MTH516 Project Report

Group 16

2024-11-06

Working with the data

We obtained our dataset from the 3 indicators listed in the Science and Technology section of the World Bank dataset:

- **High technology exports (US$)** (Link): The gain from the export of products involving high intensity of research and development for production.
- **Patent applications by residents** (Link): The number of patent applications filed by the residents of a country.
- **Research and development expenditure (% of GDP)** (Link): The percentage of GDP spent on basic research, applied research, and experimental development.

Importing data

We obtained plain csv files from the provided links and imported them here:

```r
rd_expenditure <- read.csv("R&D_expenditure.csv")
patent_applications <- read.csv("Patent_applications.csv")
tech_exports <- read.csv("High_technology_exports.csv")
```

Data Cleaning

In the given dataset, a lot of NA values are found, thus we have to remove the countries for which we have no data. Additionally, different indicators have different numbers of NA values. Therefore we consider only the intersection of the countries with data for all the indicators.

Further, we interpolate the few values that are missing, ensuring no extra-polation is involved for most accurate data filling.

```r
remove_na_rows_and_columns <- function(data) {
    data <- data[rowSums(is.na(data[,-1])) < 0.5 * (ncol(data)-1), ]
    data <- data[, colSums(is.na(data)) < 0.5 * nrow(data)]
    return(data)
}

na_r_rdexp = remove_na_rows_and_columns(rd_expenditure)
na_r_pa = remove_na_rows_and_columns(patent_applications)
na_r_te = remove_na_rows_and_columns(tech_exports)
```

```r
common_countries <- Reduce(intersect, list(
    na_r_rdexp$Country.Name,
    na_r_pa$Country.Name,
    na_r_te$Country.Name
))

rdexp <- na_r_rdexp[na_r_rdexp$Country.Name %in% common_countries, ]
pa <- na_r_pa[na_r_pa$Country.Name %in% common_countries, ]
hte <- na_r_te[na_r_te$Country.Name %in% common_countries, ]


interpolate_data <- function(data) {

    data[] <- lapply(data[], function(col) {
        return(na.approx(col, na.rm = FALSE))
    })
    return(data)
}

rdexpi <- cbind(
  rdexp[1],as.data.frame(t(interpolate_data(as.data.frame(t(rdexp[-1])))))
)
pai <- cbind(
  rdexp[1],as.data.frame(t(interpolate_data(as.data.frame(t(pa[-1])))))
)
htei <- cbind(
  rdexp[1],as.data.frame(t(interpolate_data(as.data.frame(t(hte[-1])))))
)
```

One-sample Methods

Adding column names and row names to dataset.

```r
rdexp.named <- data.frame(rdexpi)
rownames(rdexp.named) <- tolower( trimws(rdexp.named$Country.Name) )
rdexp.named$Country.Name <- NULL
colnames(rdexp.named) <- substr(
  trimws( colnames(rdexp.named) ),
  start = 2,
  stop = 5
)

pai.named <- data.frame(pai)
rownames(pai.named) <- tolower( trimws(pai.named$Country.Name) )
pai.named$Country.Name <- NULL
colnames(pai.named) <- substr(
  trimws( colnames(pai.named) ),
  start = 2,
  stop = 5
)
```

```r
htei.named <- data.frame(htei)
rownames(htei.named) <- tolower( trimws(htei.named$Country.Name) )
htei.named$Country.Name <- NULL
colnames(htei.named) <- substr(
  trimws( colnames(htei.named) ),
  start = 2,
  stop = 5
)
```

Run test

We examine the trends in R&D expenditure as a percentage of GDP, high-tech export volume (in US$), and the number of patent applications by residents from 2008 to 2021. Our goal is to assess whether these trends have remained stagnant or experienced notable changes over the years.

Two-sided run test is able to tell us whether a given sequence is random or not. In this context, applying run test over the time series data allows us to determine if the observed values fluctuate randomly or we see a predictable trend over the years.

We analyze the percentage of developing and developed nations that have stagnant trends for these parameters over the past two decades. The developing and developed nations have been divided based on World Bank's categorisation.

We have created the following functions for this analysis:

- `perform.run.test.vec`: Performs run test on a numeric vector, ensuring it is not affected by NA values. We get p-value, number of runs, and the actual length of the vector (after removing NA values).
- `perform.run.test`: Performs run test on a dataframe row-by-row, following consolidation of results.
- `proc.run.test.res`: Obtains the verdict for ech row based on a level of significance (default 10%).
- `run.test.res.class`: Returns the class of each country followed by percentage calculation.

```r
perform.run.test.vec <- function (data){
    res <- runs.test(
        data,
        alternative = "two.sided",
        threshold = median( data, na.rm = TRUE ),
        pvalue = "exact"
    )
    return(c(res$p.value, res$runs, as.numeric(res$parameter['n'])))
}

# start, end -> indices of years
perform.run.test <- function (df, start = NULL, end = NULL) {
    res.runs <- matrix(nrow = 0, ncol = 4)
    for (con in rownames(df)){
        if ( !is.null(start) & !is.null(end) ){
            res <- perform.run.test.vec( as.numeric(df[con, start:end]) )
            res.runs <- rbind(res.runs, c(con, res[1], res[2], res[3]))
        } else {
            res <- perform.run.test.vec( as.numeric(df[con, ]) )
            res.runs <- rbind(res.runs, c(con, res[1], res[2], res[3]))
        }
    }

    res.runs <- as.data.frame(res.runs)

    colnames(res.runs) <- c("country", "pvalue", "runs", "length")
    rownames(res.runs) <- res.runs$country
    res.runs$country <- NULL

    return(res.runs)
}

proc.run.test.res <- function (df, alpha = 0.10) {
    out <- data.frame(df)
    out['run.verdict'] <- apply(
      out['pvalue'],
      MARGIN = 1,
      FUN = function (x) if (x <= alpha) 'Reject randomness' else 'Random'
    )

    return(out)
}

developing_countries <- tolower(
  c("Argentina", "Armenia", "Azerbaijan",
    "Belarus", "Brazil", "Bulgaria", "China", "Colombia", "Costa Rica",
    "Ecuador", "Egypt, Arab Rep.", "Georgia", "Guatemala", "Croatia",
    "India", "Kazakhstan", "Kyrgyz Republic", "Latvia", "Moldova",
    "Madagascar", "Mexico", "North Macedonia", "Mongolia", "Mauritius",
    "Malaysia", "Pakistan", "Panama", "Peru", "Romania", "Russian Federation",
    "Thailand", "Tunisia", "Turkiye", "Ukraine", "Uruguay", "South Africa")
)

developed_countries <- tolower(
  c("Austria", "Belgium", "Canada", "Czechia",
    "Denmark", "Estonia", "Finland", "France", "United Kingdom", "Germany",
    "Greece", "Hong Kong SAR, China", "Hungary", "Iceland", "Israel", "Italy",
    "Japan", "Korea, Rep.", "Lithuania", "Luxembourg", "Malta", "Netherlands",
    "Norway", "Poland", "Portugal", "Singapore", "Slovak Republic", "Slovenia",
    "Spain", "Sweden", "United States")
)

country_map <- c(
  setNames(
    rep("Developing", length(developing_countries)),
    developing_countries
  ),
  setNames(
    rep("Developed", length(developed_countries)),
    developed_countries
  )
```

```
)

run.test.res.class <- function(df){
  df.new <- data.frame(df)
  df.new['class'] <- country_map[rownames(df.new)]

  df.new <- df.new[ ! is.na(df.new['class']) ,]

  out <- df.new %>%
    group_by(class, run.verdict) %>%
    summarise(count = n(), .groups = "drop") %>%
    pivot_wider(names_from = run.verdict, values_from = count, values_fill = 0)

  # Percentage of non-random
  out['total'] <- out[2] + out[3]
  out['percent'] <- 100 * (out[2] / out[4])

  return(out)
}
```

For each parameter, we obtain the percentage of countries per class (developing and developed) which have the verdict of not rejecting the hypothesis of randomness. We use 10% level of significance of obtain the verdicts for each country.

Table 1: R&D Expenditure: Class-wise percentage of countries with random trends

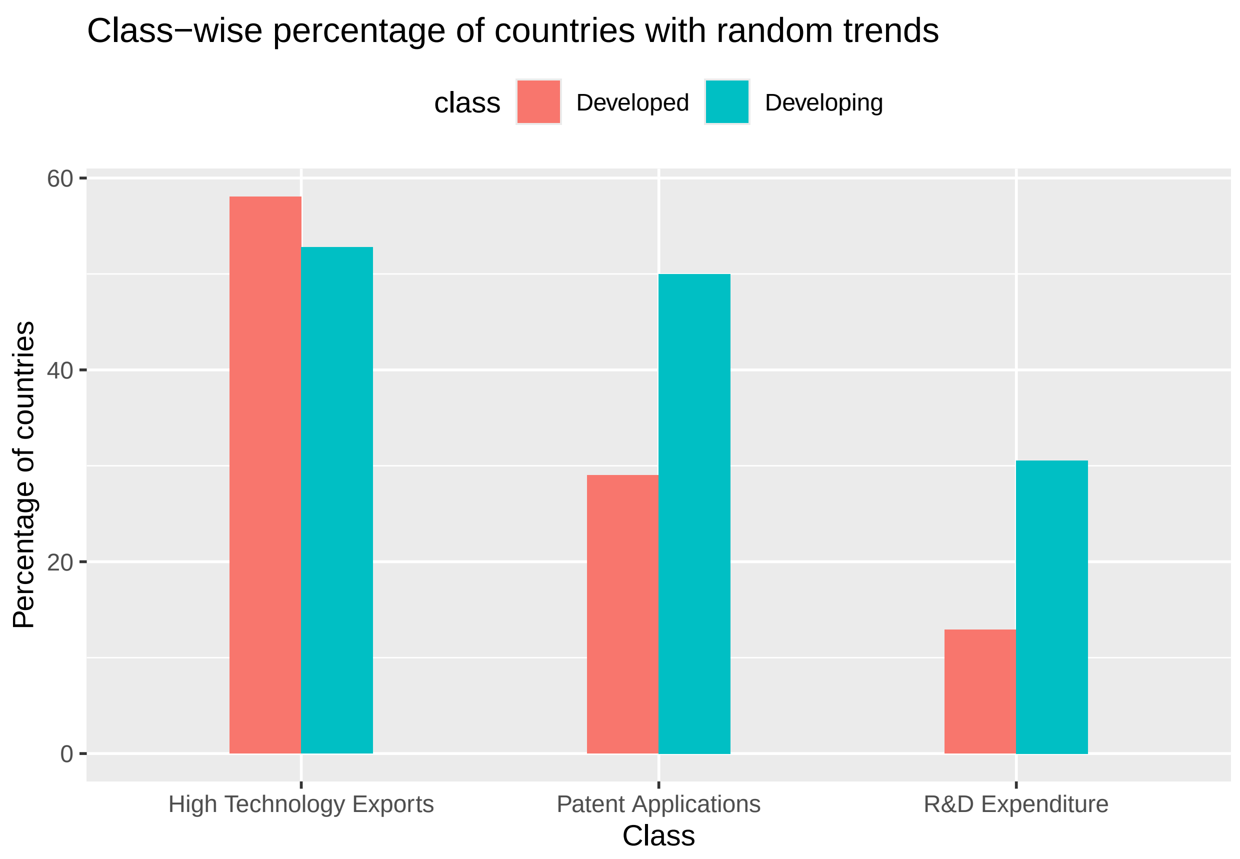
class	percent
Developed	12.90323
Developing	30.55556

