ECON0128 Tutorial 4

Fayyaz

2024-11-13

Introduction

This report describes the process of scraping the Federal Reserve 2016 meeting minutes, specifically extracting the "Staff Review of the Economic Situation" section for each meeting and compiling it into a structured dataset.

Setup and Libraries

The following libraries are required for web scraping and data manipulation:

```
library(tidyverse)  # For data manipulation
library(rvest)      # For web scraping
library(purrr)      # For functional programming
```

Define URLs and Extract Links

We start by defining the base URLs for the Federal Reserve website and extracting links to each meeting's minutes

```
# Define the base URL and main URL for the Federal Reserve 2016 FOMC page
root_url <- "https://www.federalreserve.gov/"
main_url <- paste0(root_url, "monetarypolicy/fomchistorical2016.htm")

# Read the main page and extract all links to meeting minutes
webpage <- read_html(main_url)
rel_paths <- webpage %>%
  html_elements("a") %>%          # Select all anchor tags
  html_attr("href") %>%           # Extract the 'href' attribute (URL) from each <a> tag
  str_subset("minutes") %>%       # Keep only URLs that contain "minutes" (relevant links)
  str_subset("htm")               # Further filter for ".htm" links
```

```
#Define Helper Functions for Data Extraction

#extract_paragraphs() Function
#This function identifies and extracts paragraphs within the "Staff Review of the Economic Situation" s

# Define section heading for locating content in each meeting page
heading_title <- "Staff Review of the Economic Situation"

# Function to extract relevant paragraphs within the targeted section
```

```
extract_paragraphs <- function(page) {
  paragraphs <- page %>%
    html_elements("p") %>%          # Select all paragraph elements
    html_text(trim = TRUE)          # Get the text content and trim whitespace

  # Locate the start of the target section
  start_index <- which(str_detect(paragraphs, heading_title))
  if (length(start_index) == 0) return(NA)  # If not found, return NA

  # Find the end of the section (next empty paragraph or paragraph with <strong>)
  end_index <- start_index + which(str_detect(paragraphs[(start_index + 1):length(paragraphs)], "^\\s*$
  if (is.na(end_index)) end_index <- length(paragraphs)

  # Extract paragraphs in the identified range and clean up the text
  section_text <- paragraphs[start_index:(end_index - 1)] %>%
    str_remove_all("\r|\n|Staff Review of the Economic Situation") %>%
    paste(collapse = " ")

  return(section_text)
}
```

get_meeting_data() Function This function scrapes each meeting's page for the meeting date and the extracted text from the specified section.

```
# Function to get the date and text for each meeting
get_meeting_data <- function(rel_path) {
  page <- read_html(paste0(root_url, rel_path))

  # Extract the meeting date from the meta tag
  date <- page %>%
    html_elements("meta[name='twitter:title']") %>%
    html_attr("content") %>%
    str_remove("FOMC Minutes, ")  # Clean up the date text

  # Extract paragraphs for the "Staff Review of the Economic Situation" section
  text <- extract_paragraphs(page)

  # Return a tibble with the Date and Paragraphs columns
  tibble(Date = date, Paragraphs = text)
}
```

Compile Data for All Meetings

Using the functions above, we iterate over each link to compile the data into a single data frame.

```
# Apply get_meeting_data to each URL in rel_paths to compile all data
minutes_df <- map_df(rel_paths, get_meeting_data)
```

```
# Save the data frame to a CSV file
write_csv(minutes_df, "minutes_df.csv")
```