Project 2

Dashboard and Dashboard Tools

Data

The dataset for this project was compiled from the gapminder website Gapminder Fasttrack Github page. [1]

Datasets

The datasets downloaded were measures of Carbon Footprint per Capita by Country, Gross Domestic Product per Capita by Country and Gini Coefficient by country from the last several decades, and beyond in some cases. For the analysis the data from 1980 to 2020 was taken. This was due to the fact that all the data seemed to be reasonably reliable across all datasets over that period. Gapminder fills in missing values and it is not clear exactly how each datapoint is filled (see documentation here.[2] The data is only suitable for observing trends and examining relationships but is not intended for accurate numerical calculations. Below is a description of each dataset.

Carbon Footprint per Capita

The carbon footprint per capita is calculated using the Carbon Dioxide Equivalent framework put forward by the Intergovernmental Panel on Climate Change. [3] [4]. It is measured in million metric tonnes of carbon dioxide equivalents and includes all metrics that impact Global Warming (converting other gases into equivalent CO_2). The total value for a year for a country is divided by the population.

Gross Domestic Product (GDP) per Capita

The GDP per capita is the total value of everything produced by a country over a year divided by the population. For this table, data has been sourced (by Gapminder) from the World Bank, along with various other sources. Changes in country borders are ignored and all values are adjusted for cost of living and inflation.[5]

Gini Coefficient

The Gini coefficient is the proportion of total money earned by everyone in a country that is earned by the bottom 50% of earners over a year. It is between 0 and 1 and if the number is high, it means that relative inequality is high, i.e the wealthier people have a greater share of the money. The dataset goes back to 1800, however much of this is projected back by time-series.[6] More recent data is sourced from POVCALNET who provide data based on the World Bank. This is much more reliable.

Analytics

For the analysis, visualisations and relevant calculations were performed in both R Shiny and Power BI. For each set of visualisations, the difference between the R Shiny and Power BI outputs are compared and discussed

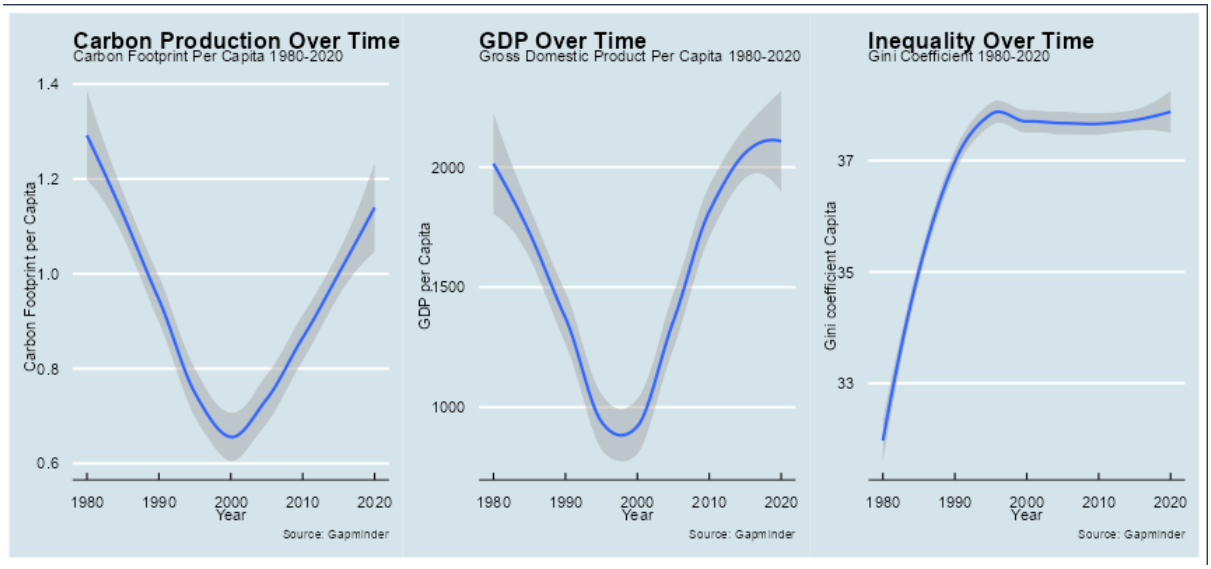
Trends

In order to view the overall trend of each metric over time, three trend graphs were produced.

R Shiny

In R Shiny, the GGplot2 package was used to create the graphics. The graphs produced are shown below.