Table 2: Patent Applications: Class-wise percentage of countries with random trends

class	percent
Developed	29.03226
Developing	50.00000

Table 3: High Technology Exports: Class-wise percentage of countries with random trends

class	percent
Developed	58.06452
Developing	52.77778



The bar graph illustrates that both developed and developing countries haven't seen much change in high technology exports over the past decades, although this has been couples with significant changes in R&D expenditures by the government.

We also see that developing nations have greater stagnancy in R&D expenditure and patent applications as compared to developed countries. It would be interesting to see whether these are increasing trends or decreasing trends.

Mann-Kendall Test

The Mann-Kendall is a non-parametric test used to identify monotonic trends in time-series data. It tests the null hypothesis of randomness (no trend) against either one-sided or two-sided alternatives, indicating either an increasing or decreasing trend, or any monotonic trend, respectively.

The test statistic is computed as the sum of signs of differences between each pair of time-ordered data points. Extreme values suggest significant trends, while moderate values indicate absence of any such trend.

The Mann-Kendall's test is closely related to the Kendall's tau correlation test. Essentially, the Mann-Kendall statistic is just the Kendall's tau computed between the time series data and time.

Similar to the run test above, we have the following functions:

- `perform.mk.test.vec`: Performs the Mann-kendall test on a numeric vector, ensuring it is not affected by NA values. We get p-value, number of runs, and the actual length of the vector (after removing NA values).

- `perform.mk.test`: Performs the Mann-kendall test on a dataframe row-by-row, following consolidation of results.
- `proc.mk.test.res`: Obtains the verdict for each row based on a level of significance (default 10%).
- `mk.test.res.class`: Returns the class of each country followed by percentage calculation.

```
perform.mk.test.vec <- function (data, alternative = c("two.sided", "greater", "less")){
    res <- mk.test(
        data[ ! is.na(data) ],
        alternative = alternative
    )
    return( c(
      as.numeric(res$p.value),
      as.numeric(res$parameter),
      as.numeric(res$estimates['S']))
    )
}


# start, end -> indices of years
perform.mk.test <- function (df, start = NULL, end = NULL) {
    # 2-sided alternative
    res.2.sided <- matrix(nrow = 0, ncol = 4)
    for (con in rownames(df)){
        if ( !is.null(start) & !is.null(end) ){
            res <- perform.mk.test.vec( as.numeric(df[con, start:end]) )
            res.2.sided <- rbind(res.2.sided, c(con, res[1], res[2], res[3]))
        } else {
            res <- perform.mk.test.vec( as.numeric(df[con, ]) )
            res.2.sided <- rbind(res.2.sided, c(con, res[1], res[2], res[3]))
        }
    }

    res.2.sided <- as.data.frame(res.2.sided)

    colnames(res.2.sided) <- c("country", "two.sided.pvalue", "length", "S")

    # Greater alternative
    res.greater <- matrix(nrow = 0, ncol = 4)
    for (con in rownames(df)){
        if ( !is.null(start) & !is.null(end) ){
            res <- perform.mk.test.vec( as.numeric(df[con, start:end]), "greater" )
            res.greater <- rbind(res.greater, c(con, res[1], res[2], res[3]))
        } else {
            res <- perform.mk.test.vec( as.numeric(df[con, ]), "greater" )
            res.greater <- rbind(res.greater, c(con, res[1], res[2], res[3]))
        }
    }
    res.greater <- as.data.frame(res.greater)

    colnames(res.greater) <- c("country", "greater.pvalue", "length", "S")
    res.greater <- res.greater[-c(3,4)]

    # Lesser alternative
    res.lesser <- matrix(nrow = 0, ncol = 4)
    for (con in rownames(df)){
        if ( !is.null(start) & !is.null(end) ){
            res <- perform.mk.test.vec( as.numeric(df[con, start:end]), "less" )
            res.lesser <- rbind(res.lesser, c(con, res[1], res[2], res[3]))
        } else {
            res <- perform.mk.test.vec( as.numeric(df[con, ]), "less" )
            res.lesser <- rbind(res.lesser, c(con, res[1], res[2], res[3]))
        }
    }
    res.lesser <- as.data.frame(res.lesser)

    colnames(res.lesser) <- c("country", "lesser.pvalue", "length", "S")
    res.lesser <- res.lesser[-c(3,4)]

    # Combining all the p-values
    out <- merge(
      merge(res.2.sided, res.greater, by="country"), res.lesser, by="country"
    )

    rownames(out) <- out$country
    out$country <- NULL

    out <- out[, c("S", "length", "two.sided.pvalue", "greater.pvalue", "lesser.pvalue")]
    out["two.sided.pvalue"] <- as.numeric( out[["two.sided.pvalue"]] )
    out["greater.pvalue"] <- as.numeric( out[["greater.pvalue"]] )
    out["lesser.pvalue"] <- as.numeric( out[["lesser.pvalue"]] )

    return(out)
}

proc.mk.test.res <- function (df, alpha = 0.10) {
    out <- data.frame(df)
    out["mk.verdict"] <- NA

    for (con in rownames(out)){
        if (out[con, "two.sided.pvalue"] > alpha) {
            out[con, "mk.verdict"] <- 'no.trend'
        }
        else if (out[con, "greater.pvalue"] <= alpha) {
            out[con, "mk.verdict"] <- 'greater'
        }
        else if (out[con, "lesser.pvalue"] <= alpha) {
            out[con, "mk.verdict"] <- 'lesser'
        }
    }

    return(out)
}
mk.test.res.class <- function(df){
    df.new <- data.frame(df)
    df.new['class'] <- country_map[rownames(df.new)]

    df.new <- df.new[ ! is.na(df.new['class']) ,]
```

```r
    out <- df.new %>%
      group_by(class, mk.verdict) %>%
      summarise(count = n(), .groups = "drop") %>%
      pivot_wider(names_from = mk.verdict, values_from = count, values_fill = 0)

    out["greater.percent"] <- 100 * out[2] / (out[2] + out[3] + out[4])
    out["lesser.percent"] <- 100 * out[3] / (out[2] + out[3] + out[4])
    out["no.trend.percent"] <- 100 * out[4] / (out[2] + out[3] + out[4])

    return(out)
}
```

Again, for each parameter, we obtain the percentage of countries per class (developing and developed) which have different verdicts. We use 10% level of significance of obtain the verdicts for each country.

Here, we use all the three alternatives to ensure we have the most likely verdict. We use the following rule to obtain the verdict:

1. Two-sided test: Rejection of the null hypothesis indicates trend. However, if the null is not rejected, we give the verdict of "no.trend".
2. One-sided tests: The test is performed for both one-sided alternatives. We base the verdict on the alternative with a reasonably small p-value ignoring the other alternative.

Table 4: R&D Expenditure: Class-wise percentage of countries

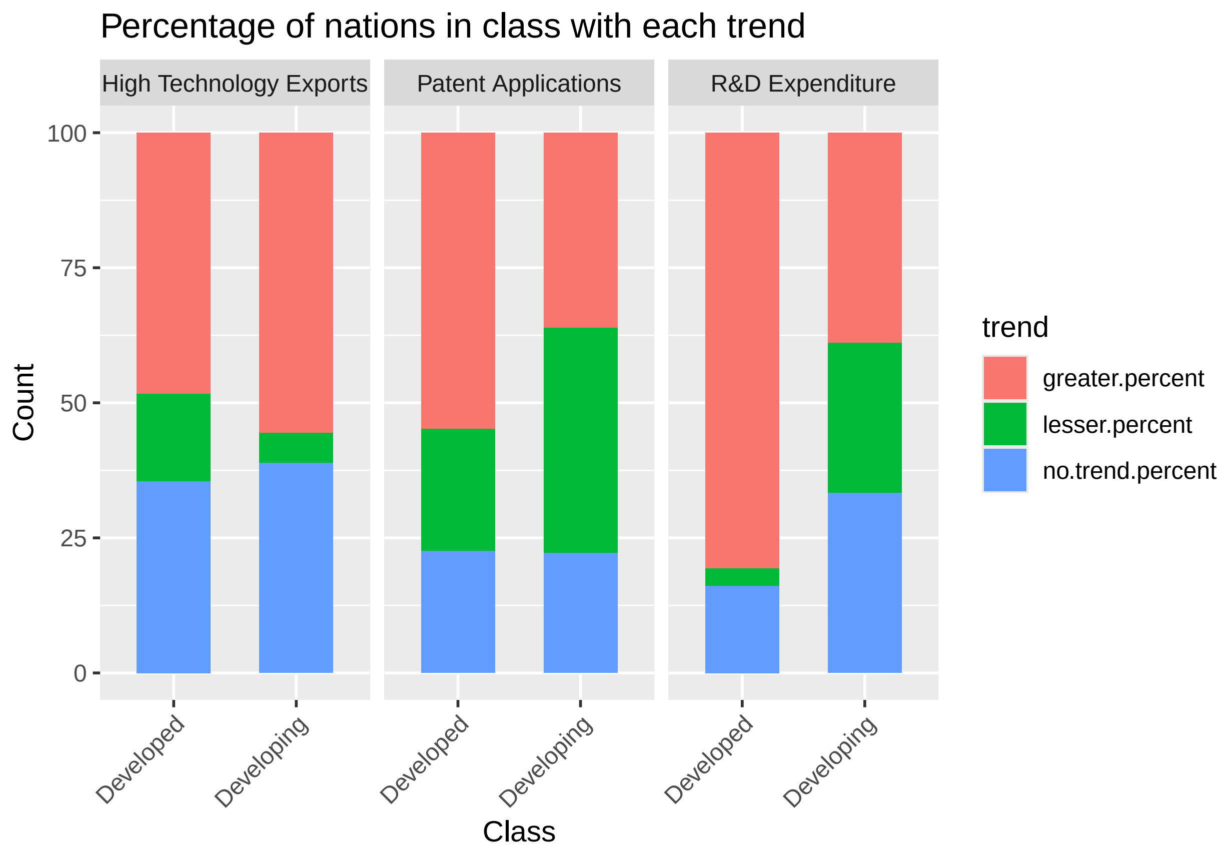
class	greater.percent	lesser.percent	no.trend.percent
Developed	80.64516	3.225807	16.12903
Developing	38.88889	27.777778	33.33333

