```r
# Load necessary libraries
library(readxl)
library(ggplot2)
library(ggpmisc)
```

```
## Loading required package: ggpp
```

```
## Registered S3 methods overwritten by 'ggpp':
##   method                  from
##   heightDetails.titleGrob ggplot2
##   widthDetails.titleGrob  ggplot2
```

```
##
## Attaching package: 'ggpp'
```

```
## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```r
library (tidyr)
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
```

```r
library(broom)
library(tseries)   # For Augmented Dickey-Fuller Test
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```r
library(stats)     # For building linear models
library(forecast)
library(tidyverse)
```

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

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ ggpp::annotate() masks ggplot2::annotate()
## ✖ dplyr::filter()  masks stats::filter()
## ✖ dplyr::lag()     masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(zoo) #for as.quartryear function
```

```
##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```r
library (urca)
library(ggpubr)
```

```
##
## Attaching package: 'ggpubr'
##
## The following object is masked from 'package:forecast':
##
##     gghistogram
##
## The following objects are masked from 'package:ggpp':
##
##     as_npc, as_npcx, as_npcy
```

```
library (TSstudio)
library (tvReg)
```

```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
## Funded by the Horizon 2020. Framework Programme of the European Union.
##
##
## Attaching package: 'tvReg'
##
## The following object is masked from 'package:forecast':
##
##     forecast
```

```
library(hpfilter) #for hodrick-prescott filter
library (mFilter)
library (scales)
```

```
##
## Attaching package: 'scales'
##
## The following object is masked from 'package:purrr':
##
##     discard
##
## The following object is masked from 'package:readr':
##
##     col_factor
```

Descriptive statistics:

```
library(writexl)
library(e1071)

# Load the Excel file
file_path <- "C:/Users/user/Documents/Egyetem/Tanítás/2024 ősz/Vállalkozások költségvetési kapcsolatai/Okun model
1.xlsx"
data <- read_excel(file_path)

# Convert the date column to Date format
data$Date <- as.Date(data$Date)
data_without_date <- data %>% select(-Date)

# Calculate summary statistics using summarytools
mydata <- summarytools::descr(data_without_date, stats = c("mean", "sd", "min", "max", "med", "iqr", "n.valid", "
skewness", "kurtosis"))

# Convert the summary statistics to a dataframe
summary_df <- as.data.frame(mydata)

# Add row names as a column
summary_df <- tibble::rownames_to_column(summary_df, "Statistic")

# Save the summary statistics to a new Excel file
#write_xlsx(summary_df, "summary_statistics.xlsx")

print("Summary statistics have been saved to 'summary_statistics.xlsx'.")
```

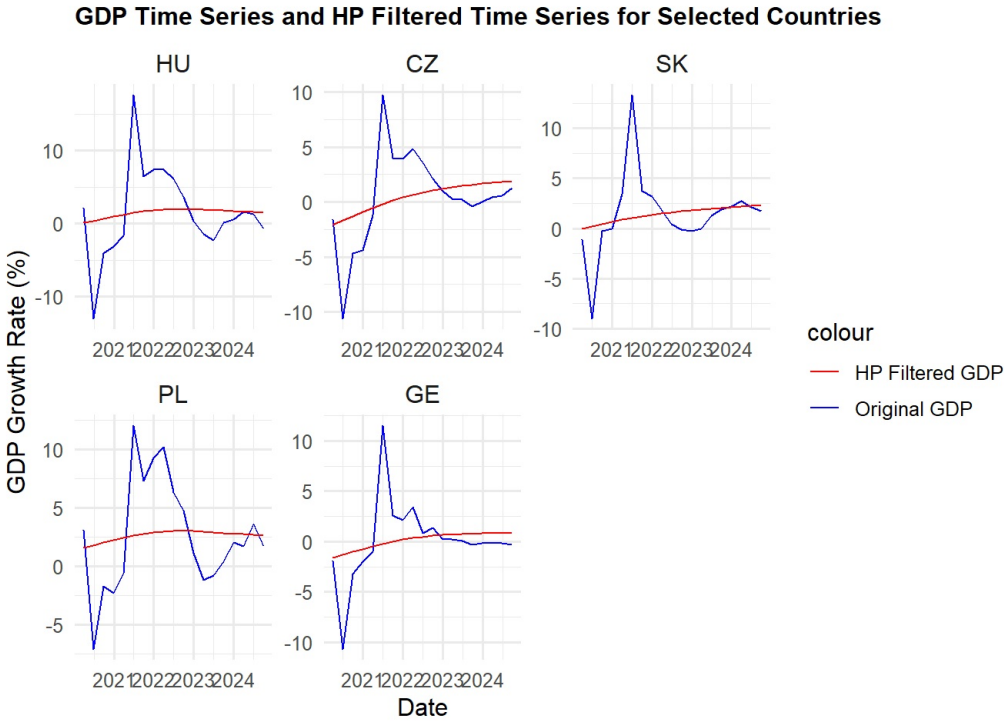
```
## [1] "Summary statistics have been saved to 'summary_statistics.xlsx'."
```

Hodrick-Prescott filters Unemployment

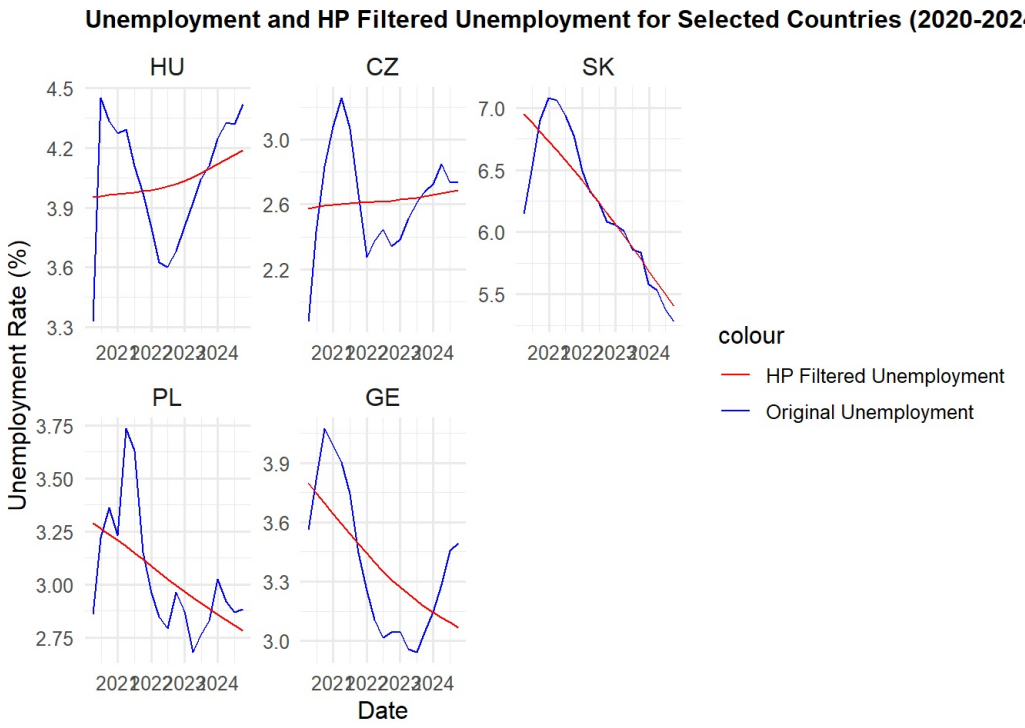
```
# Load the dataset (adjust the path as necessary)
data <- read_excel("C:/Users/user/Documents/Egyetem/Tanítás/2024 ősz/Vállalkozások költségvetési kapcsolatai/Okun
model 1.xlsx")

# Convert the date column to Date format
data$Date <- as.Date(data$Date)
data <- data %>% filter(Date >= as.Date("2020-01-01"))

# Define country codes
countries <- c("HU", "CZ", "SK", "PL", "GE")

# Initialize an empty list to store data frames for each country
hp_data <- list()

# Loop over each country to apply the HP filter to GDP and Unemployment data
for (country in countries) {
  # Dynamically find the column names for GDP growth and unemployment based on country code
  gdp_col <- grep(paste0("^", country, ".*GDP_Growth_rate$"), names(data), value = TRUE)
  unemployment_col <- grep(paste0("^", country, ".*Unemployment"), names(data), value = TRUE)

  # Ensure both columns are found
  if (length(gdp_col) > 0 && length(unemployment_col) > 0) {
    # Extract data for GDP growth and unemployment for the country
    country_data <- data %>%
      select(Date, GDP = all_of(gdp_col), Unemployment = all_of(unemployment_col)) %>%
      # Remove rows with NA values
      na.omit() %>%
      # Add a country column for easy plotting and convert it to a factor
      mutate(Country = factor(country))

    # Apply HP filter to GDP and Unemployment
    hp_gdp <- hpfilter(country_data$GDP, freq = 1600)
    hp_unemployment <- hpfilter(country_data$Unemployment, freq = 1600)

    # Add the filtered series to the data frame
    country_data <- country_data %>%
      mutate(GDP_Trend = hp_gdp$trend,
             GDP_Cycle = hp_gdp$cycle,
             Unemployment_Trend = hp_unemployment$trend,
             Unemployment_Cycle = hp_unemployment$cycle)

    # Add the processed data frame to the list
    hp_data[[country]] <- country_data
  }
}

# Combine data for all countries into one data frame and check structure
hp_data_all <- do.call(rbind, hp_data)

# Ensure combined data is correctly formatted
str(hp_data_all)
```

```
## tibble [95 × 8] (S3: tbl_df/tbl/data.frame)
##  $ Date              : Date[1:95], format: "2020-03-31" "2020-06-30" ...
##  $ GDP               : num [1:95] 2.1 -13 -4.1 -3.1 -1.7 17.6 6.5 7.4 7.4 6.1 ...
##  $ Unemployment      : num [1:95] 3.33 4.45 4.33 4.27 4.29 ...
##  $ Country           : Factor w/ 5 levels "HU","CZ","SK",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ GDP_Trend         : num [1:95, 1] 0.0962 0.387 0.6791 0.9654 1.2357 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : NULL
##  $ GDP_Cycle         : num [1:95] 2 -13.39 -4.78 -4.07 -2.94 ...
##  $ Unemployment_Trend: num [1:95, 1] 3.95 3.96 3.96 3.97 3.97 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : NULL
##  $ Unemployment_Cycle: num [1:95] -0.623 0.492 0.371 0.304 0.321 ...
```

```
# Plot the GDP time series and the HP filtered time series for each country
gdp_plot <- ggplot(hp_data_all, aes(x = Date)) +
  geom_line(aes(y = GDP, color = "Original GDP")) +
  geom_line(aes(y = GDP_Trend, color = "HP Filtered GDP")) +
  facet_wrap(~ Country, scales = "free") +
  labs(title = "GDP Time Series and HP Filtered Time Series for Selected Countries",
       x = "Date", y = "GDP Growth Rate (%)") +
  scale_color_manual(values = c("Original GDP" = "blue", "HP Filtered GDP" = "red")) +
  theme_minimal(base_size = 12) +
  theme(plot.title = element_text(size = 12, face = "bold"),
        strip.text = element_text(size = 12))

# Plot the Unemployment time series and the HP filtered time series for each country
unemployment_plot <- ggplot(hp_data_all, aes(x = Date)) +
  geom_line(aes(y = Unemployment, color = "Original Unemployment")) +
  geom_line(aes(y = Unemployment_Trend, color = "HP Filtered Unemployment")) +
  facet_wrap(~ Country, scales = "free") +
  labs(title = "Unemployment and HP Filtered Unemployment for Selected Countries (2020-2024)",
       x = "Date", y = "Unemployment Rate (%)") +
  scale_color_manual(values = c("Original Unemployment" = "blue", "HP Filtered Unemployment" = "red")) +
  theme_minimal(base_size = 12) +
  theme(plot.title = element_text(size = 12, face = "bold"),
        strip.text = element_text(size = 12))