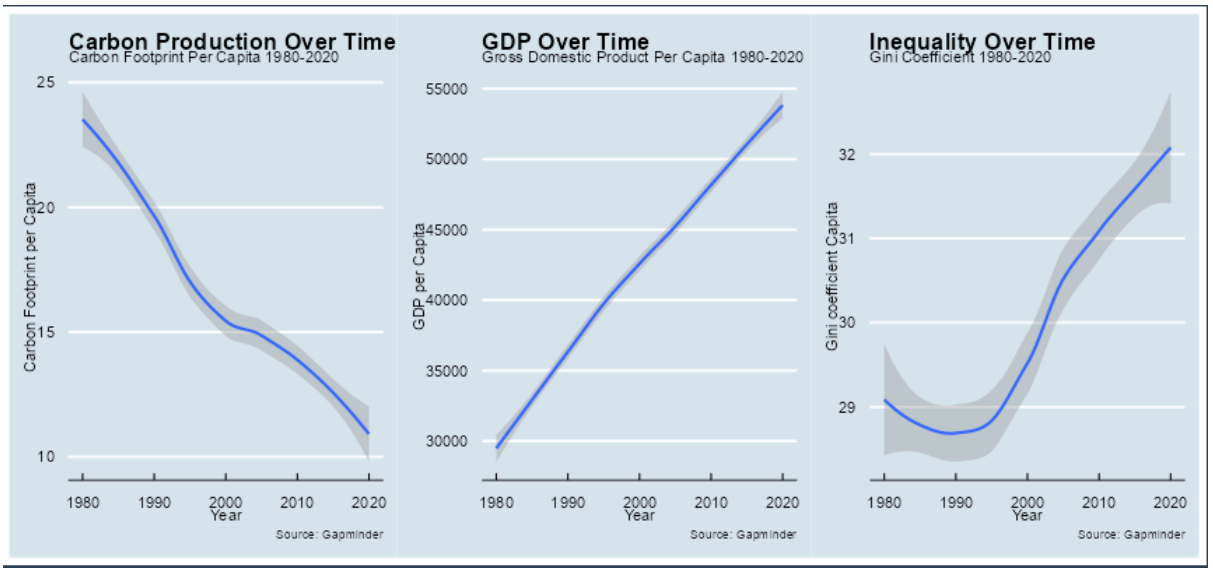
Trends for Afghanistan



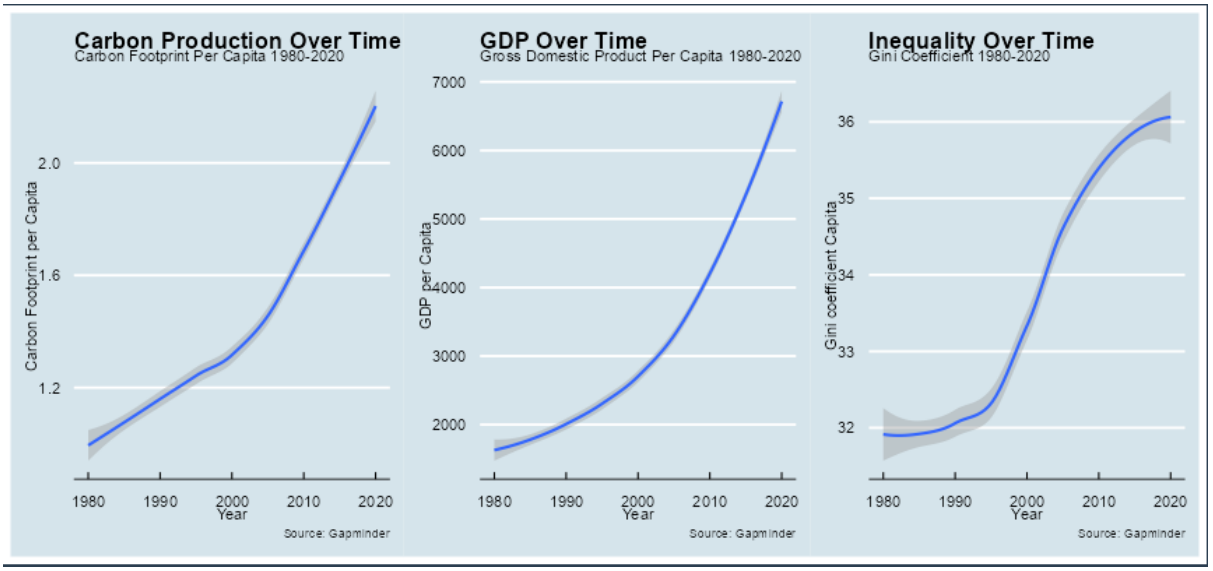
GGpot2 was chosen as the geom_smooth() function emphasised the trend by “smoothing” the curves. This made it very easy to see the general trends over time. Due to the varying quality in the data, a trend chart could have been misleading. Also, a package called “Patchwork” that works with “GGplot2” made the arrangements of graphs on the Shiny dashboard extremely intuitive and quick. The aesthetic was produced using the “theme_economist()” function in the “ggthemes” library. This gave an aesthetic based on “The Economist” magazine. These trends summarise the data well and also provide insight into the relationship between Carbon footprint and GDP for different countries.

The following graphs showed an interesting contrast.

Trends for Germany in R Shiny



Trends for India in R Shiny

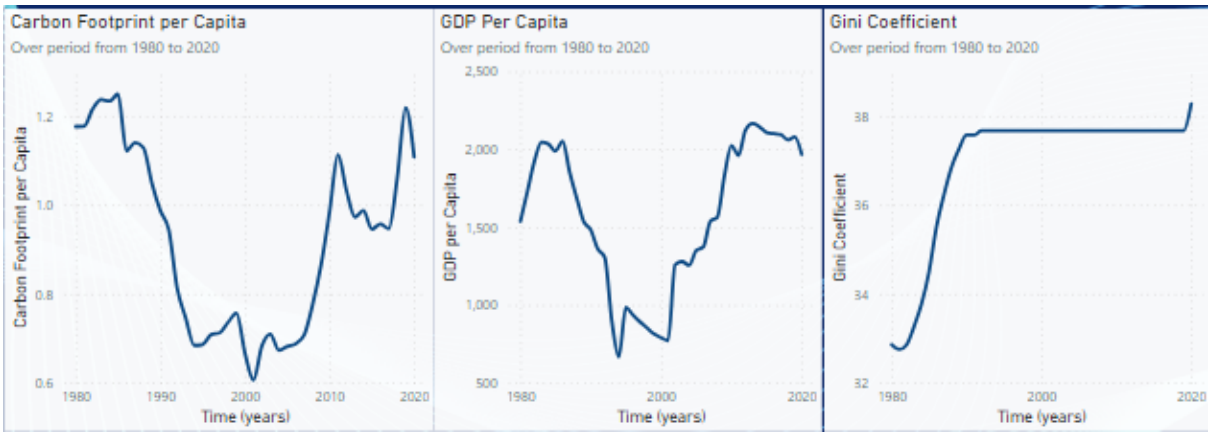


It was observed that Germany appears to have an inverse relationship between GDP and Carbon Footprint while India has a strong positive relationship. This indicates that there may be relationships between economic growth and climate change that vary significantly between countries.

Power BI

The trend graphs were very easily produced in Power BI. As of July 2023, smoothing trend graphs is supported. However the aesthetic is quite different to geom_smooth()

Smooth trends in Power BI for Afghanistan



This can be compared to the above plot from R Shiny. There was much more detail shown by Power BI but it is questionable whether the details are meaningful given the data.