Table 5: Patent applications: Class-wise percentage of countries

class	greater.percent	lesser.percent	no.trend.percent
Developed	54.83871	22.58065	22.58065
Developing	36.11111	41.66667	22.22222

Table 6: High technology exports: Class-wise percentage of countries

class	greater.percent	lesser.percent	no.trend.percent
Developed	48.38710	16.129032	35.48387
Developing	55.55556	5.555556	38.88889



First, we address the R&D expenditure indicator's results. They indicate that many developed nations have increased their expenditure in research and development, although the trend is mixed for developing countries. Additionally, the results agree with the run test for this indicator.

Second, mann-kendall test indicated lower percentage of stagnancy for high technology exports and patent applications than run test. Considering the mixed results, we can conclude that there may exist weak trends in these parameters.

Finally, the high technology exports from both the developing and developed nations has been steadily increasing. Since the run test showed significant stagnancy at 10% level of significance, these may be weak increasing trends. Further, there is a disproportionately higher percentage of increasing trends in patent applications filed from developed countries as compared to the developing countries.

Two Sample Methods

We have defined a function that will take two rows of our data. The data rows can belong to same country different indicator or different country same indicator. The function computes Spearman and Kendall rank correlation of both the data and plots a line plot for normalised data.

```r
# Function to compare two time series data
compare_time_series <- function(
    data1,
    data2,
    title="Comparison of Time Series",
    series1_name="Series 1",
    series2_name="Series 2",
    ret=FALSE,
    plot=TRUE,
    normalise_forplot=TRUE
  ) {

  t1 = names(data1[,!is.na(data1)])
  t2 = names(data2[,!is.na(data2)])
  time = Reduce(intersect,list(t1,t2))
  data1_valid <- as.numeric(data1[time])
  data2_valid <- as.numeric(data2[time])

  if(length(data1_valid) < 2) {
    cat("Not enough data points")
    #cat(title," ",series1_name," ",series2_name)
    return(0)
  }

  # Compute Spearman's rho
  spearman_result <- cor.test(data1_valid, data2_valid, method = "spearman")
  spearman_rho <- spearman_result$estimate
  spearman_p <- spearman_result$p.value

  # Compute Kendall's tau
  kendall_result <- cor.test(data1_valid, data2_valid, method = "kendall")
  kendall_tau <- kendall_result$estimate
  kendall_p <- kendall_result$p.value

  if(ret)
  {
    return(list(spearman_result,kendall_result))
  }
  # Print the results
  cat("Spearman's rho:", round(spearman_rho, 4), "p-value:", spearman_p, "\n")
  cat("Kendall's tau:", round(kendall_tau, 4), "p-value:", kendall_p, "\n")

  # Normalize the data (min-max normalization)
  normalize <- function(x) {
    return((x - min(x, na.rm=TRUE)) / (max(x, na.rm=TRUE) - min(x, na.rm=TRUE)))
  }

  if(plot)
  {
    if(normalise_forplot)
    {
      data1_norm <- normalize(data1)
      data2_norm <- normalize(data2)
    }
    else
    {
      data1_norm=data1
      data2_norm=data2
    }
    time2 = Reduce(union,list(names(data1),names(data2)))
    data1_norm = data.frame(Year=names(data1_norm),Value=t(data1_norm))
    data2_norm = data.frame(Year=names(data2_norm),Value=t(data2_norm))
    data_combined <- merge(data1_norm,data2_norm,by='Year',suffixes = c("_1", "_2"))
    names(data_combined) <- c('Year','Series1','Series2')
    # Plot with ggplot2
    ggplot(data_combined, aes(x = Year)) +
      geom_line(aes(y = Series1, color = series1_name,group = 1 )) +
      geom_line(aes(y = Series2, color = series2_name,group =1)) +
      labs(x = "Year", y = "Normalised Value", title = title)+
      scale_color_manual(name = NULL, values = c("blue", "red")) +
     theme_minimal()+theme(axis.text.x = element_text(angle = 90, hjust = 1))
  }


}
```

Spearman's rho and **Kendall's tau** measure the strength and direction of the monotonic relationship between the two variables.

P-values indicate the statistical significance of the correlation. A p-value less than 0.05 typically suggests a significant correlation.

Friedman Test | Country wise

```r
run_friedman_test <- function(country) {
  # Extract data for the specified country and remove NAs
  rdexp_country <- na.omit(data.frame(
    year = colnames(rdexp.named),
    RnD_expenditure = as.numeric(rdexp.named[country, ])
  ))
  pai_country <- na.omit(data.frame(
    year = colnames(pai.named),
    Patent_applications = as.numeric(pai.named[country, ])
  ))
  hte_country <- na.omit(data.frame(
    year = colnames(htei.named),
    High_tech_exports = as.numeric(htei.named[country, ])
  ))

  # Merge data by common years across all indicators
  merged_data <- purrr::reduce(
      list(rdexp_country, pai_country, hte_country), full_join, by = "year"
    ) %>%
    na.omit()  # Keep only complete cases after merging

  # Reshape data to long format for Friedman test
  friedman_data_long <- tidyr::pivot_longer(
    merged_data,
    cols = -year,
    names_to = "Indicator",
    values_to = "Value"
  )

  # Run Friedman test
  friedman_test_result <- friedman.test(
    Value ~ Indicator | year, data = friedman_data_long
  )
  return(friedman_test_result)
}
```

The Friedman test provides a chi-squared statistic and a p-value. A significant p-value (usually less than 0.05) suggests that at least one indicator differs significantly in its average ranking across the years. This can be interpreted as evidence that the relative trends of these indicators are not aligned, meaning that the growth or decline patterns differ among R&D expenditure, patent applications, and high-tech exports for that country due to some reason.

A non-significant p-value indicates no substantial differences in the ranks of these indicators over the observed period, suggesting that they may be following similar trends over time.

```r
for( i in tolower(common_countries))
{
  result = run_friedman_test(i)
  if(result$p.value>0.01)
  {
    cat(
      "Country ",i,": We cannot reject the fact that the ranks of the \
      three indicators is statistically different"
    )
    print(result)
  }
}
```

For all countries we can reject the hypothesis that the ranks of the three indicators are different.

We will conduct pairwise tests on indicators of countries

```r
reject_count <- c(0, 0, 0)

# Lists to store countries for each comparison
patent_vs_rd_countries <- list()
patent_vs_hte_countries <- list()
rd_vs_hte_countries <- list()

print("Patent Applications vs R&D expenditure")
```

```
## [1] "Patent Applications vs R&D expenditure"
```

```
for (i in tolower(common_countries)) {
  flag <- FALSE
  ret <- compare_time_series(
    data1 = pai.named[i,],
    data2 = rdexp.named[i,],
    title = i,
    series1_name = "Patent Applications",
    series2_name = "R&D expenditure",
    ret = TRUE
  )
  if (ret[[1]]$p.value < 0.05 && ret[[2]]$p.value < 0.05) {
    reject_count[1] <- reject_count[1] + 1
    flag <- TRUE
  }
  if (flag) {
    patent_vs_rd_countries <- append(patent_vs_rd_countries, i)
  }
}

print("Patent Applications vs High tech export")
```

```
## [1] "Patent Applications vs High tech export"
```

```
for (i in tolower(common_countries)) {
  flag <- FALSE
  ret <- compare_time_series(
    data1 = pai.named[i,],
    data2 = htei.named[i,],
    title = i,
    series1_name = "Patent Applications",
    series2_name = "High tech export",
    ret = TRUE
  )
  if (ret[[1]]$p.value < 0.05 && ret[[2]]$p.value < 0.05) {
    reject_count[2] <- reject_count[2] + 1
    flag <- TRUE
  }
  if (flag) {
    patent_vs_hte_countries <- append(patent_vs_hte_countries, i)
  }
}

print("R&D expenditure vs High tech export")
```

```
## [1] "R&D expenditure vs High tech export"
```

```
for (i in tolower(common_countries)) {
  flag <- FALSE
  ret <- compare_time_series(
    data1 = rdexp.named[i,],
    data2 = htei.named[i,],
    title = i,
    series1_name = "R&D expenditure",
    series2_name = "High tech export",
    ret = TRUE
  )
  if (ret[[1]]$p.value < 0.05 && ret[[2]]$p.value < 0.05) {
    reject_count[3] <- reject_count[3] + 1
    flag <- TRUE
  }
  if (flag) {
    rd_vs_hte_countries <- append(rd_vs_hte_countries, i)
  }
}

# Display the results
cat("Countries with significant results for Patent Applications vs \
    R&D expenditure:\n", unlist(patent_vs_rd_countries), "\n")
```

```
## Countries with significant results for Patent Applications vs
##     R&D expenditure:
##  argentina armenia belgium bulgaria belarus brazil canada china colombia costa rica czechia europe &
```

```
cat("Countries with significant results for Patent Applications vs \
    High tech export:\n", unlist(patent_vs_hte_countries), "\n")
```

```
## Countries with significant results for Patent Applications vs
##     High tech export:
##  armenia azerbaijan bulgaria belarus china spain finland greece croatia india kazakhstan korea, rep.
```

```
cat("Countries with significant results for R&D expenditure vs \
    High tech export:\n", unlist(rd_vs_hte_countries), "\n")
```

```
## Countries with significant results for R&D expenditure vs
##     High tech export:
##  armenia bulgaria china colombia czechia estonia finland greece high income hong kong sar, china cro
```

```
cat("\nReject count:", reject_count, "\n")
```

```
##
## Reject count: 37 24 31
```