# Print the plots
print(gdp_plot)
```




```
print(unemployment_plot)
```

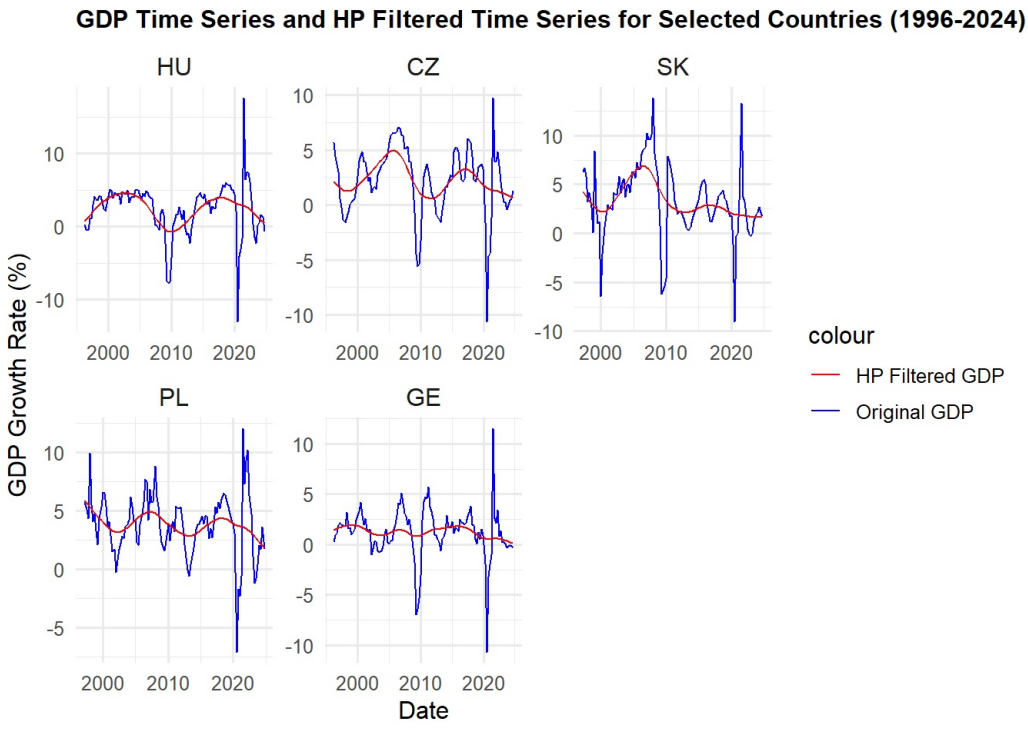


Hodrick-Prescott filters GDP:

```r
# Load the dataset (adjust the path as necessary)
data <- read_excel("C:/Users/user/Documents/Egyetem/Tanítás/2024 ősz/Vállalkozások költségvetési kapcsolatai/Okun model 1.xlsx")

# Convert the date column to Date format
data$Date <- as.Date(data$Date)

#data <- data %>% filter(Date >= as.Date("2010-01-01") & Date <= as.Date("2020-01-01"))
# Define country codes
countries <- c("HU", "CZ", "SK", "PL", "GE")

# Initialize an empty list to store data frames for each country
hp_data <- list()

# Loop over each country to apply the HP filter to GDP and Unemployment data
for (country in countries) {
  # Dynamically find the column names for GDP growth and unemployment based on country code
  gdp_col <- grep(paste0("^", country, ".*GDP_Growth_rate$"), names(data), value = TRUE)
  unemployment_col <- grep(paste0("^", country, ".*Unemployment"), names(data), value = TRUE)

  # Ensure both columns are found
  if (length(gdp_col) > 0 && length(unemployment_col) > 0) {
    # Extract data for GDP growth and unemployment for the country
    country_data <- data %>%
      select(Date, GDP = all_of(gdp_col), Unemployment = all_of(unemployment_col)) %>%
      # Remove rows with NA values
      na.omit() %>%
      # Add a country column for easy plotting and convert it to a factor
      mutate(Country = factor(country))

    # Apply HP filter to GDP and Unemployment
    hp_gdp <- hpfilter(country_data$GDP, freq = 1600)
    hp_unemployment <- hpfilter(country_data$Unemployment, freq = 1600)

    # Add the filtered series to the data frame
    country_data <- country_data %>%
      mutate(GDP_Trend = hp_gdp$trend,
             GDP_Cycle = hp_gdp$cycle,
             Unemployment_Trend = hp_unemployment$trend,
             Unemployment_Cycle = hp_unemployment$cycle)

    # Add the processed data frame to the list
    hp_data[[country]] <- country_data
  }
}

# Combine data for all countries into one data frame and check structure
hp_data_all <- do.call(rbind, hp_data)

# Ensure combined data is correctly formatted
str(hp_data_all)
```

```
## tibble [567 × 8] (S3: tbl_df/tbl/data.frame)
##  $ Date              : Date[1:567], format: "1996-03-31" "1996-06-30" ...
##  $ GDP               : num [1:567] 0.2 -0.4 -0.5 1.1 1.1 3 4.1 3.7 3.6 4.1 ...
##  $ Unemployment      : num [1:567] 10.15 9.96 9.76 9.45 9.15 ...
##  $ Country           : Factor w/ 5 levels "HU","CZ","SK",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ GDP_Trend         : num [1:567, 1] 0.751 1.019 1.286 1.552 1.814 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : NULL
##  $ GDP_Cycle         : num [1:567] -0.551 -1.419 -1.786 -0.452 -0.714 ...
##  $ Unemployment_Trend: num [1:567, 1] 10.16 9.93 9.7 9.48 9.25 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : NULL
##  $ Unemployment_Cycle: num [1:567] -0.00343 0.0302 0.05466 -0.02708 -0.10111 ...
```

```
# Plot the GDP time series and the HP filtered time series for each country
ggplot(hp_data_all, aes(x = Date)) +
  geom_line(aes(y = GDP, color = "Original GDP")) +
  geom_line(aes(y = GDP_Trend, color = "HP Filtered GDP")) +
  facet_wrap(~ Country, scales = "free") +
  labs(title = "GDP Time Series and HP Filtered Time Series for Selected Countries (1996-2024)",
       x = "Date", y = "GDP Growth Rate (%)") +
  scale_color_manual(values = c("Original GDP" = "blue", "HP Filtered GDP" = "red")) +
  theme_minimal(base_size = 12) +
  theme(plot.title = element_text(size = 12, face = "bold"),
        strip.text = element_text(size = 12))
```



Time varying regression coefficient of Okun’s law for V4 countries

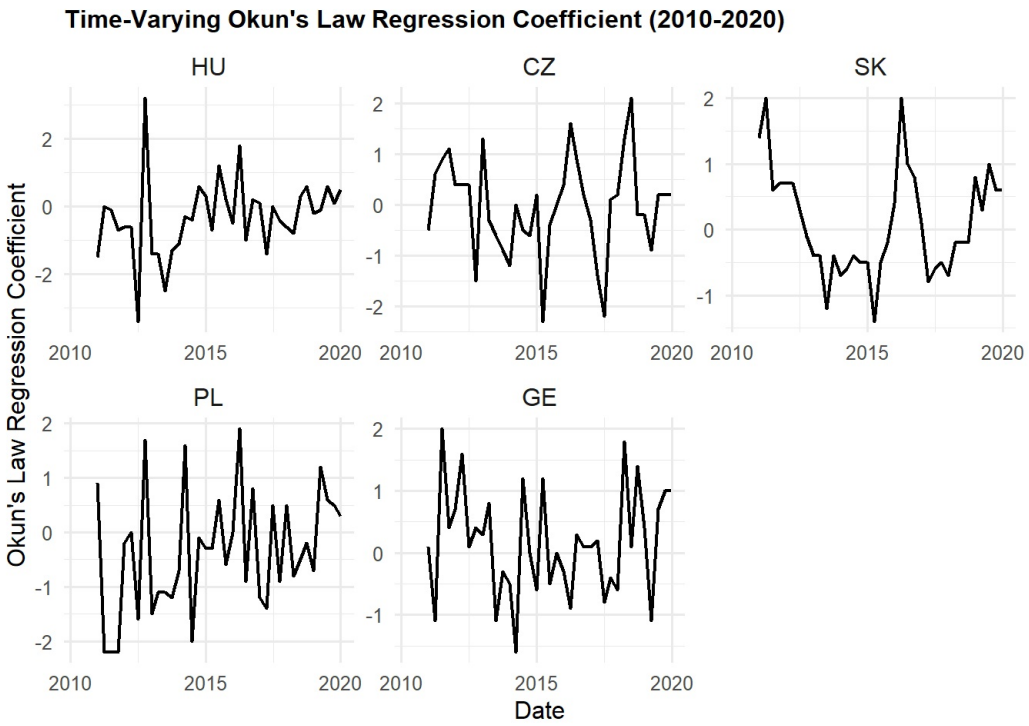
```r
# Load the dataset (adjust the path as necessary)
data <- read_excel("C:/Users/user/Documents/Egyetem/Tanítás/2024 ősz/Vállalkozások költségvetési kapcsolatai/Okun model 1.xlsx")

# Convert the date column to Date format
data$Date <- as.Date(data$Date)
data <- data %>% filter(Date >= as.Date("2010-01-01") & Date <= as.Date("2020-01-01"))

# Define country codes
countries <- c("HU", "CZ", "SK", "PL", "GE")

# Initialize an empty list to store data frames for each country
okun_data <- list()

# Loop over each country to extract unemployment and GDP growth data
for (country in countries) {
  # Dynamically find the column names for GDP growth and unemployment based on country code
  gdp_col <- grep(paste0("^", country, ".*GDP_Growth_rate$"), names(data), value = TRUE)
  unemployment_col <- grep(paste0("^", country, ".*Unemployment"), names(data), value = TRUE)

  # Ensure both columns are found
  if (length(gdp_col) > 0 && length(unemployment_col) > 0) {
    # Extract data for GDP growth and unemployment for the country
    country_data <- data %>%
      select(Date, GDP = all_of(gdp_col), Unemployment = all_of(unemployment_col)) %>%
      # Remove rows with NA values
      na.omit() %>%
      # Add a country column for easy plotting and convert it to a factor
      mutate(Country = factor(country))

    # Convert all columns to the expected types
    country_data$GDP <- as.numeric(country_data$GDP)
    country_data$Unemployment <- as.numeric(country_data$Unemployment)
    country_data$Country <- as.factor(country_data$Country)

    # Add the processed data frame to the list
    okun_data[[country]] <- country_data
  }
}

# Combine data for all countries into one data frame and check structure
okun_data_all <- do.call(rbind, okun_data)

# Ensure combined data is correctly formatted
str(okun_data_all)
```

```
## tibble [200 × 4] (S3: tbl_df/tbl/data.frame)
##  $ Date        : Date[1:200], format: "2010-03-31" "2010-06-30" ...
##  $ GDP         : num [1:200] 0.1 0.8 1.6 1.6 2.7 1.7 1 2.3 -0.3 -1.1 ...
##  $ Unemployment: num [1:200] 10.7 10.9 10.7 10.7 10.8 ...
##  $ Country     : Factor w/ 5 levels "HU","CZ","SK",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```r
# Function to calculate time-varying Okun's law regression coefficient
calculate_time_varying_coefficient <- function(df) {
  df <- df %>% arrange(Date)
  window_size <- 4  # Define the window size (e.g., 12 months)
  df$Coefficient <- rollapply(df, width = window_size, FUN = function(x) {
    model <- lm(GDP ~ Unemployment, data = as.data.frame(x))
    coef(model)[2]
  }, by.column = FALSE, align = "right", fill = NA)
  return(df)
}

# Apply the function to each country's data
okun_data_all <- okun_data_all %>% group_by(Country) %>% do(calculate_time_varying_coefficient(.))

# Plot the time-varying Okun's law regression coefficient for each country
ggplot(okun_data_all, aes(x = Date, y = Coefficient)) +
  geom_line(size = 0.8) +
  facet_wrap(~ Country, scales = "free") +
  labs(title = "Time-Varying Okun's Law Regression Coefficient (2010-2020)",
       x = "Date", y = "Okun's Law Regression Coefficient") +
  theme_minimal(base_size = 12) +
  theme(plot.title = element_text(size = 12, face = "bold"),
        strip.text = element_text(size = 12))
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## ℹ Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## Warning: Removed 3 rows containing missing values or values outside the scale range
## (`geom_line()`).
```



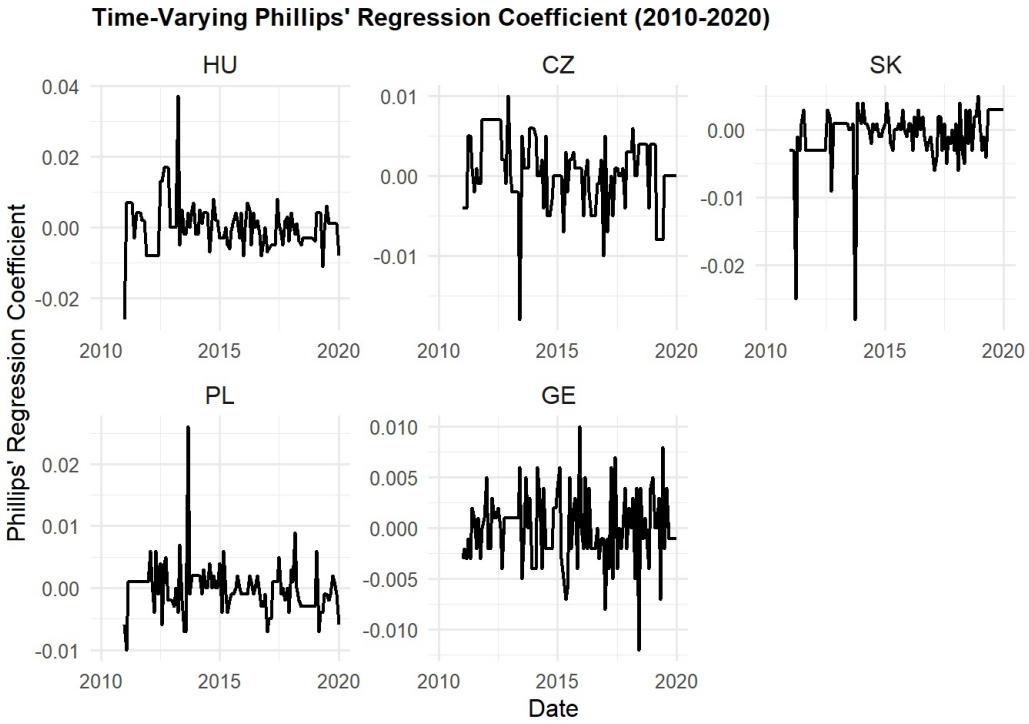
Time Varying regression coefficient of the Phillips Curve for V4 countries

