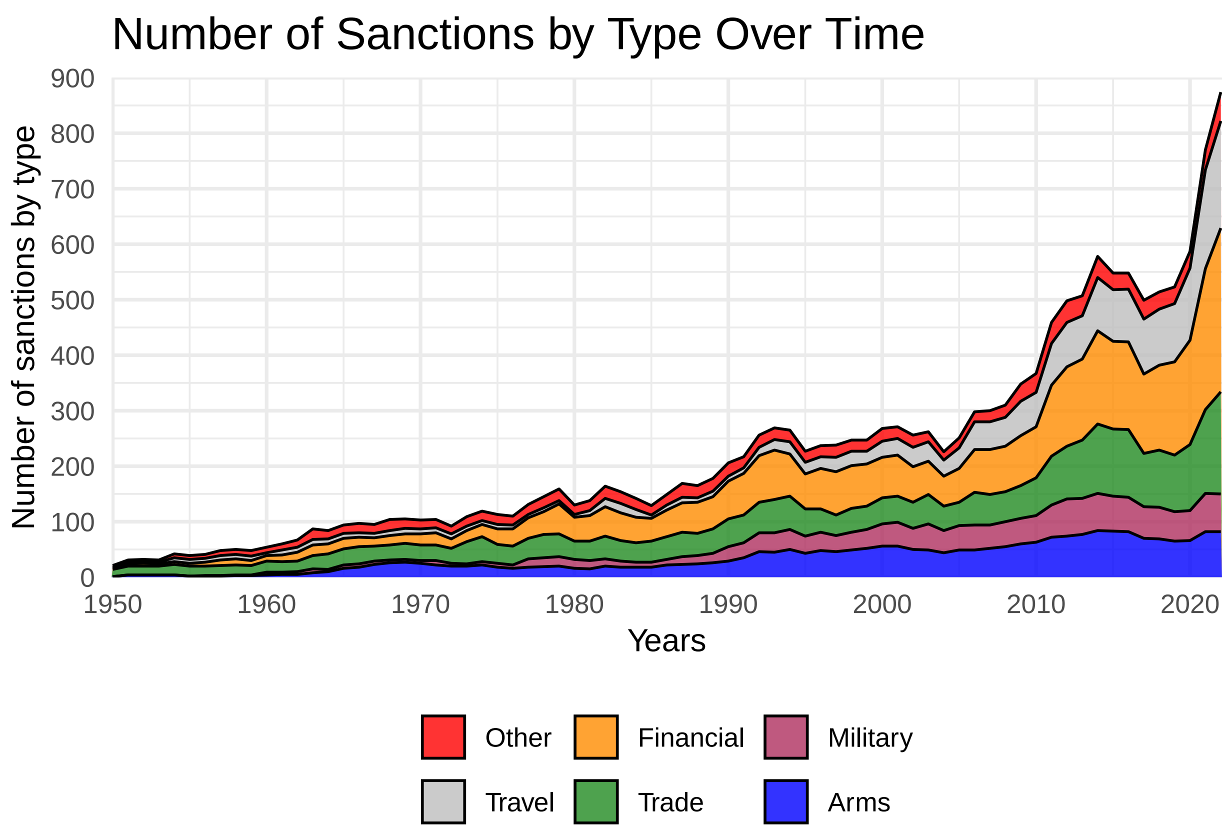
```
## Warning: There were 1027 warnings in `mutate()`.
## The first warning was:
```

```
## i In argument: `id_new = ifelse(...)`.
## i In group 3: `case_id = 3`.
## Caused by warning in `sanct_begin$case_id == case_id`:
## ! longer object length is not a multiple of shorter object length
## i Run `dplyr::last_dplyr_warnings()` to see the 1026 remaining warnings.
```

```r
sanction_summary <- expanded_data %>%
  group_by(year) %>%
  summarise(
    arms = sum(arms, na.rm = TRUE),
    military = sum(military, na.rm = TRUE),
    trade = sum(trade, na.rm = TRUE),
    financial = sum(financial, na.rm = TRUE),
    travel = sum(travel, na.rm = TRUE),
    other_sanct = sum(other_sanct, na.rm = TRUE)
  ) %>%
  mutate(
    sum2 = arms + military,
    sum3 = arms + military + trade,
    sum4 = arms + military + trade + financial,
    sum5 = arms + military + trade + financial + travel,
    sum6 = arms + military + trade + financial + travel + other_sanct
  )
```

The mutate function was used again to create six new columns to properly align the graph for different sanction types. Then, only the columns required for plotting the graph were included using the select function. To group the sanction types under one column and their counts under another, the pivot_longer function was used. For ordering purposes in the graph, the levels and labels were adjusted using mutate. Finally, colors were assigned to each sanction type, and the graph was plotted.