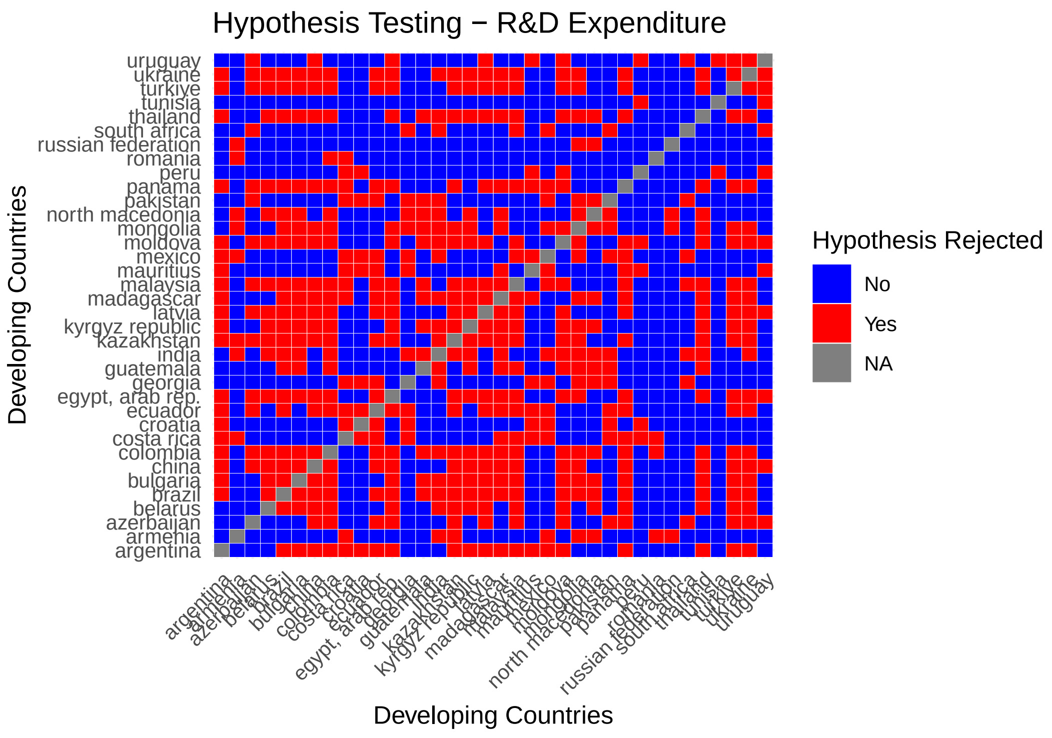
We can say that in general the parameters we have selected are not dependent on each other (not correlated).

We will now compare chosen parameters on different countries:

Comparison of two countries on a single indicator

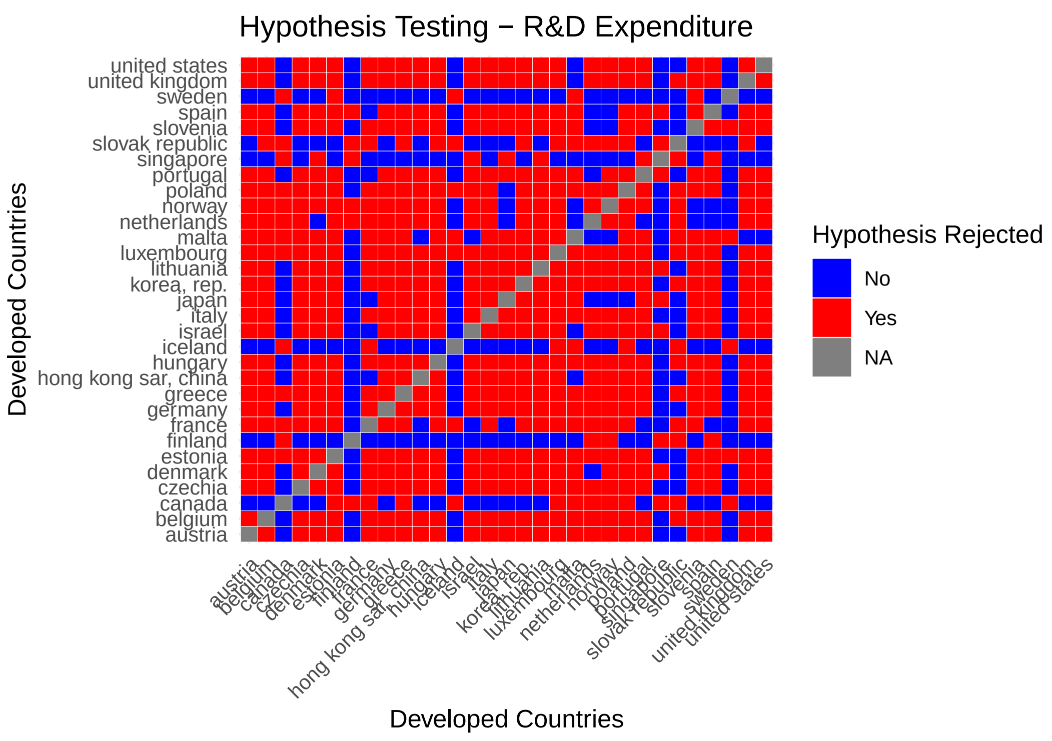
R&D Expenditure: Developing Countries

```r
# Initialize a list to store results
hypothesis_results <- list()

# Loop over all pairs of countries
for (i in developing_countries) {
  for (j in developing_countries) {
    if (i != j) {  # To avoid comparing the same country with itself
      # Call the compare_time_series function and get results
      ret <- compare_time_series(
        data1 = rdexp.named[i, ],
        data2 = rdexp.named[j, ],
        title = "R&D export",
        series1_name = i,
        series2_name = j,
        ret = TRUE,
        plot = FALSE
      )

      # Check if the function returned a valid result
      if (is.list(ret)) {
        # Extract p-values if valid results are returned
        spearman_p <- ret[[1]]$p.value
        kendall_p <- ret[[2]]$p.value

        # Check if both Spearman and Kendall p-values are less than 0.05
        hypothesis_rejected <- (spearman_p < 0.05) && (kendall_p < 0.05)
      } else {
        # If not enough data points, mark as NA or FALSE
        hypothesis_rejected <- NA
      }

      # Store the result in the list
      hypothesis_results[[paste(i, j, sep = "_vs_")]] <- hypothesis_rejected
    }
  }
}

# Convert results to a data frame for easy viewing
hypothesis_df <- data.frame(Pair = names(hypothesis_results),
                            Hypothesis_Rejected = unlist(hypothesis_results))
hypothesis_df <- cbind(
  colsplit(hypothesis_df$Pair, "_vs_", names = c("Country1", "Country2")),
  Hypothesis_Rejected = hypothesis_df$Hypothesis_Rejected
)

heatmap_data <- dcast(
  hypothesis_df,
  Country1 ~ Country2,
  value.var = "Hypothesis_Rejected"
)

# Convert to long format for ggplot
heatmap_data_long <- melt(
  heatmap_data,
  id.vars = "Country1",
  variable.name = "Country2",
  value.name = "Hypothesis_Rejected"
)

# Plot the heatmap
ggplot(heatmap_data_long, aes(x = Country1, y = Country2, fill = Hypothesis_Rejected)) +
  geom_tile(color = "white") +
  scale_fill_manual(values = c("TRUE" = "red", "FALSE" = "blue"),
                    name = "Hypothesis Rejected",
                    labels = c("No", "Yes")) +
  labs(title = "Hypothesis Testing - R&D Expenditure",
       x = "Developing Countries", y ="Developing Countries") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

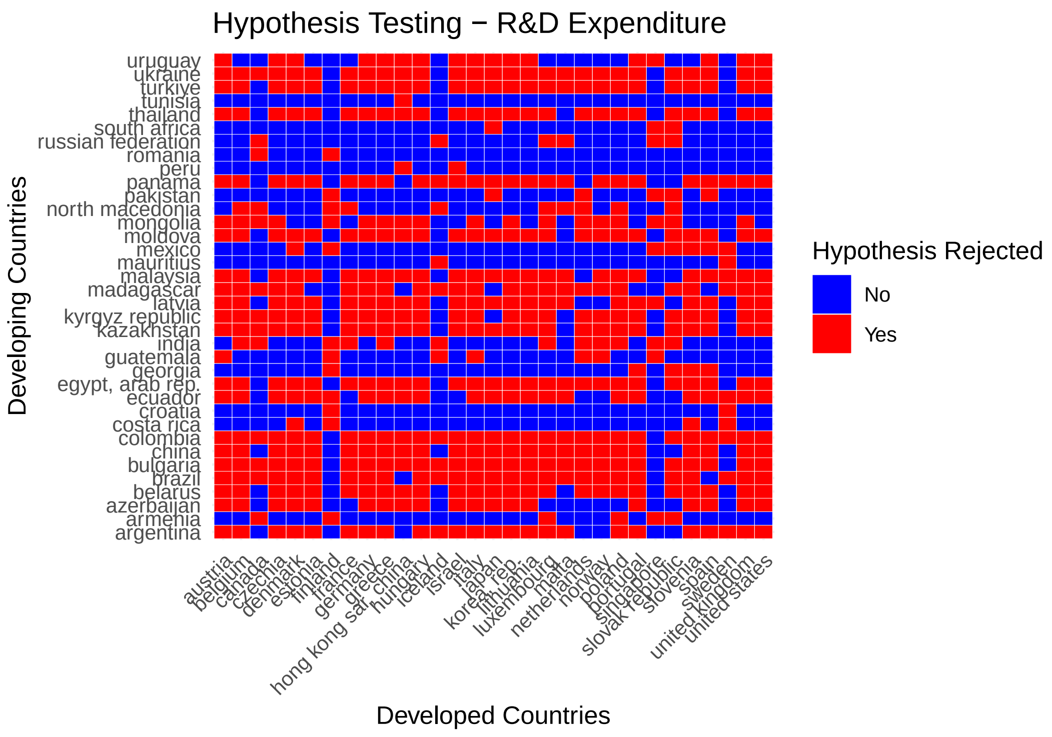