Discussions

In producing the trend graphs, both platforms were more than adequate and were quick (although Power BI was much quicker). Finding a stylish format was easier in R Shiny due to the use of the “ggtheme” library while the default settings in Power BI were aesthetically strong.

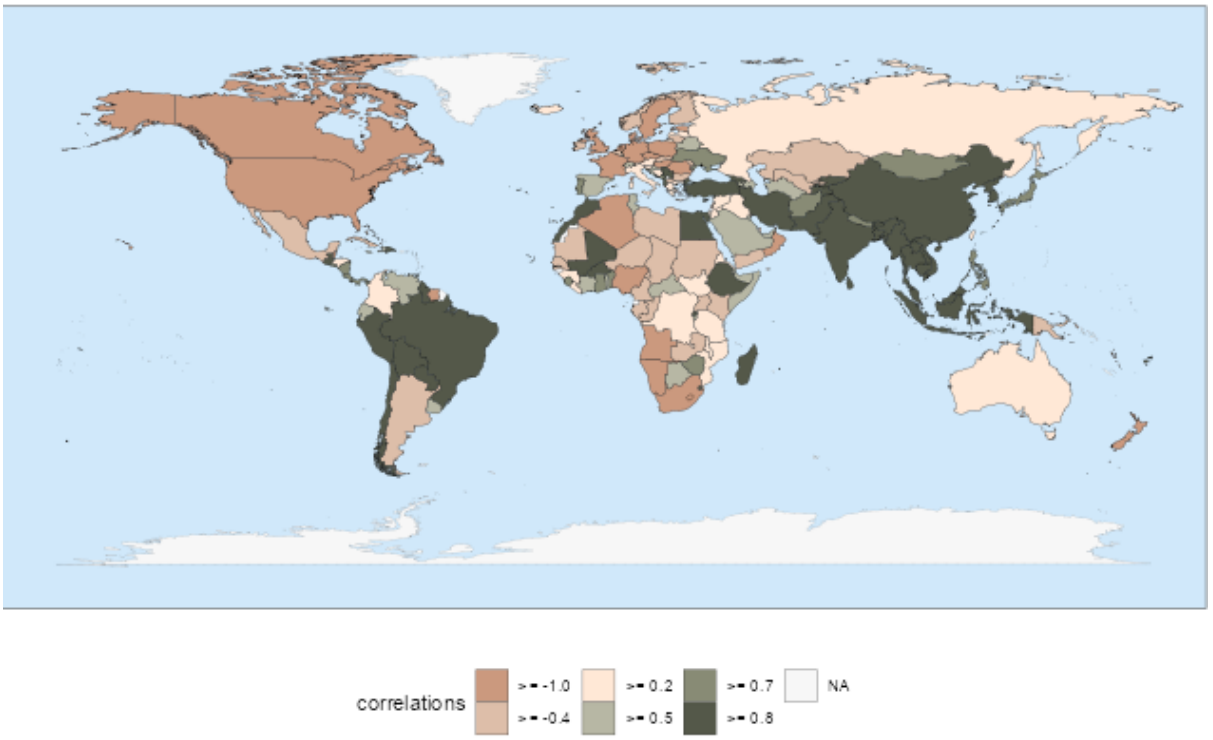
Correlations

From studying the trends produced in the Trends section, it was interesting to explore which countries tend to have stronger correlations between GDP and Carbon footprint (both negative and positive). For this, the correlation coefficient for each country was calculated over the time frame. For ease of viewing, the data was visualised as a coloured map of the world where the correlation coefficient was represented by the shade of colour.

R Shiny

In R Shiny, the function “quick_map()” made this extremely easy. The “countries” library recognised the country abbreviations and converted the abbreviations into full names. A data frame was created with all the correlation coefficients for each country and “quick_map()” was able to colour every country according to its respective correlation value using a default colour scheme. Dark colours are strong positive correlations, neutral colours are weak correlations and stronger, lighter colours are strong negative correlations.

R Shiny Correlation Map

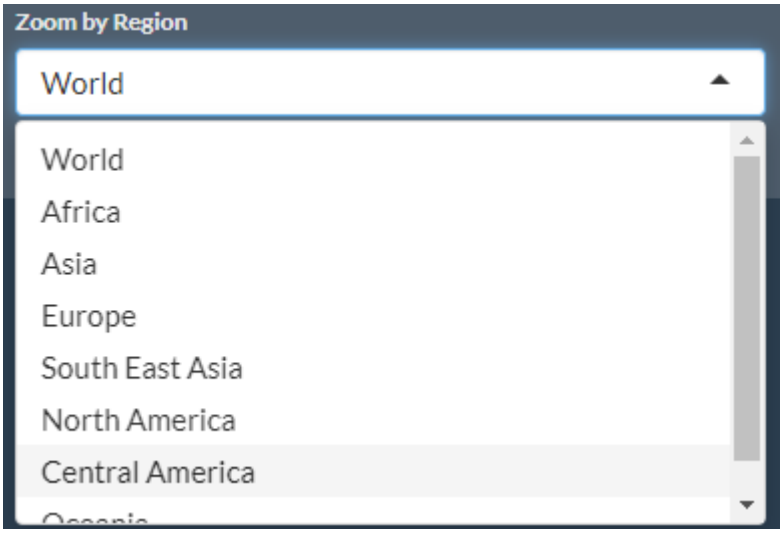


From the plot it was easily seen that certain areas of the globe seem to have strong positive correlations between GDP and Carbon Footprint and others appear to have strong negative correlations.