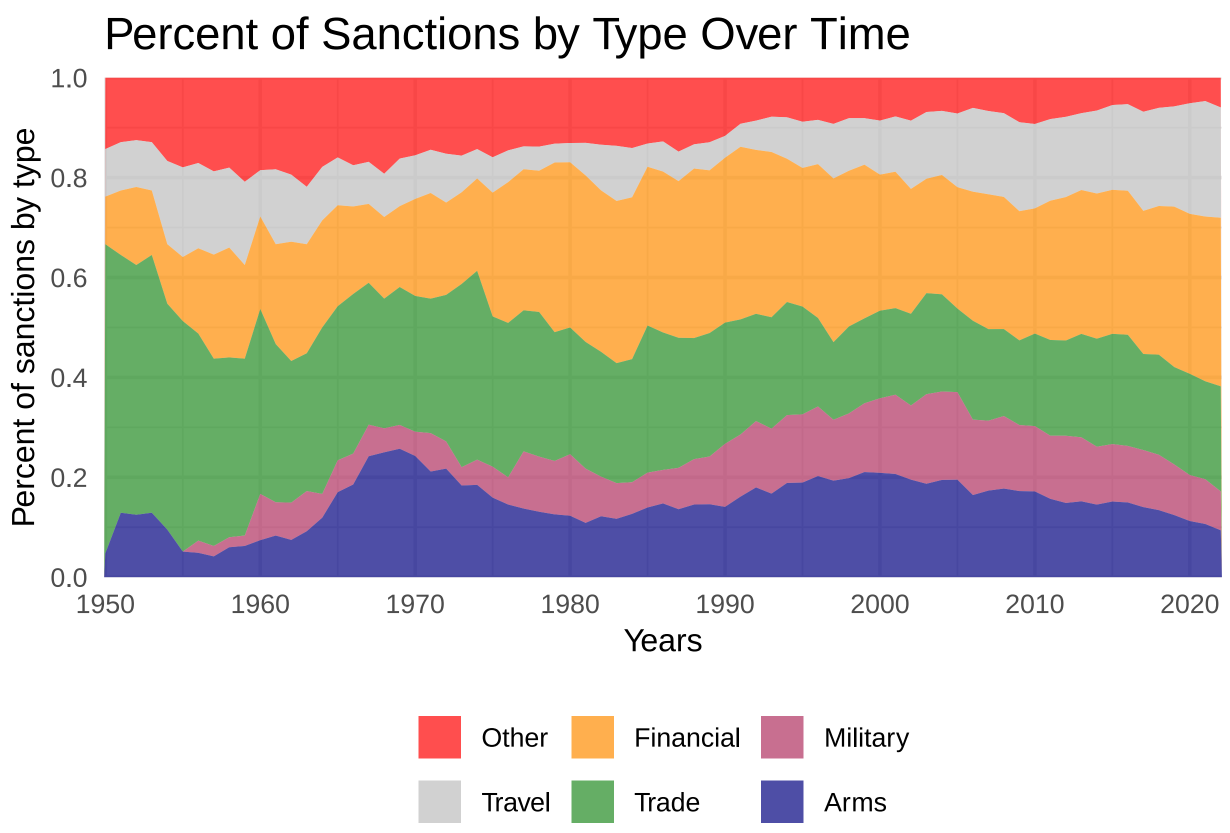
```r
plot_data <- sanction_summary %>%
  mutate(
    inc_arms = arms,
    inc_military = sum2 - arms,
    inc_trade = sum3 - sum2,
    inc_financial = sum4 - sum3,
    inc_travel = sum5 - sum4,
    inc_other = sum6 - sum5
  ) %>%
  select(year, inc_arms, inc_military, inc_trade, inc_financial, inc_travel, inc_other)
  pivot_longer(
    cols = starts_with("inc_"),
    names_to = "category",
    values_to = "value"
  ) %>%
  mutate(
    category = factor(
      category,
      levels = rev(c("inc_arms", "inc_military", "inc_trade", "inc_financial", "inc_trav
      labels = rev(c("Arms", "Military", "Trade", "Financial", "Travel", "Other"))
    )
  )

colors <- c(
  "Arms" = "blue",
  "Military" = "maroon",
  "Trade" = "forestgreen",
  "Financial" = "darkorange",
  "Travel" = "grey",
  "Other" = "red"
)

ggplot(plot_data, aes(x = year, y = value, fill = category)) +
  geom_area(alpha = 0.8, position = "stack", color = "black") +
  scale_fill_manual(values = colors) +
  labs(
    y = "Number of sanctions by type",
    x = "Years",
    title = "Number of Sanctions by Type Over Time"
  ) +
  scale_y_continuous(breaks = seq(0, 900, by = 100), limits = c(0, 900), expand = c(0, 
  scale_x_continuous(breaks = seq(1950, 2022, by = 10), expand = c(0, 0)) +
  theme_minimal(base_size = 14) +
  theme(
    legend.position = "bottom",
    legend.title = element_blank(),
    legend.text = element_text(size = 10),
    axis.text = element_text(size = 10),
    axis.title = element_text(size = 12),
    plot.background = element_rect(fill = "white", color = "white")
  )
```