```
# Display the results
#print(hypothesis_df)
```

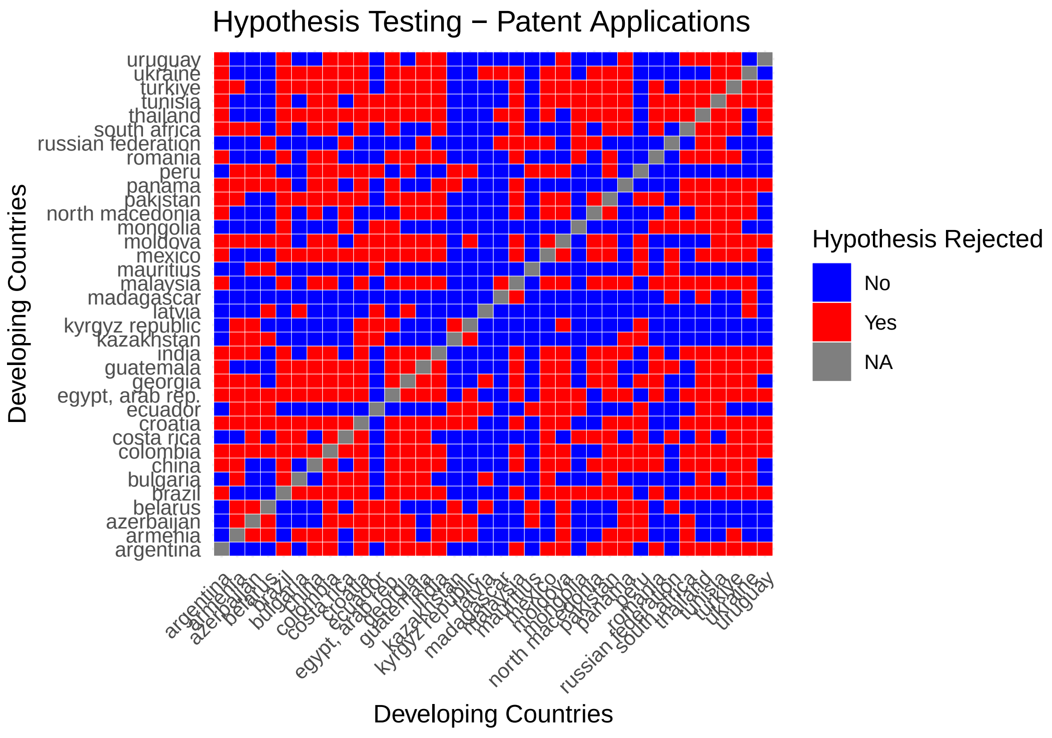
R&D Expenditure: Developed Countries



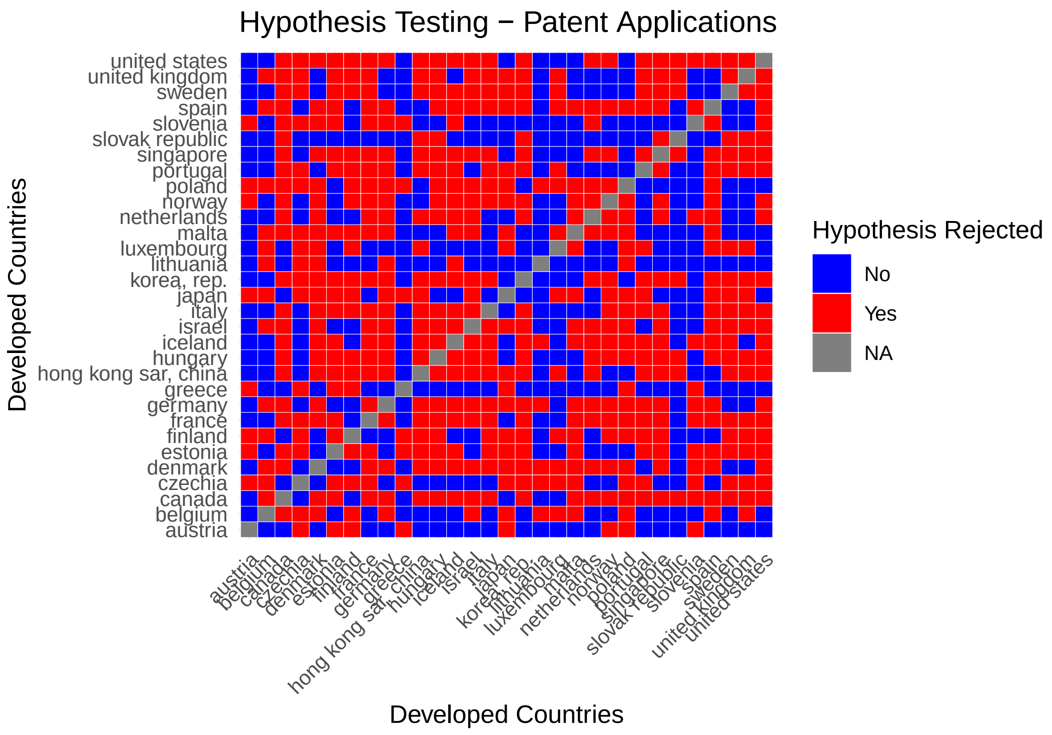
R&D Expenditure: Developed vs Developing Countries



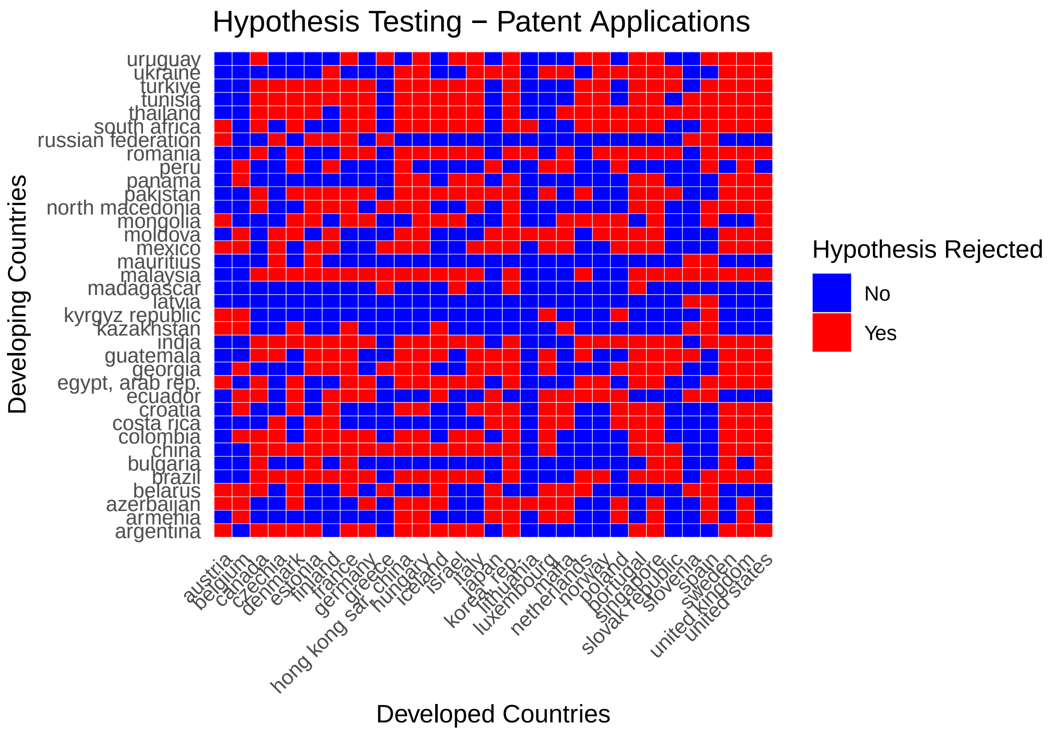
Patent Applications: Developing Countries



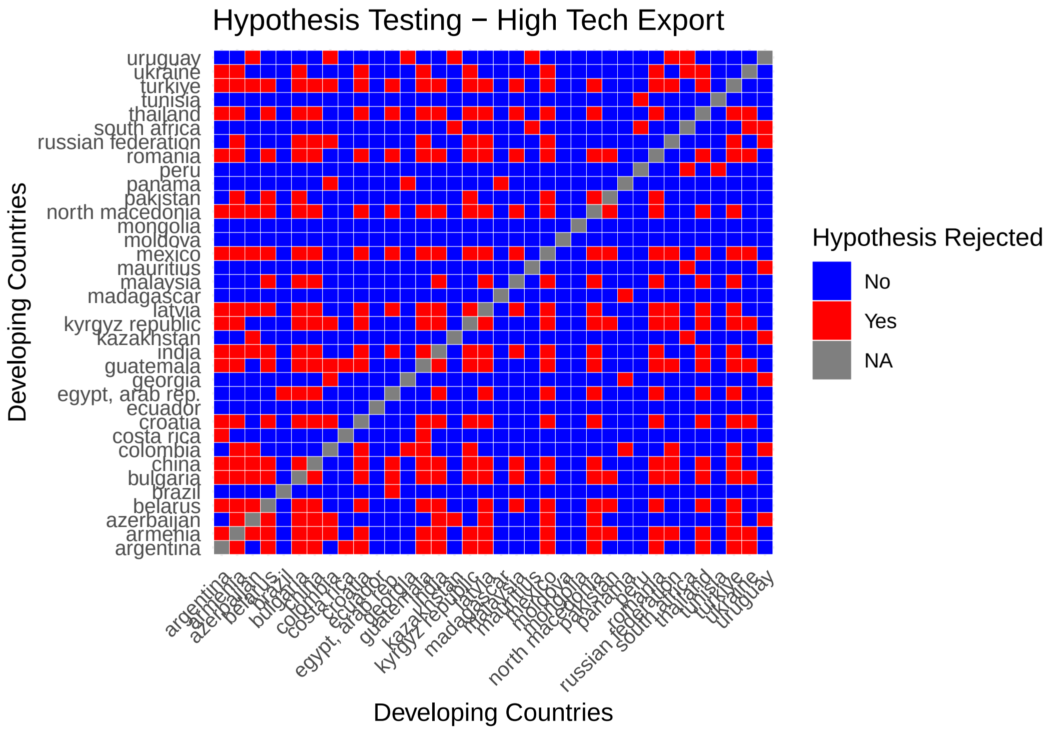
Patent Applications: Developed Countries



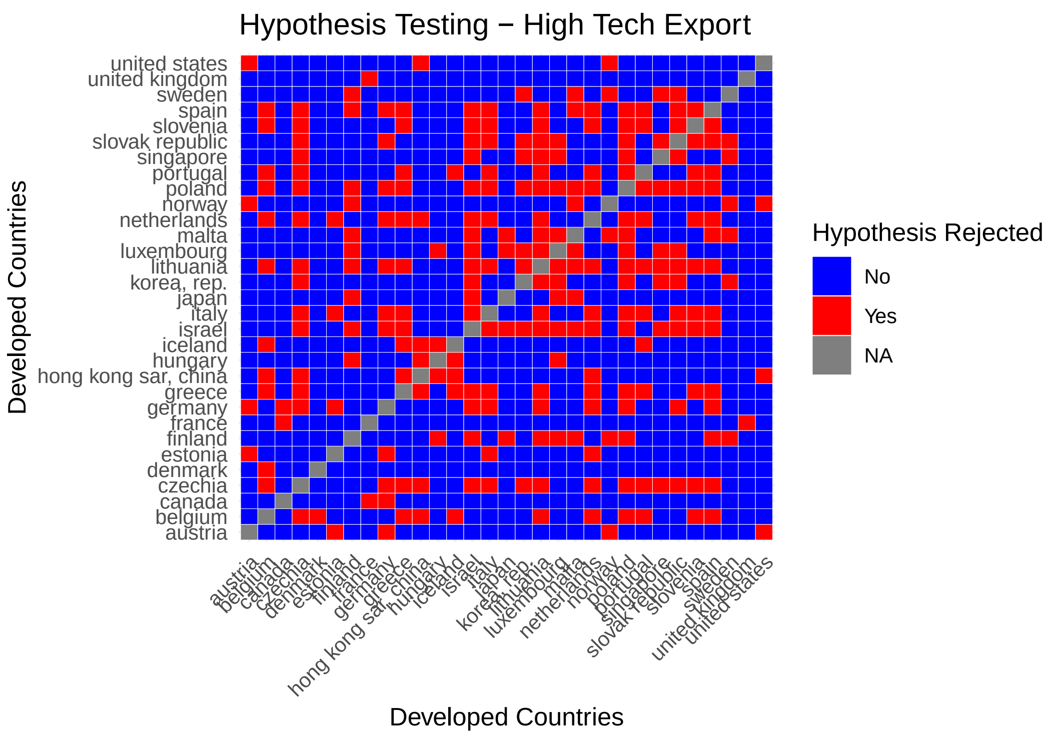
Patent Applications: Developed vs Developing Countries



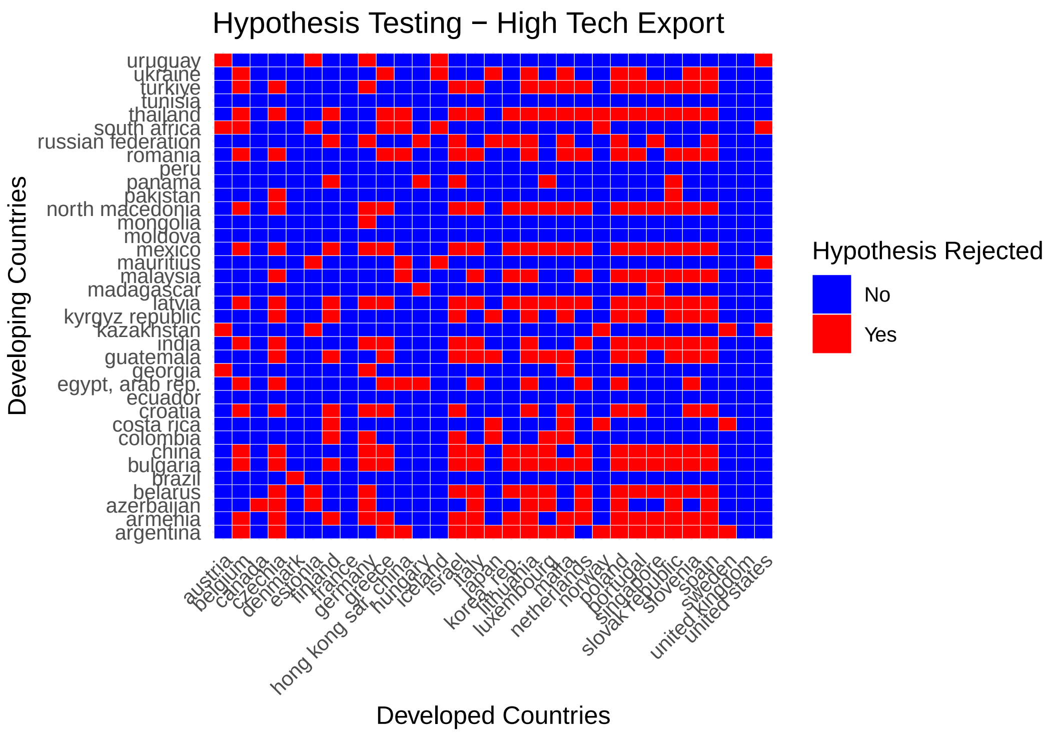
High Tech Export: Developing Countries



High Tech Export: Developed Countries



High Tech Export: Developed vs Developing Countries



A major portion of high tech export are colored Blue. Which means we cannot reject the hypothesis that the high tech export ranks are correlated.

We can say that since trade between countries is interlinked, high tech exports are also interlinked.

Non-parametric Time series forecasting

By assuming the time series follows a stationary Markov process, forecasts are derived by nonparametrically estimating the autoregressive function. This study explores two distinct forecasting methods, each employing a different kernel-type estimator: the *Nadaraya-Watson estimator* and the *local linear estimator*.

Core Challenges in Nonparametric Forecasting

The use of nonparametric forecasting methods involves addressing three main challenges:

1. **Autoregressor Variable Selection**: Identifying which lagged values provide the most relevant information for accurate forecasting.
2. **Smoothing Parameter Selection**: Determining the optimal bandwidth to achieve a balance between bias and variance in the estimate.

Considering a strictly stationary univariate time series $\{Z_t\}_{t\in\mathbb{Z}}$, observed within $1 \leq t \leq n$, the series is modeled under the Markov assumption as:

$$Z_t = m(Z_{t-i_1}, Z_{t-i_2}, \ldots, Z_{t-i_p}) + \epsilon_t$$

where: - m denotes the autoregressive function, - $Z_{t-i_1}, Z_{t-i_2}, \ldots, Z_{t-i_p}$ are previous values (lags), - ϵ_t represents an error term, assumed to be independent of the past of Z_t and to have a mean of zero.

Nonparametric Regression and Forecasting

Traditional methods estimate m parametrically by assuming it belongs to a function class with a fixed number of parameters, such as in ARIMA models. *Nonparametric regression*, however, offers greater flexibility by avoiding strict structural assumptions about m, which is advantageous for forecasting complex, irregular time series patterns.

This study employs *kernel-based estimation* of the regression function as a nonparametric approach to forecasting, which frames forecasting as a specific case of nonparametric regression in the context of dependent data.

Nadaraya-Watson Estimator

The *Nadaraya-Watson estimator* facilitates kernel-based nonparametric estimation of m. For any given point u, the estimator is computed as:

$$\hat{m}NW(u) = \frac{\sum j = 1^{n'} K_h(X_j - u)Y_j}{\sum_{j=1}^{n'} K_h(X_j - u)}$$

where: - K_h represents the kernel function, assigning weights based on proximity to the target u, - h is the bandwidth, which controls the smoothness of the estimate, - X_j are selected lagged values, and Y_j are the corresponding observed values.

Kernel and Bandwidth Selection

The bandwidth matrix H is generally diagonal, taking the form $H = \text{diag}(h, h, \ldots, h)$, where h is a scalar bandwidth parameter. A symmetric kernel function (such as Gaussian or Epanechnikov) is often used, offering adaptability in smoothing based on the local density of data points.