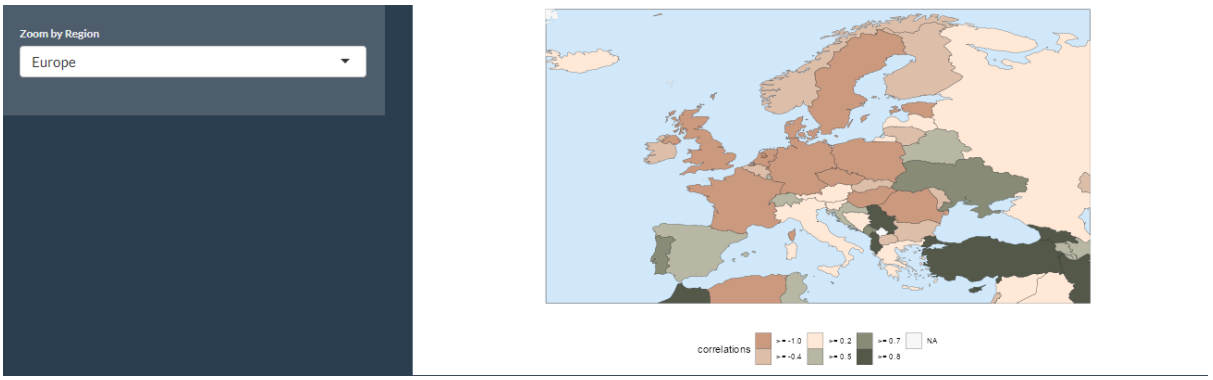
Zooming in R Shiny quick_map()

Zooming in on certain regions was facilitated very easily with the zoom keyword in quick_map(). However, only certain geographical regions were included in the list, as seen below.

List of regions in quick_map()



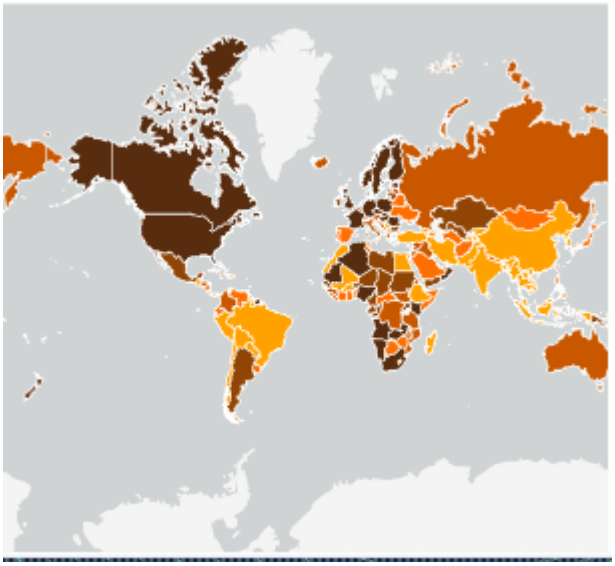
Zooming in quick_map() - Europe



Power BI

While it is possible to calculate correlation coefficients in Power BI, it was found to be easier to reuse R code for filtering the data by country, calculating the correlations and creating the dataframe in the Power Query Editor. Once the data frame was created, simply clicking on the “ArcGis Maps” visual and dragging in the countries as locations and correlations as colours produced a very nice visual. Hovering over countries displayed a “tooltip” with the country name and the correlation coefficient.

Power BI Correlation Map

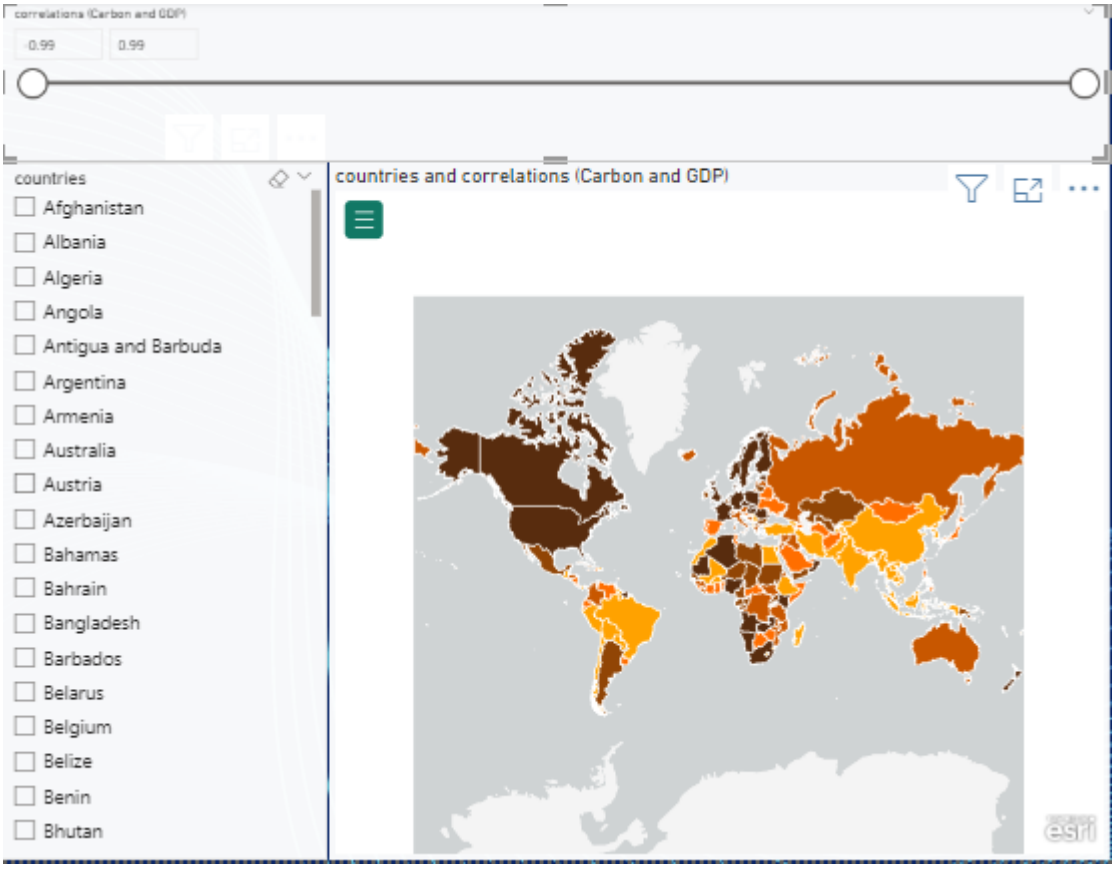


I could not find a way of editing the colour scheme in the Power BI documentation for ArcGis [6]. The dark colours represent strong negative correlations while bright yellow colours represent strong positive correlations.Brown colours represent weak correlations.