Replicating graph 3:

For plotting the graph, six columns are created, each column containing the cumulative percentage of sanction types added consecutively. Perc6 is assigned the value 1 because it represents the total of all sanction types divided by itself, resulting in a value of 1. Next, the inc1, inc2, inc3, inc4, inc5, and inc6 columns are created, with each column showing the difference between consecutive perc values calculated in the previous step. Using these values, the graph is plotted and saved.

```
sanction_summary <- sanction_summary %>%
  mutate(
    perc1 = arms / (arms + military + trade + financial + travel + other_sanct),
    perc2 = (arms + military) / (arms + military + trade + financial + travel + other_sa
    perc3 = (arms + military + trade) / (arms + military + trade + financial + travel +
    perc4 = (arms + military + trade + financial) / (arms + military + trade + financial
    perc5 = (arms + military + trade + financial + travel) / (arms + military + trade +
    perc6 = 1,
    inc1 = perc1,
    inc2 = perc2 - perc1,
    inc3 = perc3 - perc2,
    inc4 = perc4 - perc3,
    inc5 = perc5 - perc4,
    inc6 = perc6 - perc5
  )

plot_data <- sanction_summary %>%
  select(year, inc1, inc2, inc3, inc4, inc5, inc6) %>%
  pivot_longer(cols = starts_with("inc"),
               names_to = "category", values_to = "value") %>%
  mutate(

    category = factor(category,
                      levels = rev(c("inc1", "inc2", "inc3", "inc4", "inc5", "inc6")),
                      labels = rev(c("Arms", "Military", "Trade", "Financial", "Travel",
  )

colors <- c("Arms" = "navy",
            "Military" = "maroon",
            "Trade" = "forestgreen",
            "Financial" = "darkorange",
            "Travel" = "grey",
            "Other" = "red")



ggplot(plot_data, aes(x = year, y = value, fill = category)) +
  geom_area(alpha = 0.7) +
  scale_fill_manual(values = colors) +
  labs(
    y = "Percent of sanctions by type",
    x = "Years",
    title = "Percent of Sanctions by Type Over Time"
  ) +
  scale_y_continuous(breaks = seq(0, 1, by = 0.2), limits = c(0, 1), expand = c(0, 0))
  scale_x_continuous(breaks = seq(1950, 2022, by = 10), expand = c(0, 0)) +
  theme_minimal(base_size = 14) +
  theme(
    legend.position = "bottom",
    legend.title = element_blank(),
    legend.text = element_text(size = 10),
    axis.text = element_text(size = 10),
    axis.title = element_text(size = 12),
    plot.background = element_rect(fill = "white", color = "white")
  )
```