```r
# Data loading
rd_expenditure <- read.csv("R&D_expenditure.csv")
patent_applications <- read.csv("Patent_applications.csv")
tech_exports <- read.csv("High_technology_exports.csv")

# Define preprocessing functions as you provided
remove_na_rows_and_columns <- function(data) {
  data <- data[rowSums(is.na(data[,-1])) < 0.5 * (ncol(data)-1), ]
  data <- data[, colSums(is.na(data)) < 0.5 * nrow(data)]
  return(data)
}

# Remove rows and columns with >50% NA values
na_r_rdexp = remove_na_rows_and_columns(rd_expenditure)
na_r_pa = remove_na_rows_and_columns(patent_applications)
na_r_te = remove_na_rows_and_columns(tech_exports)

# Intersect countries across datasets
common_countries <- Reduce(intersect, list(
  na_r_rdexp$Country.Name,
  na_r_pa$Country.Name,
  na_r_te$Country.Name
))

# Filter data based on common countries
rdexp <- na_r_rdexp[na_r_rdexp$Country.Name %in% common_countries, ]
pa <- na_r_pa[na_r_pa$Country.Name %in% common_countries, ]
hte <- na_r_te[na_r_te$Country.Name %in% common_countries, ]

# Interpolate missing values
interpolate_data <- function(data) {
  data[] <- lapply(data[], function(col) {
    return(na.approx(col, na.rm = FALSE))
  })
  return(data)
}

# Apply interpolation
rdexpi = cbind(rdexp[1], as.data.frame(t(interpolate_data(as.data.frame(t(rdexp[-1]))))))
pai = cbind(rdexp[1], as.data.frame(t(interpolate_data(as.data.frame(t(pa[-1]))))))
htei = cbind(rdexp[1], as.data.frame(t(interpolate_data(as.data.frame(t(hte[-1]))))))

# Gaussian Kernel Function
gaussian_kernel <- function(x, bandwidth) {
  return((1 / (sqrt(2 * pi) * bandwidth)) * exp(-0.5 * (x / bandwidth)^2))
}