```r
# Load the dataset (adjust the path as necessary)
data <- read_excel("C:/Users/user/Documents/Egyetem/Tanítás/2024 ősz/Vállalkozások költségvetési kapcsolatai/Infl
ation model 3.xlsx")

# Convert the date column to Date format
data$Date <- as.Date(data$Date)
data <- data %>% filter(Date >= as.Date("2010-01-01") & Date <= as.Date("2020-01-01"))

# Define country codes
countries <- c("HU", "CZ", "SK", "PL", "GE")

# Initialize an empty list to store data frames for each country
phillips_data <- list()

# Loop over each country to extract unemployment and inflation data
for (country in countries) {
  # Dynamically find the column names for inflation and unemployment based on country code
  inflation_col <- grep(paste0("^", country, ".*Inflation$"), names(data), value = TRUE)
  unemployment_col <- grep(paste0("^", country, ".*Unemployment"), names(data), value = TRUE)

  # Ensure both columns are found
  if (length(inflation_col) > 0 && length(unemployment_col) > 0) {
    # Extract data for inflation and unemployment for the country
    country_data <- data %>%
      select(Date, Inflation = all_of(inflation_col), Unemployment = all_of(unemployment_col)) %>%
      # Remove rows with NA values
      na.omit() %>%
      # Add a country column for easy plotting and convert it to a factor
      mutate(Country = factor(country))

    # Convert all columns to the expected types
    country_data$Inflation <- as.numeric(country_data$Inflation)
    country_data$Unemployment <- as.numeric(country_data$Unemployment)
    country_data$Country <- as.factor(country_data$Country)

    # Add the processed data frame to the list
    phillips_data[[country]] <- country_data
  }
}

# Combine data for all countries into one data frame and check structure
phillips_data_all <- do.call(rbind, phillips_data)

# Ensure combined data is correctly formatted
str(phillips_data_all)
```

```
## tibble [600 × 4] (S3: tbl_df/tbl/data.frame)
##  $ Date        : Date[1:600], format: "2010-01-31" "2010-02-28" ...
##  $ Inflation   : num [1:600] 0.062 0.056 0.058 0.057 0.049 0.05 0.036 0.036 0.037 0.043 ...
##  $ Unemployment: num [1:600] 0.106 0.108 0.109 0.108 0.109 ...
##  $ Country     : Factor w/ 5 levels "HU","CZ","SK",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```r
# Function to calculate time-varying Phillips' regression coefficient
calculate_time_varying_coefficient <- function(df) {
  df <- df %>% arrange(Date)
  window_size <- 12  # Define the window size (e.g., 12 months)
  df$Coefficient <- rollapply(df, width = window_size, FUN = function(x) {
    model <- lm(Inflation ~ Unemployment, data = as.data.frame(x))
    coef(model)[2]
  }, by.column = FALSE, align = "right", fill = NA)
  return(df)
}

# Apply the function to each country's data
phillips_data_all <- phillips_data_all %>% group_by(Country) %>% do(calculate_time_varying_coefficient(.))

# Plot the time-varying Phillips' regression coefficient for each country
ggplot(phillips_data_all, aes(x = Date, y = Coefficient)) +
  geom_line(size = 0.8) +
  facet_wrap(~ Country, scales = "free") +
  labs(title = "Time-Varying Phillips' Regression Coefficient (2010-2020)",
       x = "Date", y = "Phillips' Regression Coefficient") +
  theme_minimal(base_size = 12) +
  theme(plot.title = element_text(size = 12, face = "bold"),
        strip.text = element_text(size = 12))
```

```
## Warning: Removed 11 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

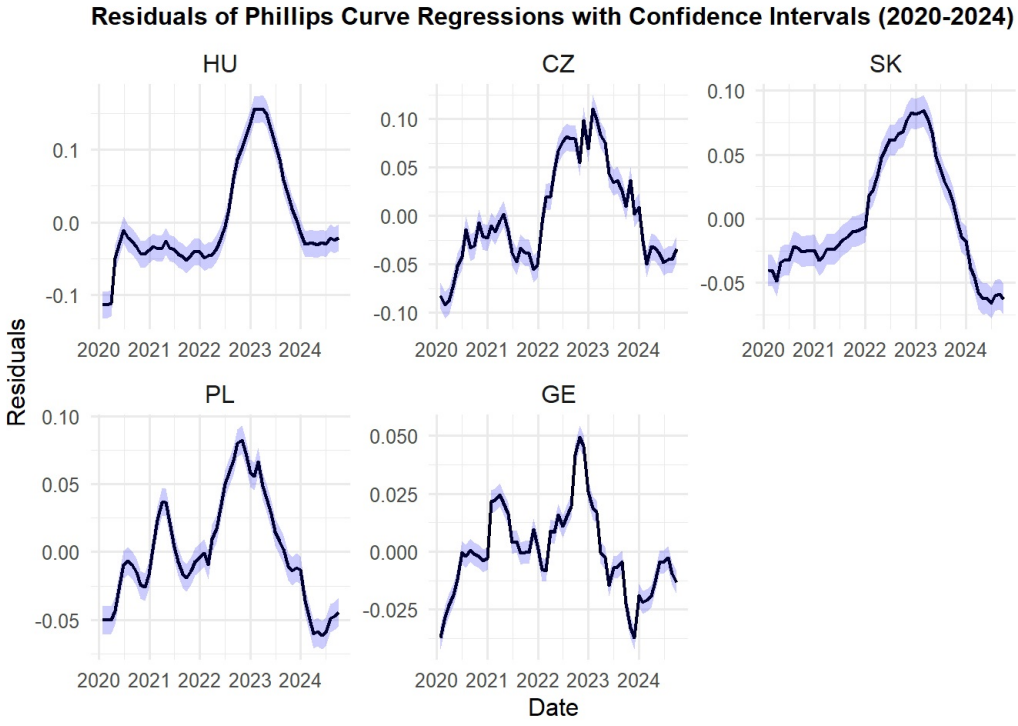


Residuals of Phillips Curve

```
# Load the dataset (adjust the path as necessary)
data <- read_excel("C:/Users/user/Documents/Egyetem/Tanítás/2024 ősz/Vállalkozások költségvetési kapcsolatai/Infl
ation model 3.xlsx")

# Convert the date column to Date format
data$Date <- as.Date(data$Date)
data <- data %>% filter(Date >= as.Date("2020-01-01"))

# Define country codes
countries <- c("HU", "CZ", "SK", "PL", "GE")

# Initialize an empty list to store data frames for each country
phillips_data <- list()

# Loop over each country to extract unemployment and inflation data
for (country in countries) {
  # Dynamically find the column names for inflation and unemployment based on country code
  inflation_col <- grep(paste0("^", country, ".*Inflation$"), names(data), value = TRUE)
  unemployment_col <- grep(paste0("^", country, ".*Unemployment"), names(data), value = TRUE)

  # Ensure both columns are found
  if (length(inflation_col) > 0 && length(unemployment_col) > 0) {
    # Extract data for inflation and unemployment for the country
    country_data <- data %>%
      select(Date, Inflation = all_of(inflation_col), Unemployment = all_of(unemployment_col)) %>%
      # Remove rows with NA values
      na.omit() %>%
      # Add a country column for easy plotting and convert it to a factor
      mutate(Country = factor(country))

    # Convert all columns to the expected types
    country_data$Inflation <- as.numeric(country_data$Inflation)
    country_data$Unemployment <- as.numeric(country_data$Unemployment)
    country_data$Country <- as.factor(country_data$Country)

    # Add the processed data frame to the list
    phillips_data[[country]] <- country_data
  }
}

# Combine data for all countries into one data frame and check structure
phillips_data_all <- do.call(rbind, phillips_data)

# Ensure combined data is correctly formatted
str(phillips_data_all)
```

```
## tibble [285 × 4] (S3: tbl_df/tbl/data.frame)
##  $ Date        : Date[1:285], format: "2020-01-31" "2020-02-29" ...
##  $ Inflation   : num [1:285] 0.047 0.044 0.039 0.025 0.022 0.029 0.039 0.04 0.034 0.03 ...
##  $ Unemployment: num [1:285] 0.0328 0.0331 0.034 0.0424 0.045 ...
##  $ Country     : Factor w/ 5 levels "HU","CZ","SK",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
# Function to calculate residuals and confidence intervals
calculate_residuals_and_ci <- function(df) {
  model <- lm(Inflation ~ Unemployment, data = df)
  df$Residuals <- residuals(model)
  ci <- 1.96 * sd(df$Residuals) / sqrt(nrow(df))
  df$CI_lower <- df$Residuals - ci
  df$CI_upper <- df$Residuals + ci
  return(df)
}

# Apply the function to each country's data
phillips_data_all <- phillips_data_all %>% group_by(Country) %>% do(calculate_residuals_and_ci(.))

# Plot the residuals against the dates for each country in a time series format
ggplot(phillips_data_all, aes(x = Date, y = Residuals)) +
  geom_line(size = 0.8) +
  geom_ribbon(aes(ymin = CI_lower, ymax = CI_upper), alpha = 0.2, fill = "blue") +
  facet_wrap(~ Country, scales = "free") +
  labs(title = "Residuals of Phillips Curve Regressions with Confidence Intervals (2020-2024)",
       x = "Date", y = "Residuals") +
  theme_minimal(base_size = 12) +
  theme(plot.title = element_text(size = 12, face = "bold"),
        strip.text = element_text(size = 12))
```