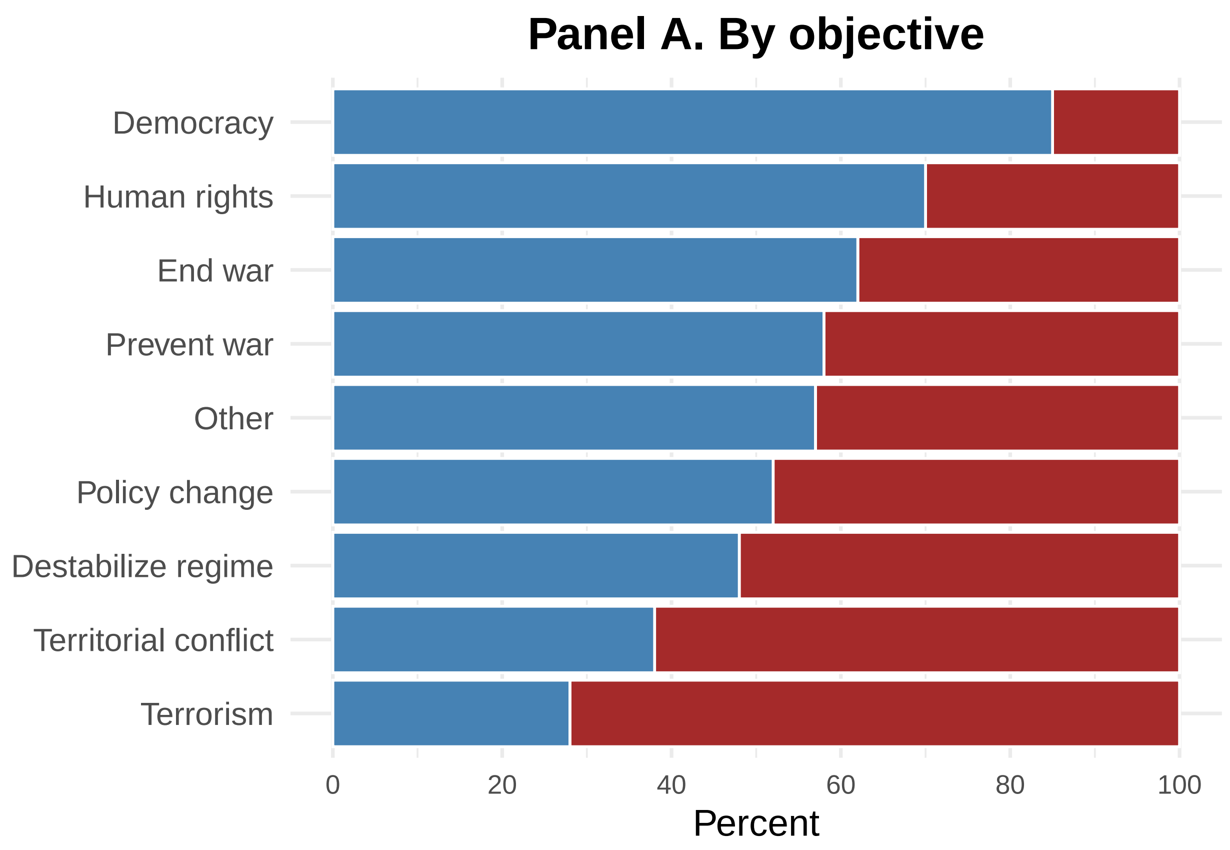


```r
ggsave("evol_type_perc.png", width = 10, height = 6, dpi = 300)
ggsave("figure_2b.pdf", width = 10, height = 6)
```

Replicating graph 4:

For plotting the 4th graph, a new dataset is created with specific columns such as Objective, Category1, and Category2. Category1 and Category2 contain percentage values for the objectives. By default, categorical variables are unordered or ordered alphabetically. If a specific order is required, the factor function can be used. Finally, to plot the graph, the categories are aggregated into a single column using the pivot_longer function.

```r
data <- data.frame(
  Objective = c("Terrorism", "Territorial conflict", "Destabilize regime",
                "Policy change", "Other", "Prevent war", "End war",
                "Human rights", "Democracy"),
  Category1 = c(72, 62, 52, 48, 43, 42, 38, 30, 15),
  Category2 = c(28, 38, 48, 52, 57, 58, 62, 70, 85)
)

data$Objective <- factor(data$Objective,
                         levels = c("Terrorism", "Territorial conflict",
                                    "Destabilize regime", "Policy change",
                                    "Other", "Prevent war", "End war",
                                    "Human rights", "Democracy"))

data_long <- data %>%
  tidyr::pivot_longer(cols = c("Category1", "Category2"),
                      names_to = "Category", values_to = "Percent")

ggplot(data_long, aes(x = Percent, y = Objective, fill = Category)) +
  geom_bar(stat = "identity", position = "stack", color = "white") +
  scale_fill_manual(values = c("Category1" = "brown", "Category2" = "steelblue")) +
  scale_x_continuous(breaks = seq(0, 100, by = 20)) +
  labs(x = "Percent", y = NULL, title = "Panel A. By objective") +
  theme_minimal(base_size = 14) +
  theme(
    legend.position = "none",
    axis.text.y = element_text(size = 12),
    axis.text.x = element_text(size = 10),
    plot.title = element_text(hjust = 0.5, face = "bold")
  )
```



Learning with R language

R is a completely new programming language for me. When I first learned about the course that included R, it sparked my interest. By attending Prof. Huber's lectures and following his notes, I was able to gain a solid understanding of the functions and packages used in R, as well as the way it be manipulated to achieve the desired results. In addition, I used online platforms to learn and deepen my comprehension of various functions. Although I still have a long way to go in this learning process, I can now confidently say that I know how to use the tools required to code in the R programming language.