#Transform time series
```

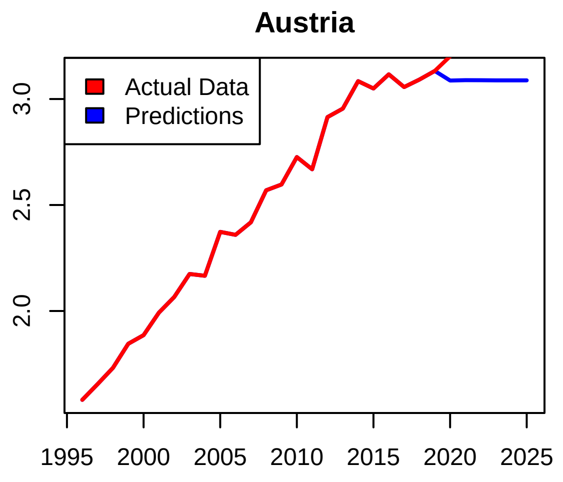
```
transform_series <- function(data, p) {
  n <- length(data)
  if (n <= p) stop("The length of the time series must be greater than p")
  dat <- matrix(0, nrow = n - p, ncol = p + 1)
  for (i in 1:(n - p)) {
    dat[i, ] <- data[i:(i + p)]
  }
  dat <- as.data.frame(dat)
  colnames(dat) <- paste0("Z_", 0:p)
  return(dat)
}

# Compute weights
compute_weight <- function(data, x, h) {
  d <- dim(data)
  if (d[2] - 1 != length(x)) stop("Dimension of x is not compatible")
  weights <- rep(1, d[1])
  for (i in 1:d[1]) {
    u <- as.numeric(data[i, 1:(d[2] - 1)])
    for (j in 1:(d[2] - 1)) {
      k <- gaussian_kernel(u[j] - x[j], h)
      weights[i] <- weights[i] * k
    }
  }
  return(weights)
}


trdexpi <- rdexpi[,1:25]
tpai <- pai[, 1:39]
thtei <- htei[,1:12]

# Nadarya-Watson Forecasting Function
Nadarya_watson <- function(country, h, p, t)
{
  data1 <- trdexpi[trdexpi$Country.Name == country, -1]
  data2 <- tpai[tpai$Country.Name == country, -1]
  data3 <- thtei[thtei$Country.Name == country, -1]

  data1 <- as.numeric(unlist(data1))
  data2 <- as.numeric(unlist(data2))
  data3 <- as.numeric(unlist(data3))

  dat1 <- transform_series(data1, p)
  dat2 <- transform_series(data2, p)
  dat3 <- transform_series(data3, p)

  forecast_data1 <- matrix(0, nrow = t, ncol = p + 1)
  forecast_data2 <- matrix(0, nrow = t, ncol = p + 1)
  forecast_data3 <- matrix(0, nrow = t, ncol = p + 1)

  for (i in 1:t)
  {
```

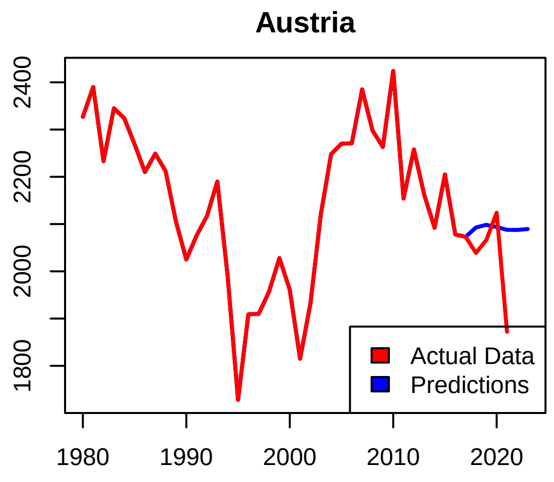
```r
    x1 <- as.numeric(dat1[(dim(dat1)[1]), -1])
    x2 <- as.numeric(dat2[(dim(dat2)[1]), -1])
    x3 <- as.numeric(dat3[(dim(dat3)[1]), -1])

    weights1 <- compute_weight(dat1, x1, h[1])
    weights2 <- compute_weight(dat2, x2, h[2])
    weights3 <- compute_weight(dat3, x3, h[3])

    forecast_data1[i, 1:p] <- x1
    forecast_data1[i, p + 1] <- sum(weights1 * dat1[, p + 1])/ sum(weights1)
    dat1 <- rbind(dat1, forecast_data1[i, ])

    forecast_data2[i, 1:p] <- x2
    forecast_data2[i, p + 1] <- sum(weights2 * dat2[, p + 1]) / sum(weights2)
    dat2 <- rbind(dat2, forecast_data2[i, ])

    forecast_data3[i, 1:p] <- x3
    forecast_data3[i, p + 1] <- sum(weights3 * dat3[, p + 1]) / sum(weights3)
    dat3 <- rbind(dat3, forecast_data3[i, ])
  }
  return(list(dat1 = dat1, dat2 = dat2, dat3 = dat3))
}

select_lags <- function(data, max_lags) {
  lags <- NULL
  best_fpe <- Inf
  for (i in 1:max_lags) {
    fpe <- final_prediction_error(data, i)
    if (fpe < best_fpe) {
      best_fpe <- fpe
      lags <- i
    }
  }
  return(lags)
}

# Final Prediction Error (FPE) Calculation
final_prediction_error <- function(data, lags) {
  # Calculate FPE using cross-validation
  model <- arima(data, order = c(lags, 0, 0))
  residuals <- model$residuals
  n <- length(residuals)
  fpe <- mean(residuals^2) * (n + lags) / (n - lags)
  return(fpe)
}

# How to use.....
## Lag for most of the countries is 1 so, we are using lag=1 by default in our functions
# e.g:
select_lags(as.numeric(as.vector(rdexpi[rdexpi$Country.Name == "Bulgaria",-1])),5)
```

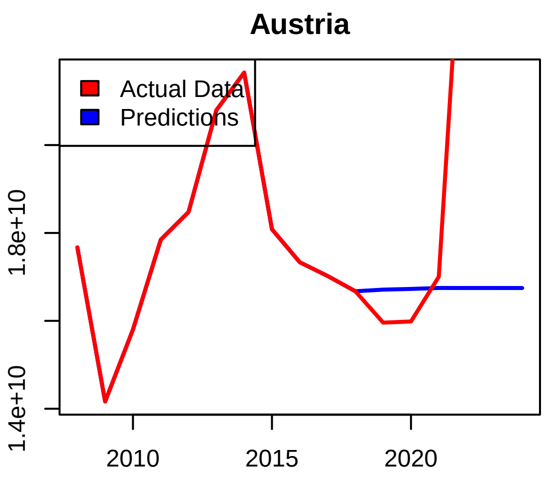
```
## [1] 1
```

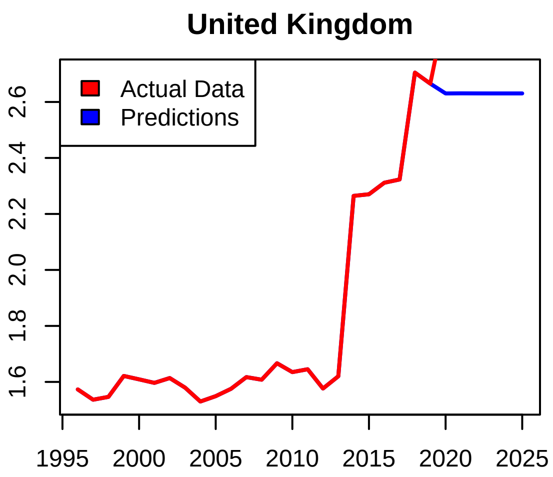
```r
k = Nadarya_watson("Bulgaria",c(0.1,10,2248380615), 3, 10)
countries <- c("Austria", "United Kingdom", "Germany", "Japan", "United States")
```

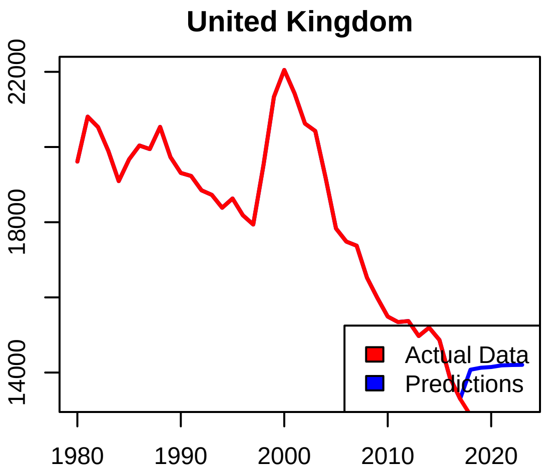
```r
par(mfrow = c(1, 2), mar = c(2.5, 2.5, 2.5, 2.5), cex = 0.75)
for(i in countries)
{
  h = vector(length=3)
  h[1] = density(na.omit(as.numeric(as.vector(rdexpi[rdexpi$Country.Name == i,-1]))))$bw
  h[2] = density(na.omit(as.numeric(as.vector(pai[pai$Country.Name == i,-1]))))$bw
  h[3] = density(na.omit(as.numeric(as.vector(htei[htei$Country.Name == i, -1]))))$bw
  k = Nadarya_watson(i,h,p=3, 6)

  d1 <- c(k$dat1[1,1],k$dat1[1,2],k$dat1[1,3],k$dat1[,4])
  d2 <- na.omit(as.numeric(as.vector(rdexpi[rdexpi$Country.Name == i,-1])))
  y1 <- 1996:2021
  y2 <- 1980:2021
  y3 <- 2008:2022
  plot(y=d1,x=1996:2025,"l",col="blue",lwd=2,
    xlab = "Years",ylab = "R&D Expenditure", main = i)
  lines(x=y1,y=d2,"l",col="red",lwd=2)
  legend("topleft",legend=c("Actual Data","Predictions"),fill = c("red","blue"))

  d1 <- c(k$dat2[1,1],k$dat2[1,2],k$dat2[1,3],k$dat2[,4])
  d2 <- na.omit(as.numeric(as.vector(pai[pai$Country.Name == i,-1])))
  plot(y=d1,x=1980:2023,"l",col="blue",lwd=2,
    xlab = "Years",ylab = "Patent Filings", main = i)
  lines(x=y2,y=d2,"l",col="red",lwd=2)
  legend("bottomright",legend=c("Actual Data","Predictions"),fill = c("red","blue"))

  d1 <- c(k$dat3[1,1],k$dat3[1,2],k$dat3[1,3],k$dat3[,4])
  d2 <- na.omit(as.numeric(as.vector(htei[htei$Country.Name == i,-1])))
  plot(y=d1,x=2008:2024,"l",col="blue",lwd=2,
    xlab = "Years",ylab = "High Technology Export", main = i)
  lines(x=y3,y=d2,"l",col="red",lwd=2)
  legend("topleft",legend=c("Actual Data","Predictions"),fill = c("red","blue"))