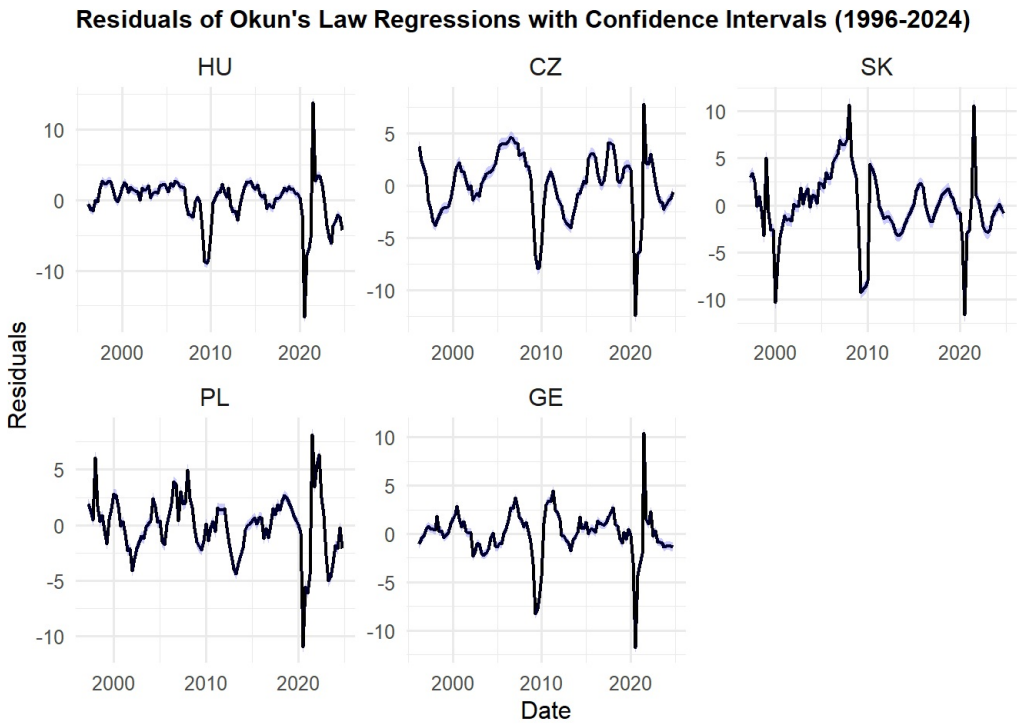


Residuals of Okun's Law

```r
data <- read_excel("C:/Users/user/Documents/Egyetem/Tanítás/2024 ősz/Vállalkozások költségvetési kapcsolatai/Okun model 1.xlsx")

# Convert the date column to Date format
data$Date <- as.Date(data$Date)
#data <- data %>% filter(Date >= as.Date("2020-01-01"))

# Define country codes
countries <- c("HU", "CZ", "SK", "PL", "GE")

# Initialize an empty list to store data frames for each country
okun_data <- list()

# Loop over each country to extract unemployment and inflation data
for (country in countries) {
  # Dynamically find the column names for inflation and unemployment based on country code
  gdp_col <- grep(paste0("^", country, ".*GDP_Growth_rate$"), names(data), value = TRUE)
  unemployment_col <- grep(paste0("^", country, ".*Unemployment"), names(data), value = TRUE)

  # Ensure both columns are found
  if (length(gdp_col) > 0 && length(unemployment_col) > 0) {
    # Extract data for inflation and unemployment for the country
    country_data <- data %>%
      select(Date, GDP = all_of(gdp_col), Unemployment = all_of(unemployment_col)) %>%
      # Remove rows with NA values
      na.omit() %>%
      # Add a country column for easy plotting and convert it to a factor
      mutate(Country = factor(country))

    # Convert all columns to the expected types
    country_data$GDP <- as.numeric(country_data$GDP)
    country_data$Unemployment <- as.numeric(country_data$Unemployment)
    country_data$Country <- as.factor(country_data$Country)

    # Add the processed data frame to the list
    okun_data[[country]] <- country_data
  }
}

# Combine data for all countries into one data frame and check structure
okun_data_all <- do.call(rbind, okun_data)

# Ensure combined data is correctly formatted
str(okun_data_all)
```

```
## tibble [567 × 4] (S3: tbl_df/tbl/data.frame)
##  $ Date        : Date[1:567], format: "1996-03-31" "1996-06-30" ...
##  $ GDP         : num [1:567] 0.2 -0.4 -0.5 1.1 1.1 3 4.1 3.7 3.6 4.1 ...
##  $ Unemployment: num [1:567] 10.15 9.96 9.76 9.45 9.15 ...
##  $ Country     : Factor w/ 5 levels "HU","CZ","SK",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```r
# Function to calculate residuals and confidence intervals
calculate_residuals_and_ci <- function(df) {
  model <- lm(GDP ~ Unemployment, data = df)
  df$Residuals <- residuals(model)
  ci <- 1.96 * sd(df$Residuals) / sqrt(nrow(df))
  df$CI_lower <- df$Residuals - ci
  df$CI_upper <- df$Residuals + ci
  return(df)
}

# Apply the function to each country's data
okun_data_all <- okun_data_all %>% group_by(Country) %>% do(calculate_residuals_and_ci(.))

# Plot the residuals against the dates for each country in a time series format
ggplot(okun_data_all, aes(x = Date, y = Residuals)) +
  geom_line(size = 0.8) +
  geom_ribbon(aes(ymin = CI_lower, ymax = CI_upper), alpha = 0.2, fill = "blue") +
  facet_wrap(~ Country, scales = "free") +
  labs(title = "Residuals of Okun's Law Regressions with Confidence Intervals (1996-2024)",
       x = "Date", y = "Residuals") +
  theme_minimal(base_size = 12) +
  theme(plot.title = element_text(size = 12, face = "bold"),
        strip.text = element_text(size = 12))
```



Okun's Law

```
data <- read_excel("C:/Users/user/Documents/Egyetem/Tanítás/2024 ősz/Vállalkozások költségvetési kapcsolatai/Okun
model 1.xlsx")

# Convert the date column to Date format
data$Date <- as.Date(data$Date)
data <- data %>% filter(Date >= as.Date("2020-01-01")) #& Date <= as.Date("2020-01-01"))
# Define country codes
countries <- c("HU", "CZ", "SK", "PL", "GE")

# Initialize an empty list to store data frames for each country
okun_data <- list()

# Loop over each country to extract unemployment and inflation data
for (country in countries) {
  # Dynamically find the column names for inflation and unemployment based on country code
  gdp_col <- grep(paste0("^", country, ".*GDP_Growth_rate$"), names(data), value = TRUE)
  unemployment_col <- grep(paste0("^", country, ".*Unemployment"), names(data), value = TRUE)

  # Ensure both columns are found
  if (length(gdp_col) > 0 && length(unemployment_col) > 0) {
    # Extract data for inflation and unemployment for the country
    country_data <- data %>%
      select(Date, GDP = all_of(gdp_col), Unemployment = all_of(unemployment_col)) %>%
      # Remove rows with NA values
      na.omit() %>%
      # Add a country column for easy plotting and convert it to a factor
      mutate(Country = factor(country))

    # Convert all columns to the expected types
    country_data$GDP <- as.numeric(country_data$GDP)
    country_data$Unemployment <- as.numeric(country_data$Unemployment)
    country_data$Country <- as.factor(country_data$Country)

    # Add the processed data frame to the list
    okun_data[[country]] <- country_data
  }
}

# Combine data for all countries into one data frame and check structure
okun_data_all <- do.call(rbind, okun_data)

# Ensure combined data is correctly formatted
str(okun_data_all)
```

```
## tibble [95 × 4] (S3: tbl_df/tbl/data.frame)
##  $ Date        : Date[1:95], format: "2020-03-31" "2020-06-30" ...
##  $ GDP         : num [1:95] 2.1 -13 -4.1 -3.1 -1.7 17.6 6.5 7.4 7.4 6.1 ...
##  $ Unemployment: num [1:95] 3.33 4.45 4.33 4.27 4.29 ...
##  $ Country     : Factor w/ 5 levels "HU","CZ","SK",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
# Plot the Phillips curve for each country with regression equation
ggplot(okun_data_all, aes(x = Unemployment, y = GDP)) +
  geom_point(size = 0.8) +
  geom_smooth(method = "lm", color = "blue") +
  facet_wrap(~ Country, scales = "free") +
  labs(title = "Okun's Law for selected countries (2020-2024)",
       x = "Unemployment Rate (%)", y = "GDP Growth rate (%)") +
  theme_minimal(base_size = 12) +
  theme(plot.title = element_text(size = 12, face = "bold"),
        strip.text = element_text(size = 12)) +
  stat_poly_eq(aes(label = paste(..eq.label.., ..rr.label.., sep = "~~~")),
               formula = y ~ x, parse = TRUE, label.x.npc = "left", label.y.npc = "top", size = 2.4, color="red")
```

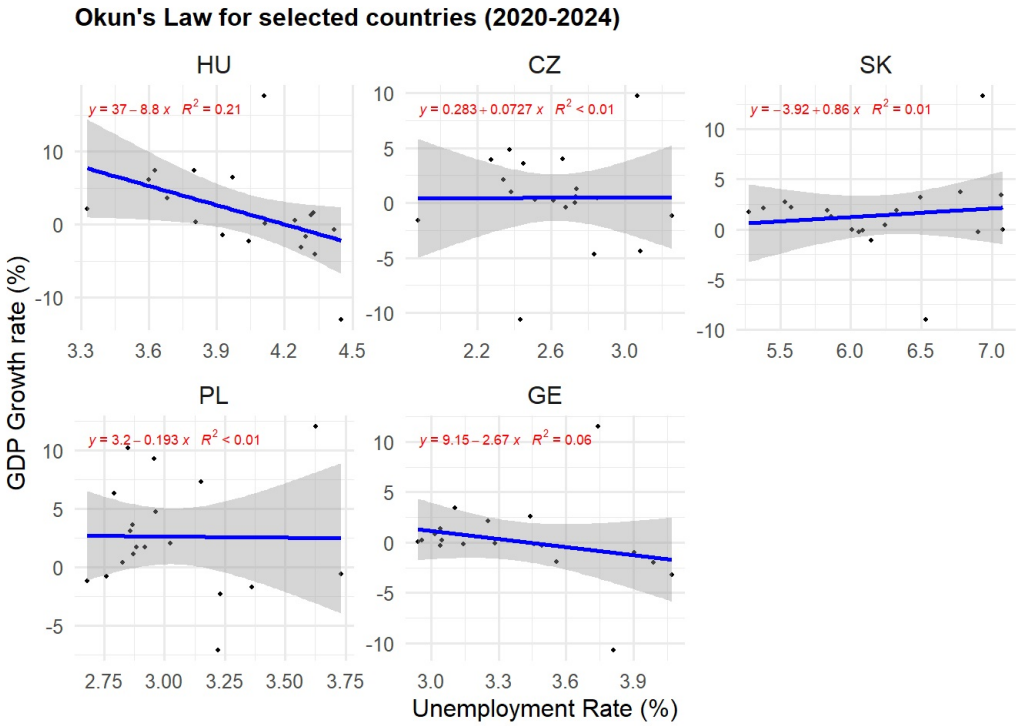
```
## Warning in stat_poly_eq(aes(label = paste(..eq.label.., ..rr.label.., sep =
## "~~~")), : Ignoring unknown parameters: `label.x.npc` and `label.y.npc`
```

```
## Warning: The dot-dot notation (`..eq.label..`) was deprecated in ggplot2 3.4.0.
## ℹ Please use `after_stat(eq.label)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

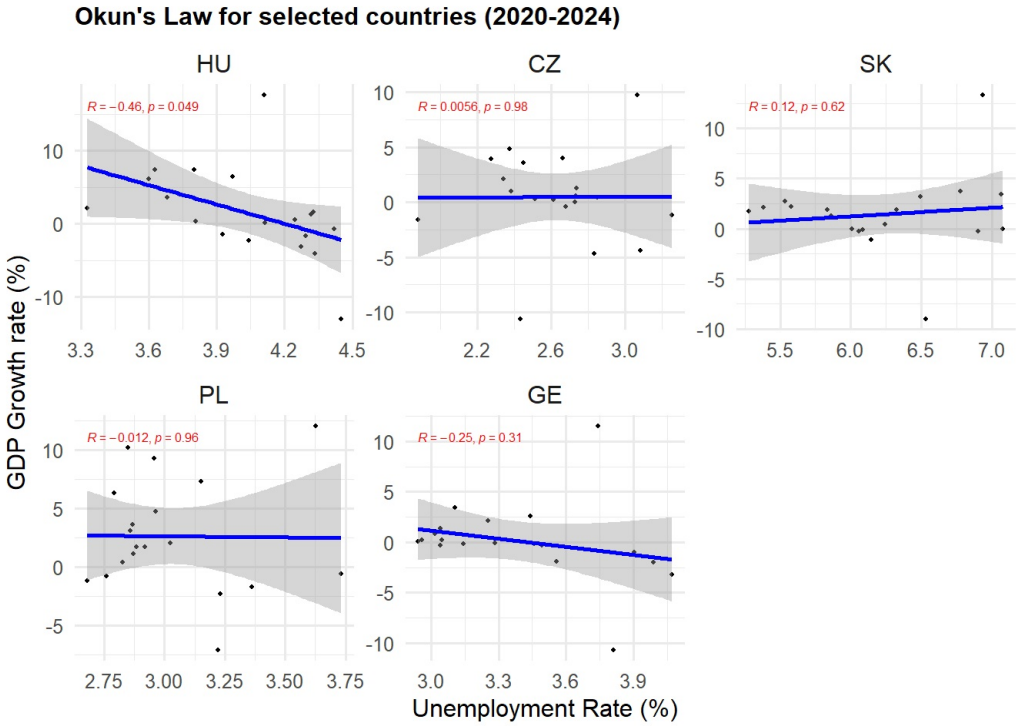
```
## Warning in ci_f_ncp(stat, df1 = df1, df2 = df2, probs = probs): Upper limit
## outside search range. Set to the maximum of the parameter range.
```

```
## Warning in compute_group(...): CI computation error: Error in check_output(cint, probs = probs, parameter_rang
e = c(0, 1)): out[1] <= out[2] is not TRUE
```



```
# Plot the Phillips curve for each country
ggplot(okun_data_all, aes(x = Unemployment, y = GDP)) +
  geom_point(size = 0.8) + stat_cor( size = 2.1, color = "red")+
  geom_smooth(method = "lm", color = "blue") +
  facet_wrap(~ Country, scales = "free") +
  labs(title = "Okun's Law for selected countries (2020-2024) ",
       x = "Unemployment Rate (%)", y = "GDP Growth rate (%)") +
  theme_minimal(base_size = 12) +
  theme(plot.title = element_text(size = 12, face = "bold"),
        strip.text = element_text(size = 12))
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