I recreated four graphs in my report, as the first three are quite similar. I have now learned how to reproduce similar graphs with varied data. The last one is a simple bar graph. Throughout the process of replicating these graphs, I learned more functions and their purposes. In this report, I have made my best effort to explain the code I wrote with the help of ChatGPT and online platforms. The provided code was given in Stata and it was difficut to understand.

Learning about Git and Github

Git and GitHub are widely used in software development projects. They help track changes in the code, facilitate sharing, and enable collaboration with others. This is particularly useful when you have written a piece of code and need someone to review it. Even though I had basic knowledge of this software, I learned more about it for this particular project with the help of my professor's tutorials and the step-by-step procedure that he shared in his Github account. It turned out to be very beneficial, as this software is widely used in professional environments.

I used GitHub to share my progress and my final report, and I also created a pull request with the help of the tutorial my professor shared with me. Specifically, I created a repository to upload my work and forked my professor's repository. Then, I cloned the forked repository to my local machine, made changes to it (to share my work for his review), and pushed the changes back to my forked repository. Finally, I created a pull request from my forked repository to my professor's repository.

Challenges Encountered

I faced difficulties while replicating the graphs, especially the stacked area graphs, as the shaded areas initially got overlapped, and I didn't know how to arrange them accordingly. To resolve this problem, I subtracted the area which overlapped from the total and stack each layer sequentially for proper alignment.

Conclusion and future work

It was a great experience for me working on this paper. By attending classes and clearing exams, I would have learned the syntax and functions of the R language, but by reproducing a graph that was coded in a different programming language, I learned how to interpret the logic between programming languages and troubleshoot the issues.

In the future, I will make efforts to try replicating different types of graphs, learning complex functionalities and algorithms. I will strive to code from scratch, explore advanced techniques, and stay updated with the latest developments in the R environment to enhance my skills further.

Affidavit

I hereby affirm that this submitted paper was authored unaided and solely by me. Additionally, no other sources than those in the reference list were used. Parts of this paper, including tables and figures, that have been taken either verbatim or analogously from other works have in each case been properly cited with regard to their origin and authorship. This paper either in parts or in its entirety, be it in the same or similar form, has not been submitted to any other examination board and has not been published.

I have read the Handbook of Academic Writing by Hildebrandt & Nelke (2019) and have endeavored to comply with the guidelines and standards set forth therein.

I acknowledge that the university may use plagiarism detection software to check my thesis. I agree to cooperate with any investigation of suspected plagiarism and to provide any additional information or evidence requested by the university.

The report includes:

- ☒ About 4000 words (+/- 500).

☒ A title page with personal details (name, email, matriculation number).

☒ An abstract.

☒ A bibliography, created using BibTeX with APA citation style.

☒ The complete R code required to reproduce the results.

☒ Detailed instructions on data acquisition and importation into

R.

☒ An introduction to guide the reader and a conclusion summarizing the work and discussing potential future extensions.

☒ All significant resources used in the report and R code development.

☒ The filled out Affidavit.

☒ A concise description of the successful use of Git and GitHub, as detailed here: - - make_a_pull_request.

☒ A concise description of the presentation published on GitHub.

☒ The project submission includes:

☒ ... The .qmd file(s) of the report.

☒ ... The _quarto.yml file of the report.

☒ ... The .pdf file of the report.

☒ ... The standalone .html file of the report.

☒ ... All necessary files (not available online) to reproduce the report and the R code.

☒ ... The standalone .html file of the presentation.

Pooja Ravanasamuthram Krishnan, 03.02.2025, Cologne, Germany

```r
text <- readLines("Final_report_DS2025.qmd")
```

```
## Warning in readLines("Final_report_DS2025.qmd"): incomplete final line found on
## 'Final_report_DS2025.qmd'
```

```r
text <- text[!grepl("^---", text)]
word_count <- sum(sapply(strsplit(text, "\\s+"), length))
cat(paste("Word count:", word_count))
```

Word count: 3599

References

1. Morgan et al. (2023)

Morgan, T. C., Syropoulos, C., & Yotov, Y. V. (2023). Economic sanctions: Evolution, consequences, and challenges. *Journal of Economic Perspectives*, *37*(1), 3–30. https://doi.org/10.1257/jep.37.1.3