Slicing in Power BI

Rather than zooming in with a dropdown like in R Shiny, which can be done with the mouse in Power Bi, the slicer functionality was chosen. A slicer with the list of countries along with a separate slicer to filter certain ranges of correlation coefficient were added to the tab.

Slicing the Correlation Map



The map was quite slow to load and appears to have reloaded after each use of a filter.

Discussion

The ArcGis Map in Power BI was much more informative and interactive than the quickmap but was also much slower to load. Using R script with the Power Query Editor made the analytical calculations very quick and efficient. On balance, Power BI was a much more efficient and effective platform for displaying this visual.

Forecasting

It was found through exploring the data, that many of the countries had some relationships between the three variables (Carbon Footprint, GDP and Gini). While many appeared linear, a linear regression model turned out to be inappropriate due to autocorrelation in the residuals for large portions of the countries. Subsequently a Time Series forecasting model called Vector Autoregression (VAR) was adopted to produce forecasts for all three variables based on their past values as well as the values of each other. A brief summary of some

diagnostic tests to check if the model was valid was also provided with the plots of the forecasts.

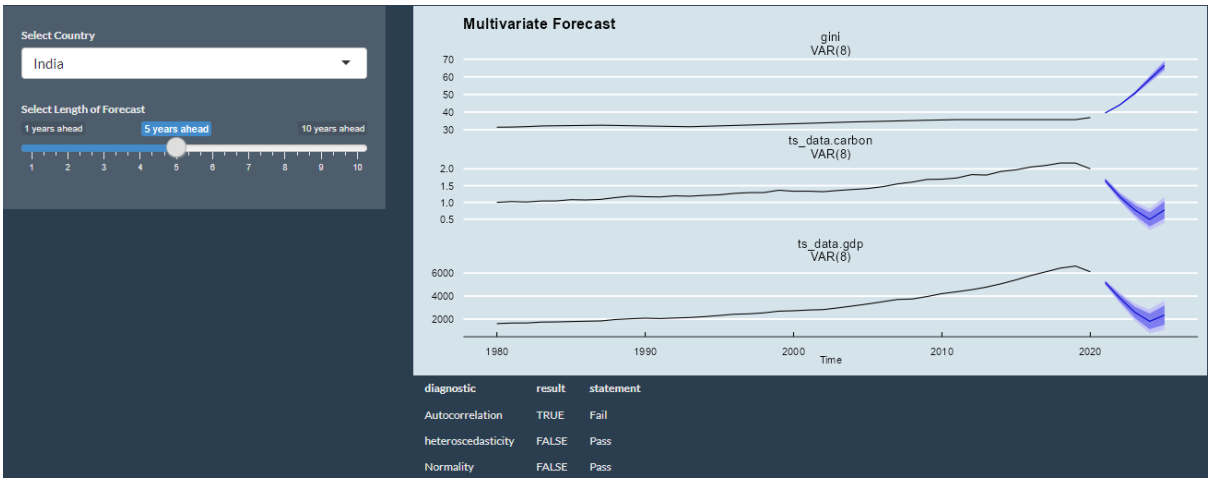
R Shiny

In R Shiny, the forecasting was produced using the “vars” package. The Akaike information criterion (AIC) was used as the criteria for selecting the optimal lag for the multivariate model. The forecasts were then plotted using the “autoplot()” function which allowed for the use of ggplot2 editing. The diagnostics were printed to the screen from a dataframe using the renderText() function in Shiny.

Comparison of Germany and India

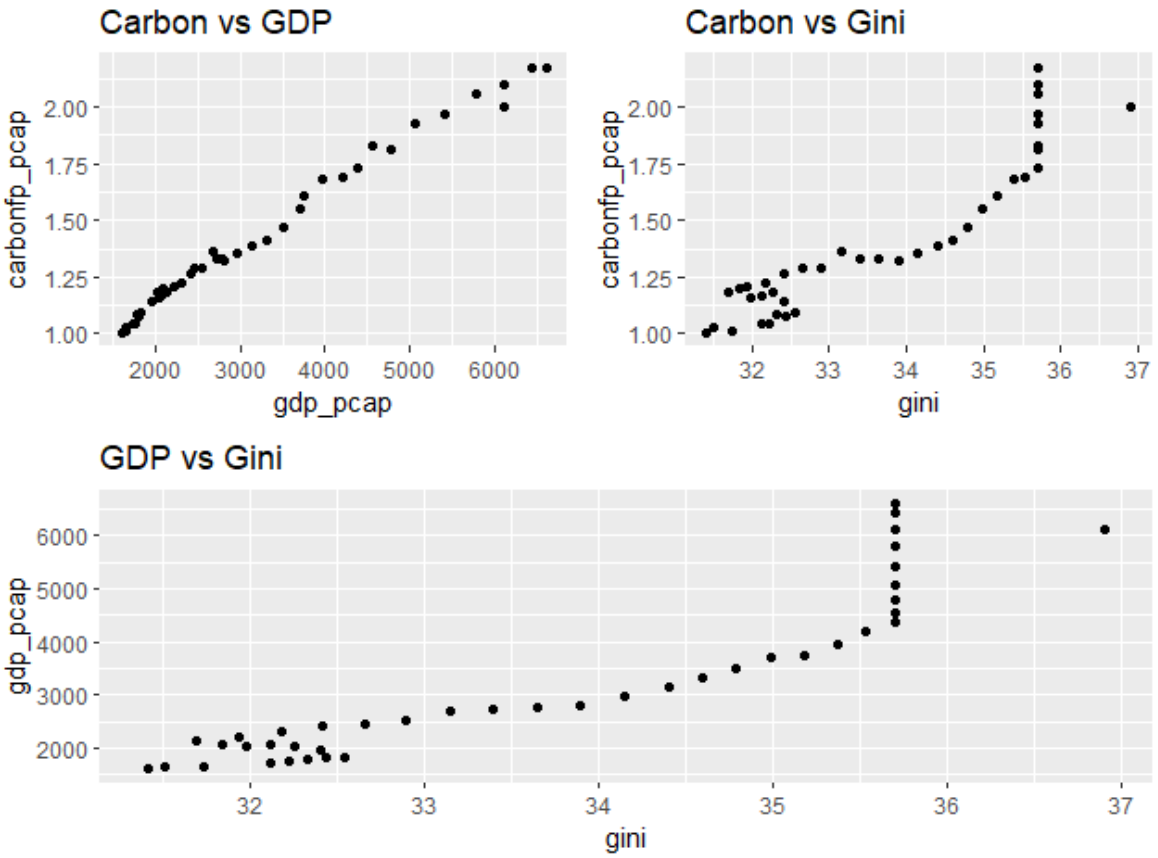
In the Trends section above (link) it was suggested that India and Germany had particularly strong relationships between GDP and Carbon Footprint. The forecasts produced strengthens this argument. It should be noted that, for India, the residuals of the model failed a test for autocorrelation (as shown on screen).