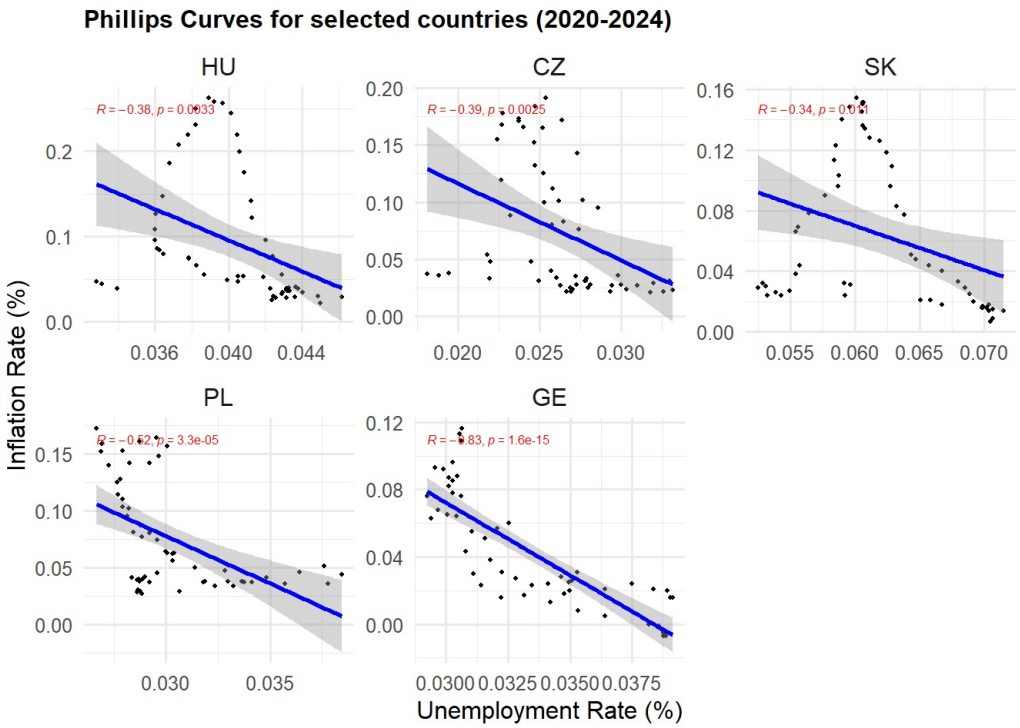
Phillips Curve

```r
# Load the dataset (adjust the path as necessary)
data <- read_excel("C:/Users/user/Documents/Egyetem/Tanítás/2024 ősz/Vállalkozások költségvetési kapcsolatai/Infl
ation model 3.xlsx")

# Convert the date column to Date format
data$Date <- as.Date(data$Date)
data <- data %>% filter(Date >= as.Date("2020-01-01"))
# Define country codes
countries <- c("HU", "CZ", "SK", "PL", "GE")

# Initialize an empty list to store data frames for each country
phillips_data <- list()

# Loop over each country to extract unemployment and inflation data
for (country in countries) {
  # Dynamically find the column names for inflation and unemployment based on country code
  inflation_col <- grep(paste0("^", country, ".*Inflation$"), names(data), value = TRUE)
  unemployment_col <- grep(paste0("^", country, ".*Unemployment"), names(data), value = TRUE)

  # Ensure both columns are found
  if (length(inflation_col) > 0 && length(unemployment_col) > 0) {
    # Extract data for inflation and unemployment for the country
    country_data <- data %>%
      select(Date, Inflation = all_of(inflation_col), Unemployment = all_of(unemployment_col)) %>%
      # Remove rows with NA values
      na.omit() %>%
      # Add a country column for easy plotting and convert it to a factor
      mutate(Country = factor(country))

    # Convert all columns to the expected types
    country_data$Inflation <- as.numeric(country_data$Inflation)
    country_data$Unemployment <- as.numeric(country_data$Unemployment)
    country_data$Country <- as.factor(country_data$Country)

    # Add the processed data frame to the list
    phillips_data[[country]] <- country_data
  }
}

# Combine data for all countries into one data frame and check structure
phillips_data_all <- do.call(rbind, phillips_data)

# Ensure combined data is correctly formatted
str(phillips_data_all)
```

```
## tibble [285 × 4] (S3: tbl_df/tbl/data.frame)
##  $ Date        : Date[1:285], format: "2020-01-31" "2020-02-29" ...
##  $ Inflation   : num [1:285] 0.047 0.044 0.039 0.025 0.022 0.029 0.039 0.04 0.034 0.03 ...
##  $ Unemployment: num [1:285] 0.0328 0.0331 0.034 0.0424 0.045 ...
##  $ Country     : Factor w/ 5 levels "HU","CZ","SK",..: 1 1 1 1 1 1 1 1 1 1 ...
```

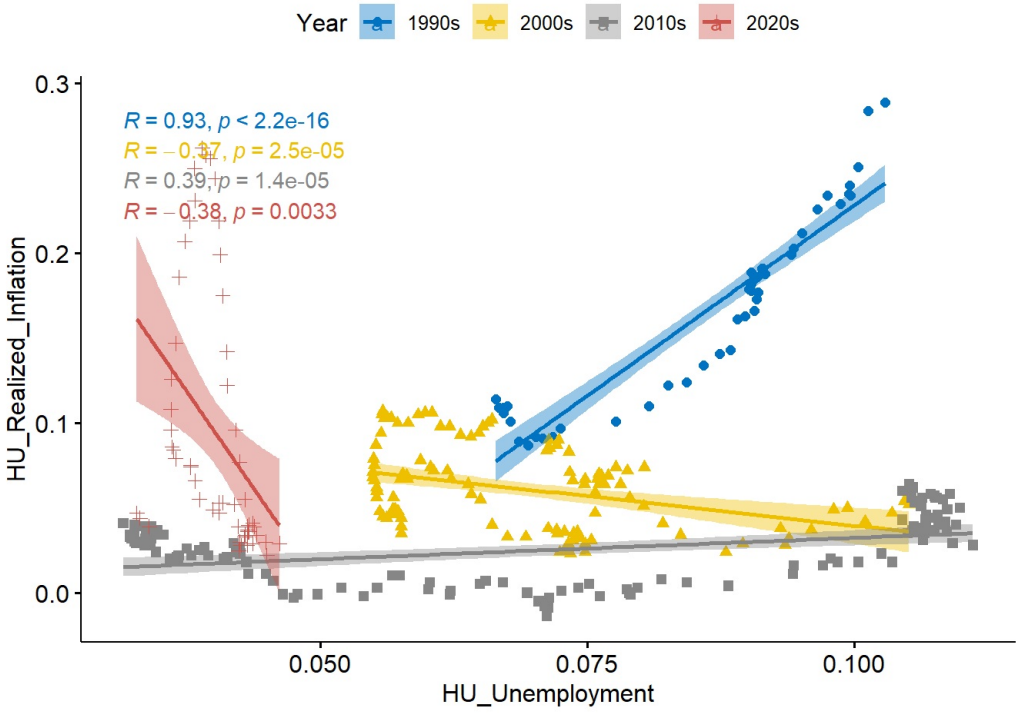
```r
# Plot the Phillips curve for each country
ggplot(phillips_data_all, aes(x = Unemployment, y = Inflation)) +
  geom_point(size = 0.8) + stat_cor( size = 2.1, color = "red")+
  geom_smooth(method = "lm", color = "blue") +
  facet_wrap(~ Country, scales = "free") +
  labs(title = "Phillips Curves for selected countries (2020-2024) ",
       x = "Unemployment Rate (%)", y = "Inflation Rate (%)") +
  theme_minimal(base_size = 12) +
  theme(plot.title = element_text(size = 12, face = "bold"),
        strip.text = element_text(size = 12))
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



Phillips Curve by decade in V4 countries

```
file_path <-
"C:/Users/user/Documents/Egyetem/Tanítás/2024 ősz/Vállalkozások költségvetési kapcsolatai/Inflation model 3.xlsx"
# Replace with the actual file path
df <- read_excel(file_path)
df <- df %>% arrange(Date)

df <- df %>% mutate(Year = paste0(10*as.numeric(substr(year(Date),1, 3)), "s"))

Curve_1 <- ggscatter(df, x= "HU_Unemployment", y = "HU_Realized_Inflation", add = "reg.line", color = "Year", pal
ette = "jco", shape = "Year", conf.int = TRUE ) +stat_cor(aes(color =Year))

Curve_1
```

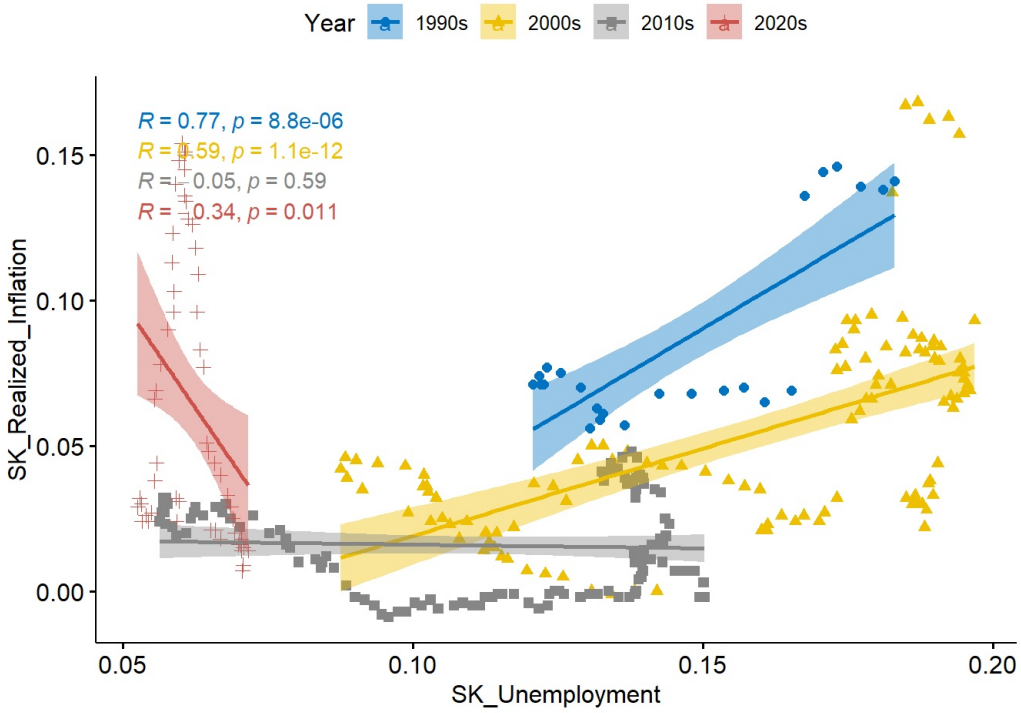



```
Curve_2 <- ggscatter(df, x= "SK_Unemployment", y = "SK_Realized_Inflation", add = "reg.line", color = "Year", pal
ette = "jco", shape = "Year", conf.int = TRUE ) +stat_cor(aes(color =Year))

Curve_2
```

```
## Warning: Removed 24 rows containing non-finite outside the scale range
## (`stat_smooth()`).
```

```
## Warning: Removed 24 rows containing non-finite outside the scale range
## (`stat_cor()`).
```

```
## Warning: Removed 24 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

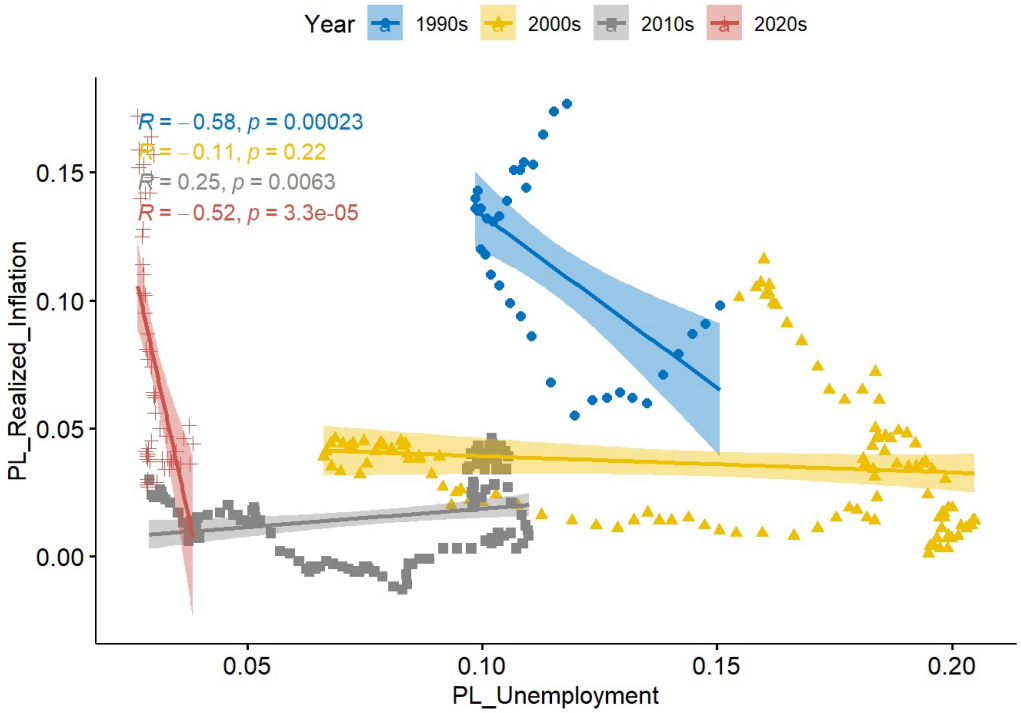



```
Curve_3 <- ggscatter(df, x= "PL_Unemployment", y = "PL_Realized_Inflation", add = "reg.line", color = "Year", pal
ette = "jco", shape = "Year", conf.int = TRUE ) +stat_cor(aes(color =Year))

Curve_3
```