}
```

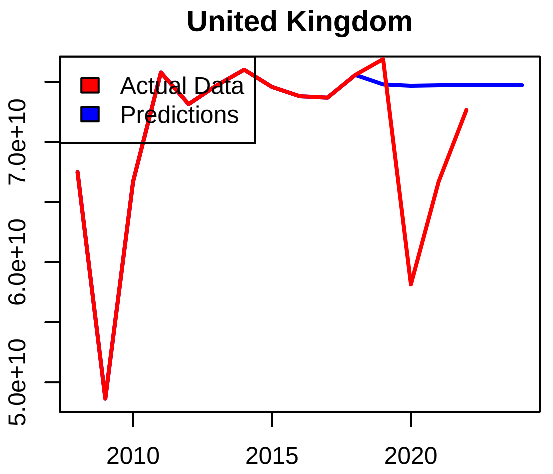


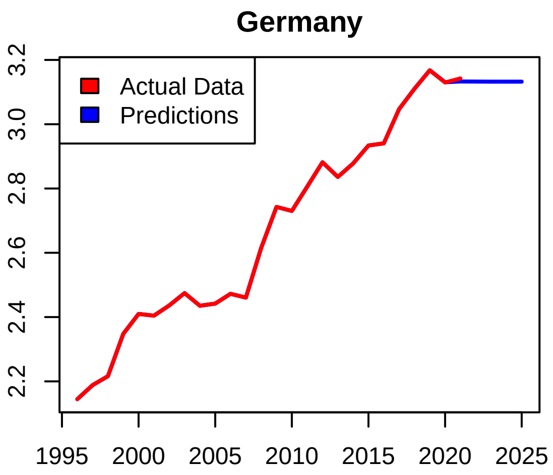


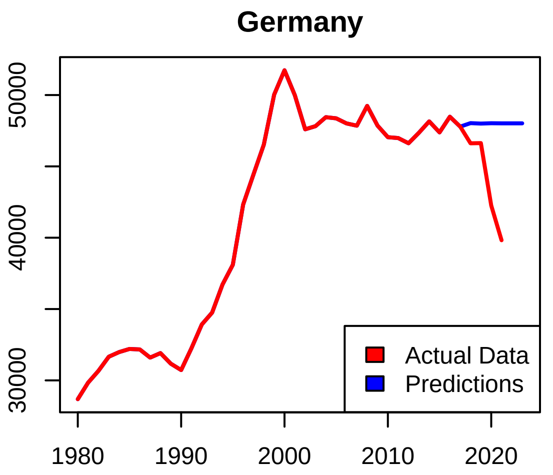

Austria
Actual Data
Predictions
1.4e+10
1.8e+10
2010
2015
2020

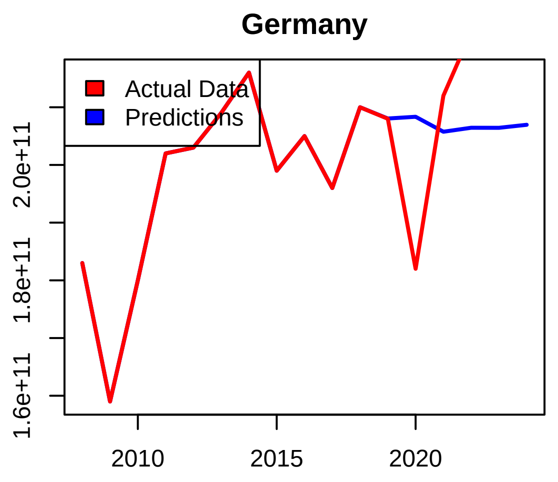

United Kingdom
Actual Data
Predictions
1.6
1.8
2.0
2.2
2.4
2.6
1995
2000
2005
2010
2015
2020
2025

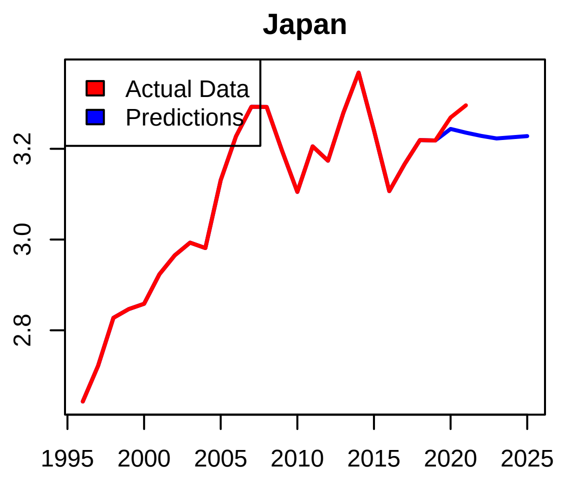

United Kingdom
Actual Data
Predictions
14000
18000
22000
1980
1990
2000
2010
2020

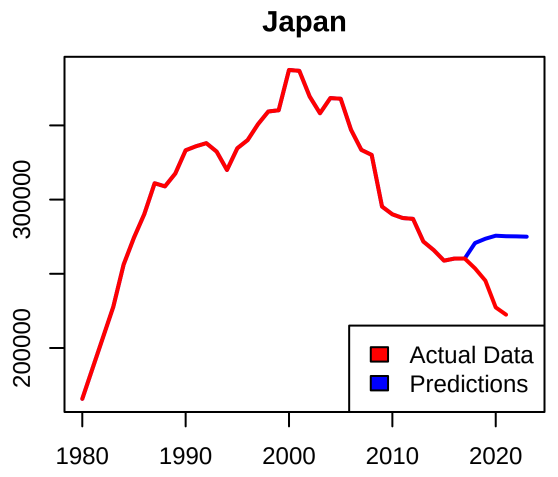

United Kingdom
Actual Data
Predictions
5.0e+10
6.0e+10
7.0e+10
2010
2015
2020

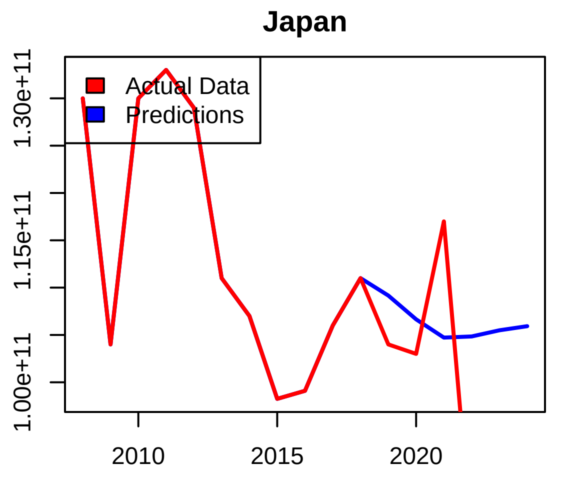

Germany
Actual Data
Predictions
2.2
2.4
2.6
2.8
3.0
3.2
1995
2000
2005
2010
2015
2020
2025

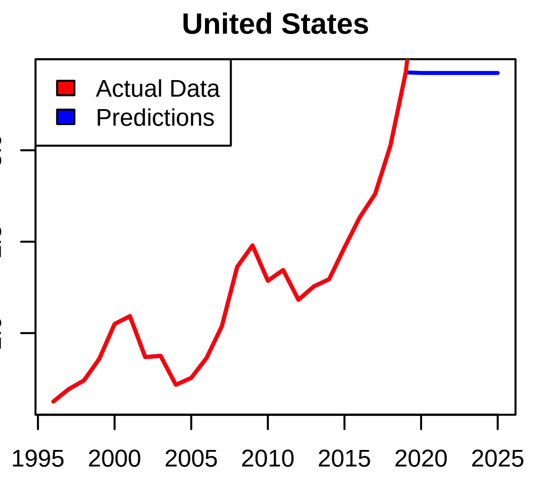

Germany
Actual Data
Predictions
30000
40000
50000
1980
1990
2000
2010
2020

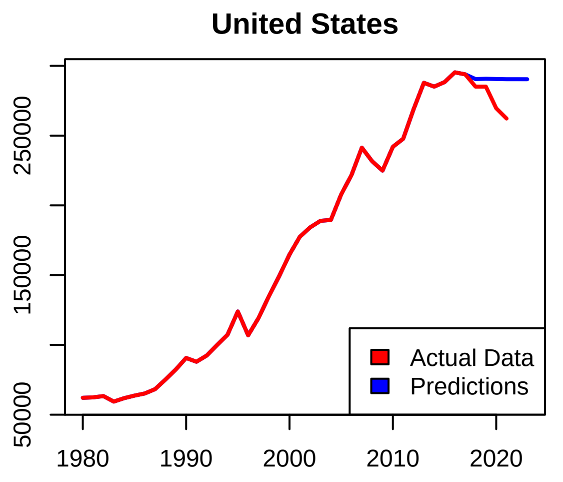

Germany
Actual Data
Predictions
1.6e+11
1.8e+11
2.0e+11
2010
2015
2020

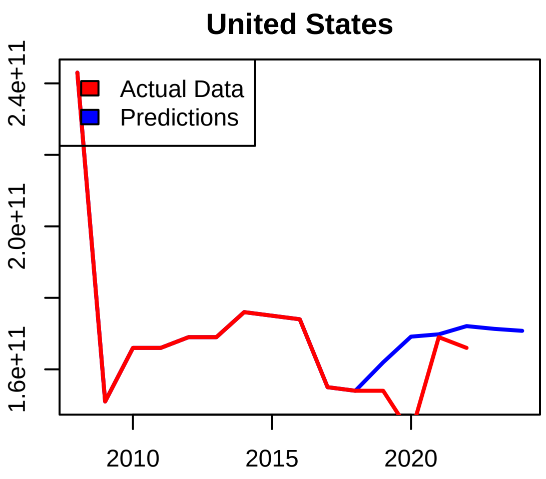

Japan
Actual Data
Predictions
2.8
3.0
3.2
1995
2000
2005
2010
2015
2020
2025


Japan
Actual Data
Predictions
200000
300000
1980
1990
2000
2010
2020


Japan
Actual Data
Predictions
1.00e+11
1.15e+11
1.30e+11
2010
2015
2020


United States
Actual Data
Predictions
2.6
2.8
3.0
1995
2000
2005
2010
2015
2020
2025


United States
Actual Data
Predictions
50000
150000
250000
1980
1990
2000
2010
2020


United States
Actual Data
Predictions
2.4e+11
2.0e+11
1.6e+11
2010
2015
2020