Multivariate Forecast for India



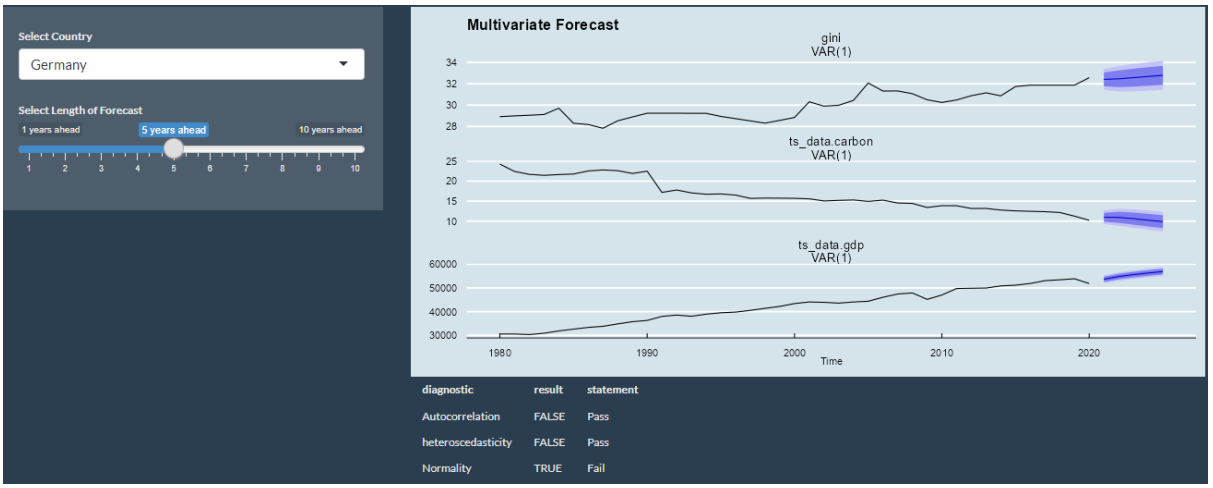
Here, it is seen that as GDP and Carbon footprint go down over the forecasted period, the Gini Coefficient appears to go up. The scatter plots below show the relationship between the different variables. A linear model was built for the relationship between gini and the other two variables and a significant interaction effect was found between carbon and gdp. This might explain the dramatic increase in inequality predicted.

Scatter Plots for relationships between variables for India



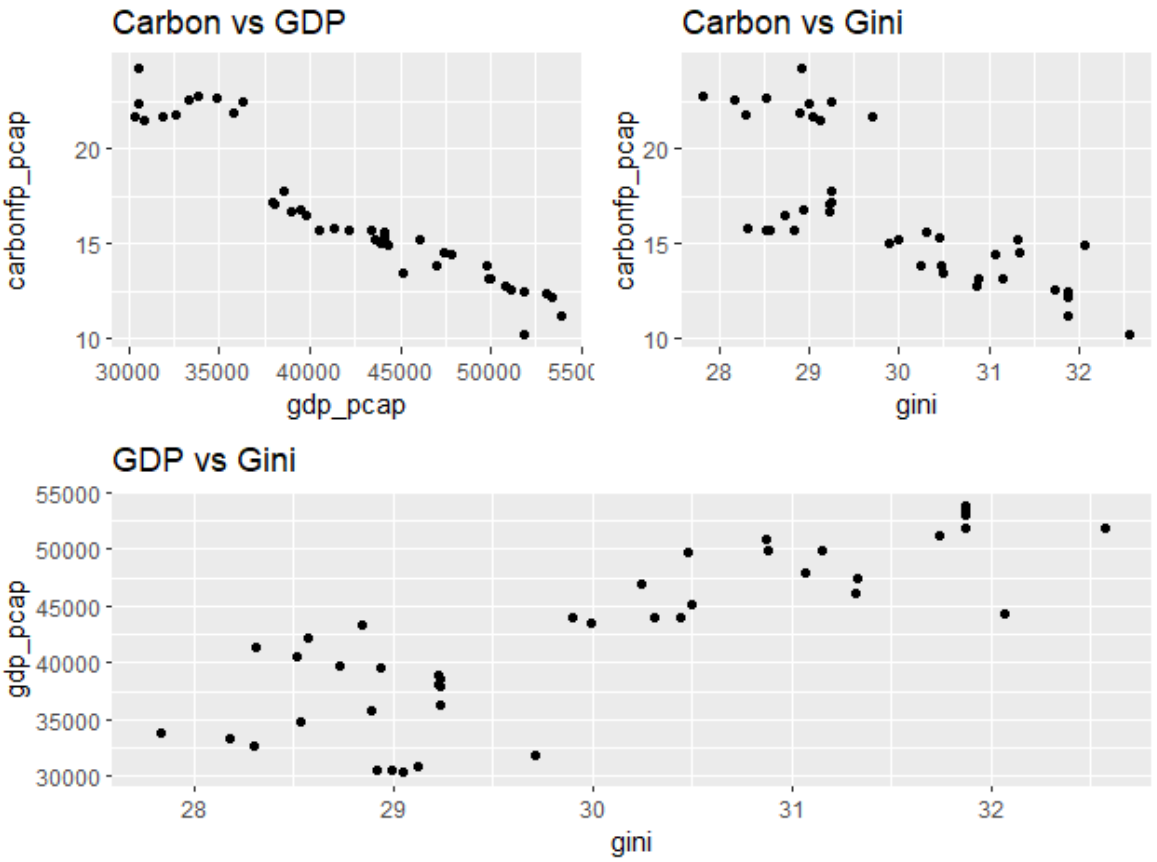
While not linear, the relationships between Gini and the other variables are not random. For many data points, Gini seems to increase as gdp and carbon increase. The increase in Gini while the other variables are projected to go down is concerning.