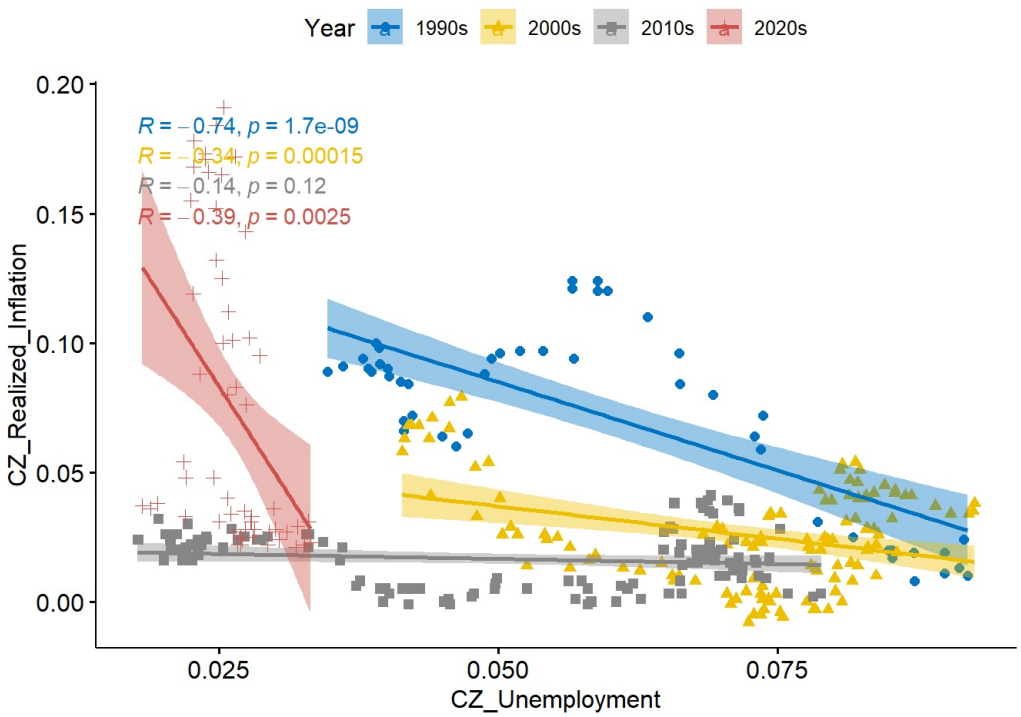
```
## Warning: Removed 12 rows containing non-finite outside the scale range
## (`stat_smooth()`).
```

```
## Warning: Removed 12 rows containing non-finite outside the scale range
## (`stat_cor()`).
```

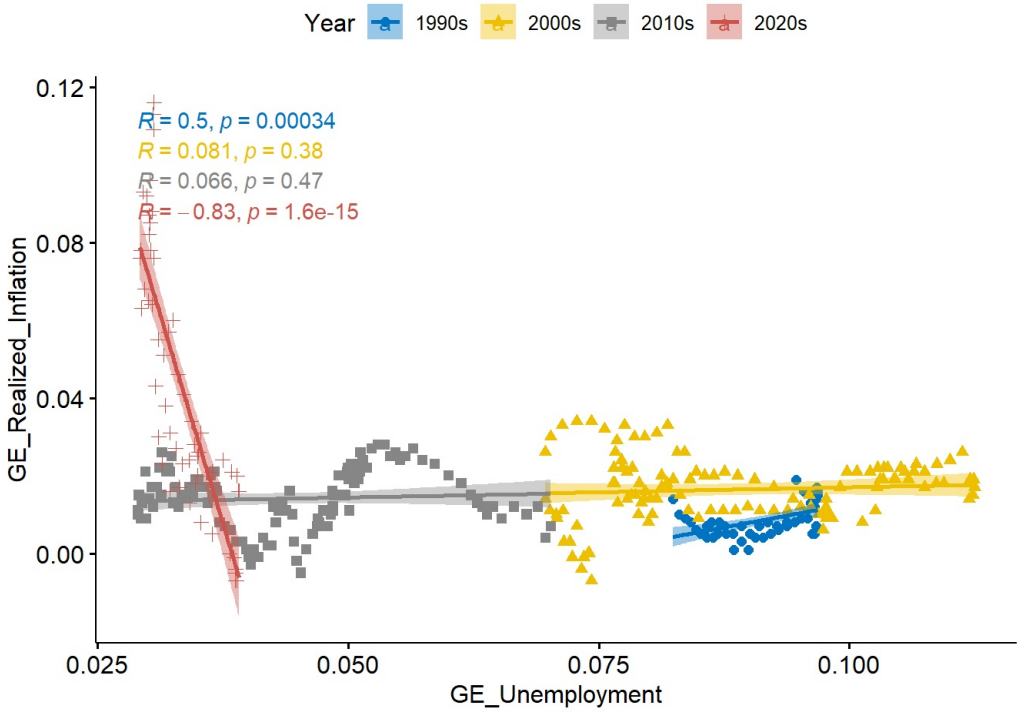
```
## Warning: Removed 12 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



```
Curve_4 <- ggscatter(df, x= "CZ_Unemployment", y = "CZ_Realized_Inflation", add = "reg.line", color = "Year", pal
ette = "jco", shape = "Year", conf.int = TRUE ) +stat_cor(aes(color =Year))

Curve_4
```




```
Curve_5 <- ggscatter(df, x= "GE_Unemployment", y = "GE_Realized_Inflation", add = "reg.line", color = "Year", pal
ette = "jco", shape = "Year", conf.int = TRUE ) +stat_cor(aes(color =Year))

Curve_5
```

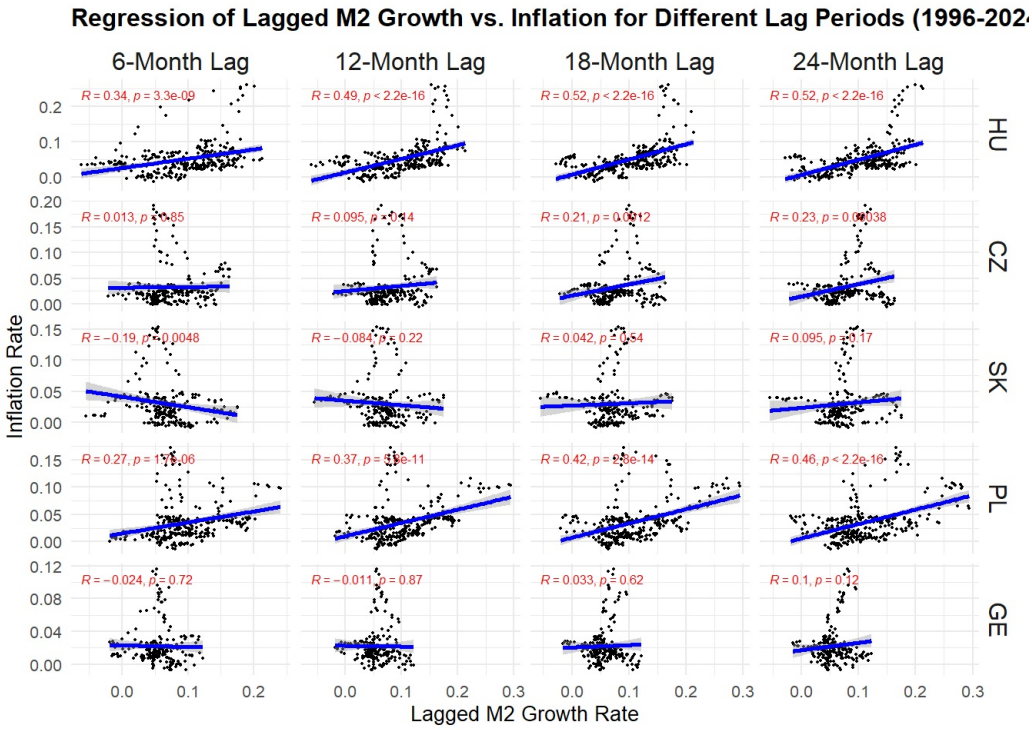


Monetary curve

```
# Load the dataset (adjust the path as necessary)
data <- read_excel("C:/Users/user/Documents/Egyetem/Tanítás/2024 ősz/Vállalkozások költségvetési kapcsolatai/Infl
ation model 3.xlsx")

# Convert the date column to Date format
data$Date <- as.Date(data$Date)

# Truncate the data from January 1, 2020
#data <- data %>% filter(Date >= as.Date("2020-01-01")) # & Date <= as.Date("2020-01-01"))
```

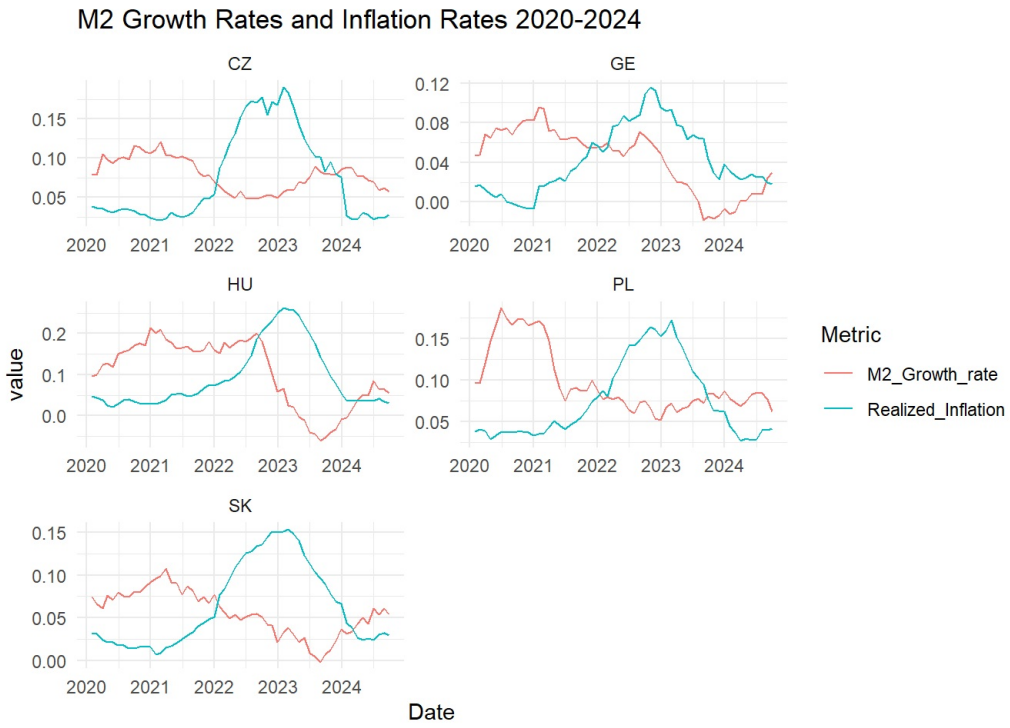
```r
# Define country codes
countries <- c("HU", "CZ", "SK", "PL", "GE")

# Initialize an empty list to store data frames for each country and lag
regression_data <- list()
r_squared_values <- data.frame(Country = character(), Lag = character(), R2 = numeric())

# Loop over each country to calculate 6, 12, and 18-month lagged M2 growth for regression
for (country in countries) {
  # Dynamically find the column names for inflation and M2 growth based on country code
  inflation_col <- grep(paste0("^", country, ".*Inflation$"), names(data), value = TRUE)
  m2_growth_col <- grep(paste0("^", country, ".*M2_Growth_rate$"), names(data), value = TRUE)

  # Ensure both columns are found
  if (length(inflation_col) > 0 && length(m2_growth_col) > 0) {
    # Extract data for inflation and M2 growth for the country
    country_data <- data %>%
      select(Date, Inflation = all_of(inflation_col), M2_Growth = all_of(m2_growth_col)) %>%
      # Calculate 6, 12, and 18-month lags for M2 growth
      mutate(M2_Growth_Lag6 = lag(M2_Growth, 6),
             M2_Growth_Lag12 = lag(M2_Growth, 12),
             M2_Growth_Lag18 = lag(M2_Growth, 18),
             M2_Growth_Lag24 = lag(M2_Growth, 24)) %>%
      # Remove rows with NA values due to lagging
      na.omit() %>%
      # Add a country column for easy plotting
      mutate(Country = country)

    # Calculate R-squared for each lag and store in r_squared_values data frame
    for (lag in c("M2_Growth_Lag6", "M2_Growth_Lag12", "M2_Growth_Lag18", "M2_Growth_Lag24")) {
      model <- lm(Inflation ~ get(lag), data = country_data)
      r_squared <- summary(model)$r.squared
      r_squared_values <- rbind(r_squared_values, data.frame(Country = country, Lag = lag, R2 = r_squared))
    }

    # Add the processed data frame to the list
    regression_data[[country]] <- country_data
  }
}

# Combine data for all countries into one data frame
regression_data_all <- do.call(rbind, regression_data)

# Reshape data for easy plotting across lags
regression_data_long <- regression_data_all %>%
  pivot_longer(cols = starts_with("M2_Growth_Lag"),
               names_to = "Lag",
               values_to = "Lagged_M2_Growth") %>%
  mutate(
    Country = factor(Country, levels = countries),  # Ensure consistent levels
    Lag = factor(Lag, levels = c("M2_Growth_Lag6", "M2_Growth_Lag12", "M2_Growth_Lag18", "M2_Growth_Lag24"),
                 labels = c("6-Month Lag", "12-Month Lag", "18-Month Lag", "24-Month Lag"))
  )

# Merge R-squared values for annotation
regression_data_long <- merge(regression_data_long, r_squared_values,
                              by.x = c("Country", "Lag"), by.y = c("Country", "Lag"),
                              all.x = TRUE)

# Plot the regressions for each lag and country with R^2 values
ggplot(regression_data_long, aes(x = Lagged_M2_Growth, y = Inflation)) +
  geom_point(size = 0.4) +
  geom_smooth(method = "lm", color = "blue", se = TRUE) +
  facet_grid(Country ~ Lag, scales ="free") +
  labs(title = "Regression of Lagged M2 Growth vs. Inflation for Different Lag Periods (1996-2024)",
       x = "Lagged M2 Growth Rate", y = "Inflation Rate") +
  theme_minimal(base_size = 10) + stat_cor( size = 2.1, color = "red")+
  theme(plot.title = element_text(size = 12, face = "bold"),
        strip.text = element_text(size = 12))
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



Plotting the M2 Growth rates against inflation

```
# Load necessary libraries
library(readxl)
library(ggplot2)
library(tidyr)
library(lmtest)
library(dplyr)

# Load the dataset (adjust the path as necessary)
data <- read_excel("C:/Users/user/Documents/Egyetem/Tanítás/2024 ősz/Vállalkozások költségvetési kapcsolatai/Infl
ation model 3.xlsx")

# Convert the date column to Date format
data$Date <- as.Date(data$Date)
data <- data %>% filter(Date >= as.Date("2020-01-01"))
# Reshape the data for plotting
data_long <- data %>%
  select(Date, ends_with("growth_rate"), ends_with("Inflation")) %>%
  pivot_longer(cols = -Date, names_to = "variable", values_to = "value") %>%
  separate(variable, into = c("Country", "Metric"), sep = "_", extra = "merge")

# Plot M2 growth rates and inflation rates for each country with uniform color for inflation
ggplot(data_long, aes(x = Date, y = value, color = Metric)) +
  geom_line() +
  facet_wrap(~ Country, scales = "free", ncol=2) +
  labs(title = "M2 Growth Rates and Inflation Rates 2020-2024") +
  theme_minimal()
```



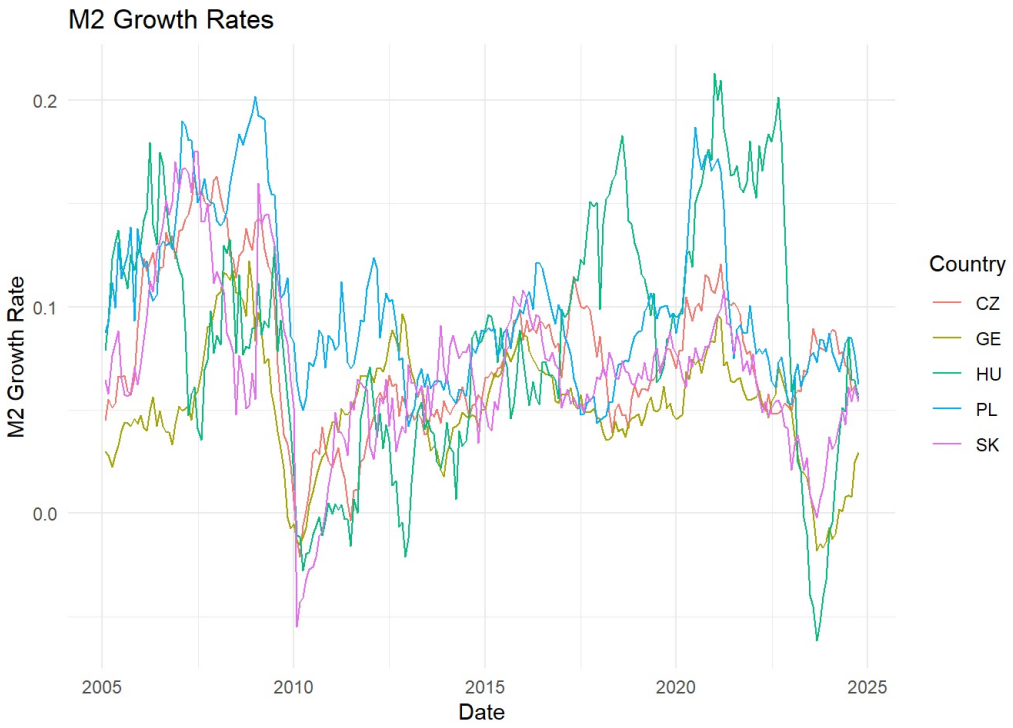
Plotting the M2 Growth rates against inflation with Granger Causality test

```
# Load required libraries
library(ggplot2)
library(reshape2)
```

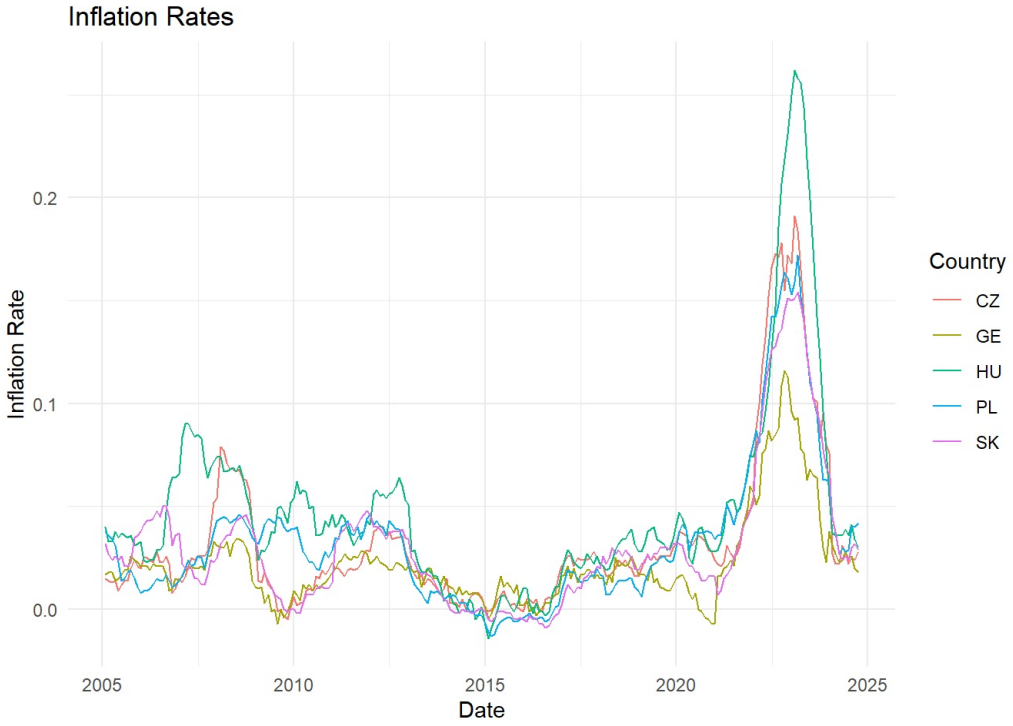
```
##
## Attaching package: 'reshape2'
```

```
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
library(lmtest)

# Load the data
library(readxl)
file_path <- "C:/Users/user/Documents/Egyetem/Tanítás/2024 ősz/Vállalkozások költségvetési kapcsolatai/Inflation
model 4.xlsx"  # Replace with your file path
data <- read_excel(file_path)

# Prepare the data
data$Date <- as.Date(data$Date)  # Convert Date column to Date type

# Define countries and relevant columns
countries <- c("HU", "CZ", "SK", "PL", "GE")
m2_columns <- paste0(countries, "_M2_Growth_rate")
inflation_columns <- paste0(countries, "_Realized_Inflation")

# Reshape data for plotting
m2_data <- melt(data[, c("Date", m2_columns)], id.vars = "Date",
                variable.name = "Country", value.name = "M2_Growth_rate")
m2_data$Country <- gsub("_M2_Growth_rate", "", m2_data$Country)

inflation_data <- melt(data[, c("Date", inflation_columns)], id.vars = "Date",
                       variable.name = "Country", value.name = "Inflation")
inflation_data$Country <- gsub("_Realized_Inflation", "", inflation_data$Country)

# Plot M2 Growth Rates and Inflation Rates
p1 <- ggplot(m2_data, aes(x = Date, y = M2_Growth_rate, color = Country)) +
  geom_line() + theme_minimal() +
  labs(title = "M2 Growth Rates", y = "M2 Growth Rate", x = "Date")

p2 <- ggplot(inflation_data, aes(x = Date, y = Inflation, color = Country)) +
  geom_line() + theme_minimal() +
  labs(title = "Inflation Rates", y = "Inflation Rate", x = "Date")

print(p1)
```




```
print(p2)
```



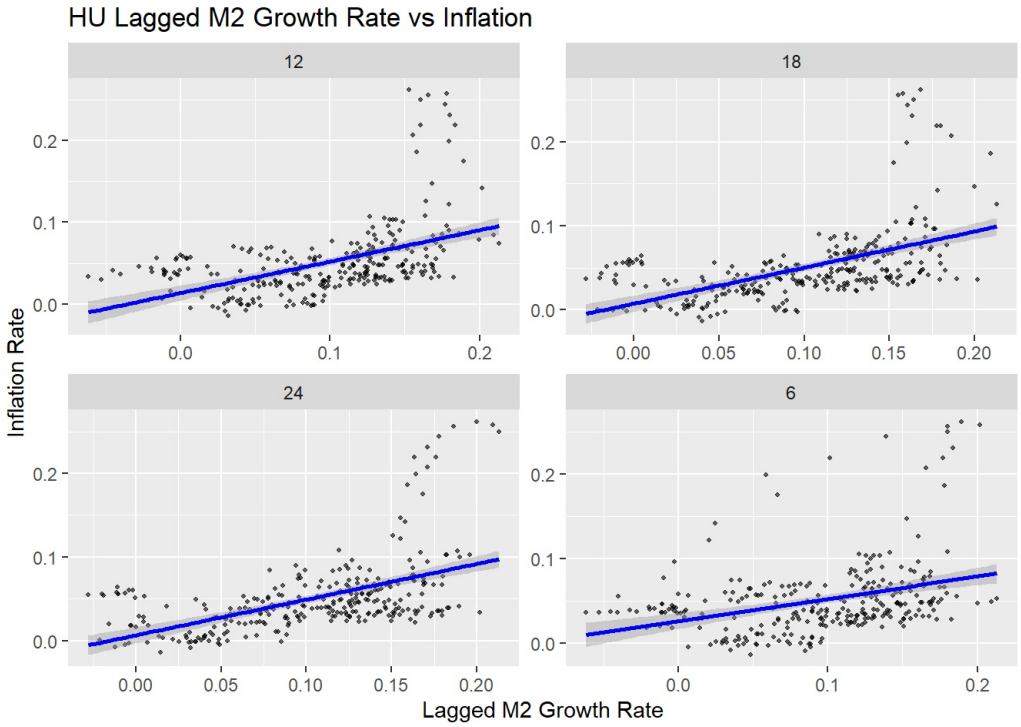

```
# Granger Causality Test
for (country in countries) {
  m2 <- data[[paste0(country, "_M2_Growth_rate")]]
  inflation <- data[[paste0(country, "_Realized_Inflation")]]

  # Remove NA values
  df <- na.omit(data.frame(m2, inflation))

  if (nrow(df) > 0) {
    cat("\nGranger Causality Test for", country, ":\n")
    test <- grangertest(inflation ~ m2, order = 2, data = df)
    print(test)
  }
}
```

```
##
## Granger Causality Test for HU :
## Granger causality test
##
## Model 1: inflation ~ Lags(inflation, 1:2) + Lags(m2, 1:2)
## Model 2: inflation ~ Lags(inflation, 1:2)
##   Res.Df Df      F    Pr(>F)
## 1    230
## 2    232 -2 10.248 5.452e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Granger Causality Test for CZ :
## Granger causality test
##
## Model 1: inflation ~ Lags(inflation, 1:2) + Lags(m2, 1:2)
## Model 2: inflation ~ Lags(inflation, 1:2)
##   Res.Df Df      F Pr(>F)
## 1    230
## 2    232 -2 0.7661  0.466
##
## Granger Causality Test for SK :
## Granger causality test
##
## Model 1: inflation ~ Lags(inflation, 1:2) + Lags(m2, 1:2)
## Model 2: inflation ~ Lags(inflation, 1:2)
##   Res.Df Df     F Pr(>F)
## 1    230
## 2    232 -2 0.641 0.5277
##
## Granger Causality Test for PL :
## Granger causality test
##
## Model 1: inflation ~ Lags(inflation, 1:2) + Lags(m2, 1:2)
## Model 2: inflation ~ Lags(inflation, 1:2)
##   Res.Df Df      F Pr(>F)
## 1    230
## 2    232 -2 1.3249 0.2678
##
## Granger Causality Test for GE :
## Granger causality test
##
## Model 1: inflation ~ Lags(inflation, 1:2) + Lags(m2, 1:2)
## Model 2: inflation ~ Lags(inflation, 1:2)
##   Res.Df Df      F Pr(>F)
## 1    230
## 2    232 -2 1.3258 0.2676
```