Multivariate Forecast for Germany



In the graphs above it was seen that carbon footprint is set to trend downwards as GDP goes up. Inequality appears to be trending up slightly.

Scatter Plots for relationships between variables for Germany

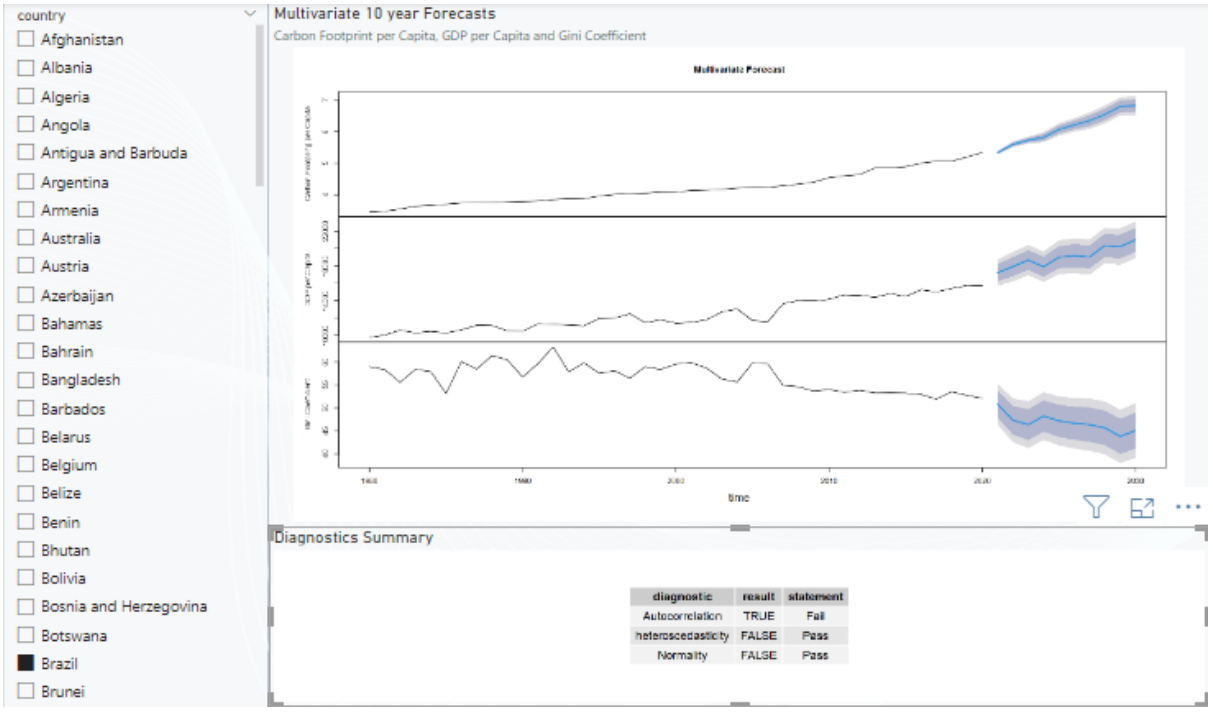


In the scatter plots examining the relationship between all three variables, all three showed some pattern. Carbon footprint seems to have a negative correlation with both GDP and Inequality while GDP and Inequality seem to have a positive relationship. Carbon footprint is associated with a drop in Gini while GDP is associated with an increase.

Power BI

For the corresponding visualisation in Power BI, R code was used to produce the forecast as the desktop version available did not support forecasting. For this reason the visual is very similar. Due to technical difficulties using ggplot2 in Power BI, it was not possible to use the autoplot function which meant that adding a style theme (from ggthemes) was not straightforward.

Multivariate Forecast for Brazil in Power BI



While it was likely possible to use a slicer to interact with the forecasting script and treat the chosen number as a variable in the code, due to time constraints, this was not completed. This meant that all the forecasts were set to 10 years in the future (as opposed to a user-selected number of years in R).

Comparison of Platforms

Data Compiling and Preprocessing

Compiling Data

For initial data cleaning and preparation, there was no great difference noted between the platforms. Having compiled the datasets into one table in R, it was more efficient to export the csv file and read it directly to Power BI as one table. It would have been equally straightforward to start with Power BI and do the reverse.

Statistical Calculations

For both the correlation map and the forecast, special statistical calculations were needed. For all of the visuals produced, slicing in Power BI was used in place of a drop down list with filtering code in R Shiny. The ability to filter for a given country was much easier in R. Slicing in Power BI was very useful for dynamic interactive visuals. However, in order to produce the correlations, for example, in the Correlations Map, the filtering had to be done before making the table with all the coefficients. While it was possible to get the correlation coefficient metrics in Power BI, it was much more straightforward to use R Script in the Power Query

Editor. This allowed for the full use of Power BI visuals. Producing the plot using R script in Power BI results in reduced functionality of the visual.

In the case of the forecasting visuals, R was the only tool able to complete the task for the version of Power BI that was accessible. In theory, the forecasting data for all countries could have been produced in a table in the Power Query Editor and plotted using Power BI visuals. However it was much easier to compute the forecasts in Power BI as an R Visualisation.