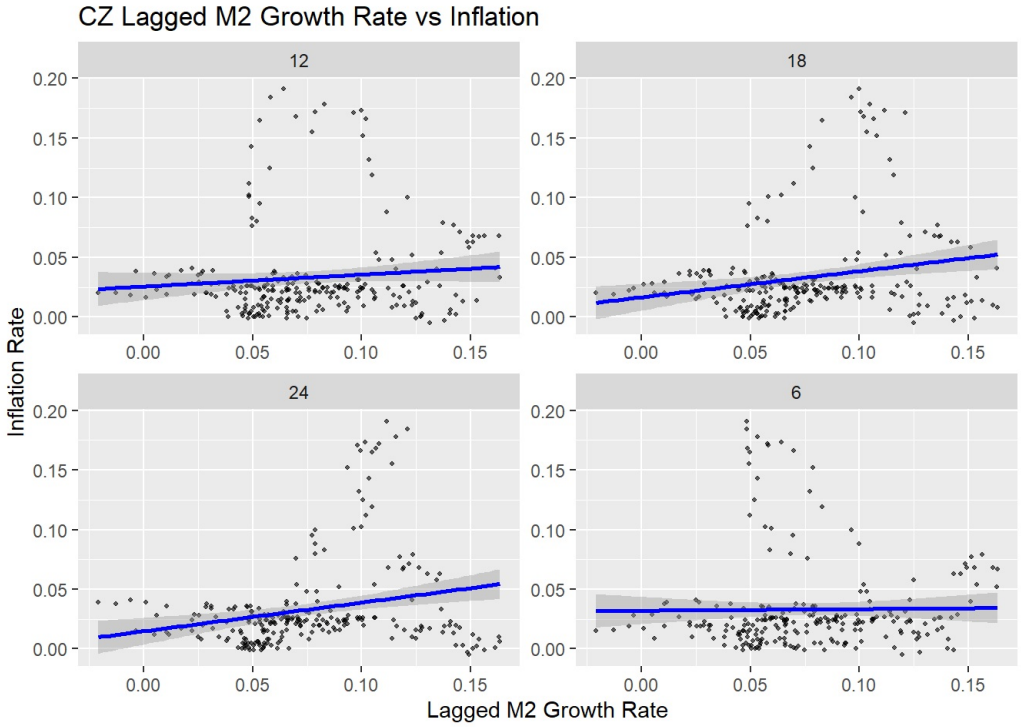
Backup Monetary curves

```r
# Load required libraries
library(ggplot2)
library(dplyr)
library(tidyr)

# Define function to compute lagged variables
add_lags <- function(df, column, lags) {
  for (lag in lags) {
    df[[paste0(column, "_lag_", lag)]] <- dplyr::lag(df[[column]], n = lag)
  }
  return(df)
}

# Lags to include
lags <- c(6, 12, 18, 24)

# Prepare the data
data <- readxl::read_excel("C:/Users/user/Documents/Egyetem/Tanítás/2024 ősz/Vállalkozások költségvetési kapcsola
tai/Inflation model 3.xlsx")  # Update path
data <- data %>% mutate(Date = as.Date(Date))

countries <- c("HU", "CZ", "SK", "PL", "GE")

# Loop over countries and plot separately
for (country in countries) {
  m2_column <- paste0(country, "_M2_Growth_rate")
  inflation_column <- paste0(country, "_Realized_Inflation")

  if (all(c(m2_column, inflation_column) %in% colnames(data))) {
    # Select and clean data
    country_data <- data %>%
      select(Date, all_of(c(m2_column, inflation_column))) %>%
      rename(M2_Growth_rate = all_of(m2_column),
             Inflation = all_of(inflation_column)) %>%
      filter(!is.na(M2_Growth_rate) & !is.na(Inflation)) %>%  # Remove rows with NAs
      add_lags("M2_Growth_rate", lags) %>%
      na.omit()  # Remove rows with NA after adding lags

    # Reshape data for ggplot
    plot_data <- country_data %>%
      pivot_longer(cols = starts_with("M2_Growth_rate_lag_"),
                   names_to = "Lag",
                   values_to = "Lagged_M2") %>%
      mutate(Lag = gsub("M2_Growth_rate_lag_", "", Lag))  # Simplify lag labels

    # Create plot
    p <- ggplot(plot_data, aes(x = Lagged_M2, y = Inflation)) +
      geom_point(alpha = 0.6, size = 0.7) +
      geom_smooth(method = "lm", color = "blue") +
      facet_wrap(~Lag, ncol = 2, scales = "free") +
      labs(title = paste(country, "Lagged M2 Growth Rate vs Inflation"),
           x = "Lagged M2 Growth Rate",
           y = "Inflation Rate")
      theme_minimal(base_size = 10) +
      theme(strip.text = element_text(size = 12),
            plot.title = element_text(size = 12, face = "bold"))

    # Save or print the plot
    print(p)
  }
}
```

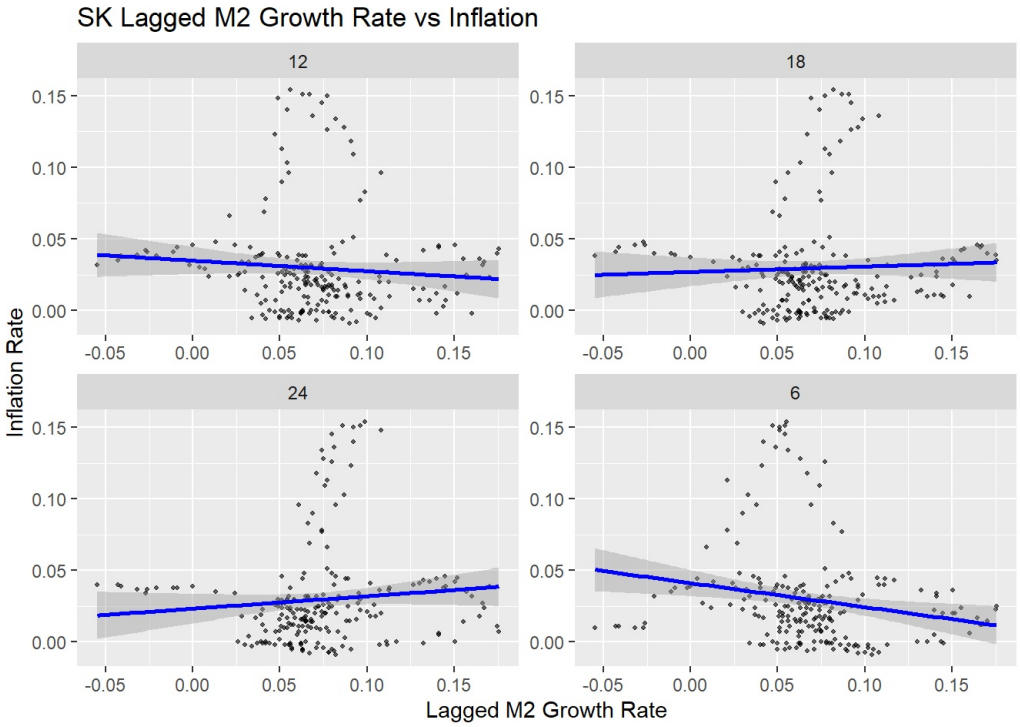
```
## `geom_smooth()` using formula = 'y ~ x'
```



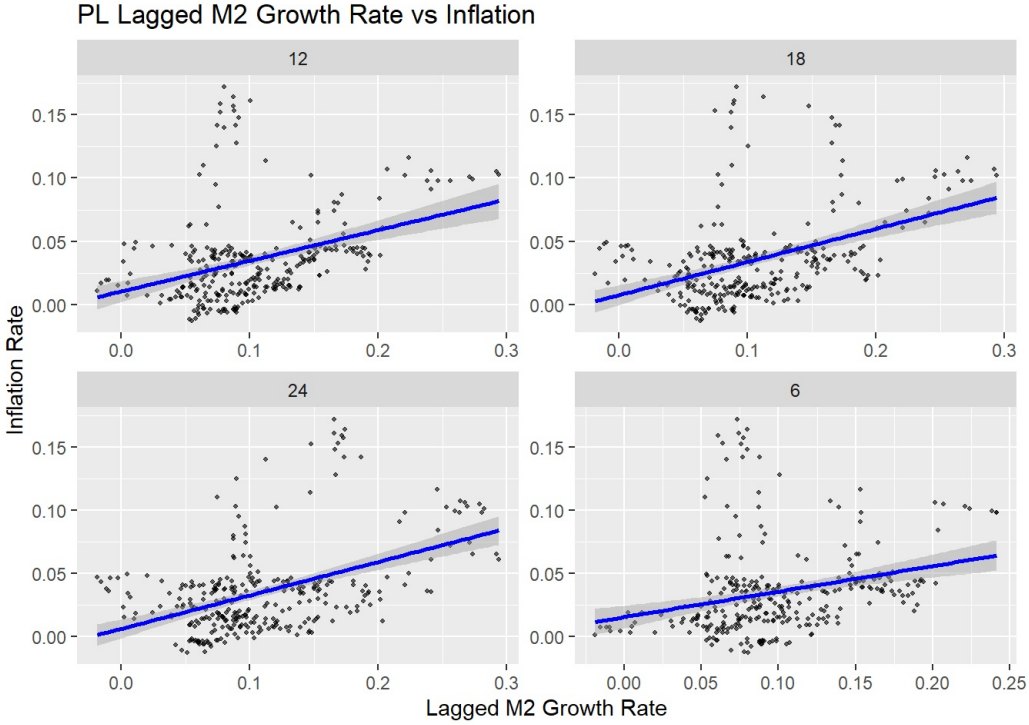

```
## `geom_smooth()` using formula = 'y ~ x'
```



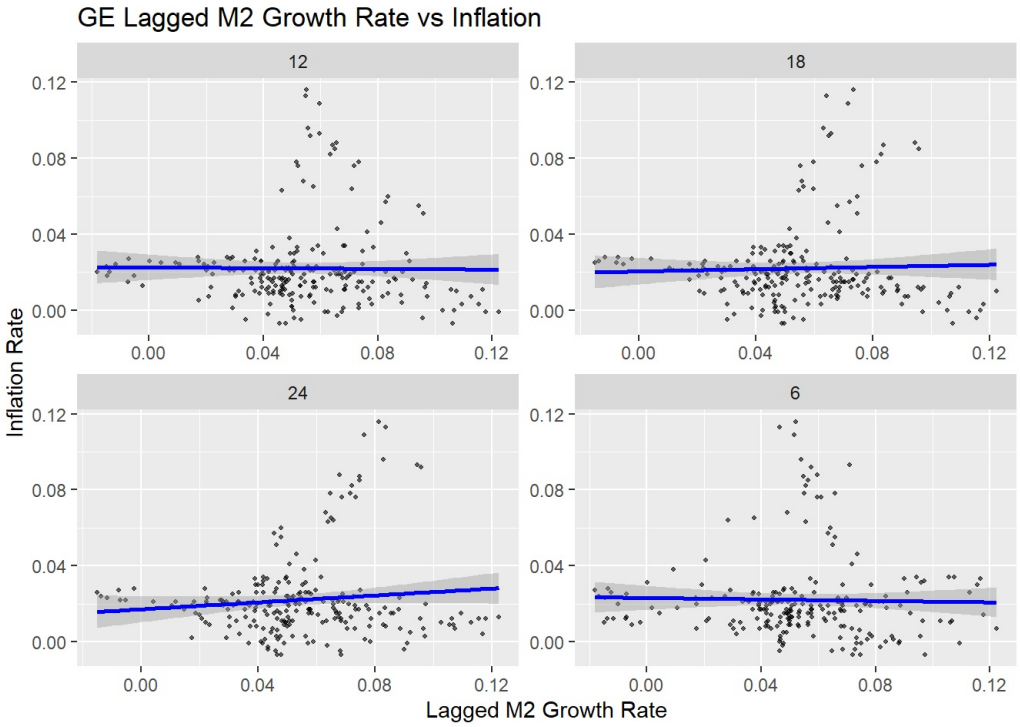

```
## `geom_smooth()` using formula = 'y ~ x'
```




```
## `geom_smooth()` using formula = 'y ~ x'
```




```
## `geom_smooth()` using formula = 'y ~ x'
```


GE Lagged M2 Growth Rate vs Inflation
12
18
24
6
Inflation Rate
Lagged M2 Growth Rate
0.00
0.04
0.08
0.12