Visualisations

In R Shiny, the package “ggplot2”, along with the “patchwork” package, was used for the trend and the forecast visualisations. This allowed for very efficient styling. The “theme_economist()” function easily provided elegant charts and consistent styling between graphs. With more time, it would have been nice to look more at “ggmap” for the correlation map visual to provide more consistency. The default charts in Power BI were excellent and only required some basic editing of titles, subtitles and axis. The ability to hover over regions in Power BI and see information was very powerful. This could have been achieved in R Shiny using Plotly, however, it was not part of the initial intention and the goal of the visualisations was to show general trends rather than accurate numerical information.

The ArcGis tool in Power BI was very easy to use and produced a very nice visual with the correlation data. The ability to use slicers made it very interactive and the process of designing it was extremely quick. That said, the map takes quite some time to load when reacting to a change in the slicers. In this regard, Power BI (with R in the Power Query) was the best option found. The “quick_map()” function was easy to use but ultimately limited with the zoom function.

For forecasting, R was used to produce the visualisation in both platforms. In R Shiny, the “autoplot()” function produced a “ggplot2” graphic while this proved problematic in Power BI. It hence produced a more basic plot that wasn’t so easy to edit. For the forecasting graphic, R Shiny was much more appropriate and producing it in Power BI involved a very similar script anyway. In this case, R Shiny was more appropriate for the task at hand.

Aesthetics

While the default aesthetics in Power BI were excellent and intuitive, the use of “shinytheme()” function from the “shinythemes” package, along with the above mentioned “theme_economist()” allowed for simple and elegant styling in Shiny. The code “shinythemes::themeSelector()” allowed the dashboard to run and have different themes sampled using a drop down list. This made the process of choosing very simple.

Conclusions

Both platforms were adequate for the tasks chosen. R was by far the best equipped for compiling data, and performing statistical calculations to produce new tables. Also, for the forecasting, R had to do the work on both platforms. Balancing quality with efficiency, Power

BI with R code used in the Power Query Editor would be the better choice for this dataset. The editing is much easier and intuitive. In Shiny, the whole code had to be re-run in order to see an edit in the server function while Power BI was dynamic throughout the process.

References

[1] Gapminder.Fasttrack Github
Dataset 1 - Carbon Footprint per Capita by Country:
.https://github.com/open-numbers/ddf--gapminder--fasttrack/blob/master/ddf--datapoints--carbonfp_pcap--by--country--time.csv
Dataset 2 - GDP per Capita by Country:
https://raw.githubusercontent.com/open-numbers/ddf--gapminder--fasttrack/master/ddf--datapoints--gdp_pcap--by--country--time.csv
Dataset 3 - Gini coefficient by Country
https://github.com/open-numbers/ddf--gapminder--fasttrack/blob/master/ddf--datapoints--gini--by--country--time.csv

[2] Gapminder Documentation. https://www.gapminder.org/data/documentation/

[3] Gapminder Documentation for GDP per Capita.
https://www.gapminder.org/data/documentation/gd001/

[4] Explanation Carbon Dioxide Equivalent, Eurostat
https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Glossary:Carbon_dioxide_equivalent

[5] Gapminder Documentation for Gini Coefficient..
https://www.gapminder.org/data/documentation/gini/

[6] ArcGIS for Power BI Documentation
https://doc.arcgis.com/en/microsoft-365/latest/power-bi/add-a-data-layer-to-the-map.htm

Appendix

[1] R Shiny Code

```
carbon_fp <- read.csv("D:\\Data Science MTU\\Data Visualisation\\Project
2\\ddf--datapoints--carbonfp_pcap--by--country--time.csv", header = T)
gdp <- read.csv("D:\\Data Science MTU\\Data Visualisation\\Project
2\\ddf--datapoints--gdp_pcap--by--country--time.csv",  header = T)
gini <- read.csv("D:\\Data Science MTU\\Data Visualisation\\Project
2\\ddf--datapoints--gini--by--country--time.csv", header = T)

################################################################################
#####
# Cutting data to 1980 - 2020
################################################################################
#####
time_start = as.integer(1980)
time_end = as.integer(2020)

carbon_fp <- subset(carbon_fp, carbon_fp$time >= time_start)
carbon_fp <- subset(carbon_fp, carbon_fp$time <= time_end)

gdp <- subset(gdp, gdp$time >= time_start)
gdp <- subset(gdp, gdp$time <= time_end)

#str(gini)
gini <- subset(gini, time >= time_start)
gini <- subset(gini, time <= time_end)

#View(carbon_fp)
#View(gdp)
data <- carbon_fp
data <- merge(data, gdp, by = c("country","time"))
data <- merge(data, gini, by = c("country","time"))
#View(data)

#write.csv(data, "C:\\Users\\jverl\\Desktop\\Data Science MTU\\Data Visualisation\\Project
2\\data.csv", row.names=FALSE)

#install.packages("countries")
library(countries)
#install.packages("shinythemes")
library(patchwork)
library(tidyverse)
library(shiny)
library(shinythemes)
library(ggthemes)
```

```
library(forecast)
library(vars)

info <- country_info(data$country)
data$country <- info$name.common
#View(data)

#View(info)

#View(info$latlng)
#library(stringr)



library(tidyverse)

countries <- unique(data$country)
num_countries <- length(countries)

correlations <- replicate(num_countries,NA)

for (i in 1:num_countries){
  place <- countries[i]
  correlations[i] <- cor(data$carbonfp_pcap[data$country==place],
                         data$gdp_pcap[data$country==place])
}
correlation_df <-data.frame(countries,correlations)

correlation_df$countries <- as.factor(correlation_df$countries)







# Define UI for app that draws a histogram ----
ui <- fluidPage(
  #shinythemes::themeSelector(),  # <--- Add this somewhere in the UI
  theme = shinytheme("superhero"),
          navbarPage("Carbon Footprint Analysis",

                     tabPanel("Trends",
                     sidebarLayout(
                     sidebarPanel(
                     selectInput(inputId = "place",
                                 label="Select Country",
                                 choices=unique(data$country),
                                 selected="Afghanistan"
```

```
                )
        ),
        mainPanel(
        plotOutput(outputId =  "trends")
        )
)
),
tabPanel("Correlations",
        sidebarLayout(
        sidebarPanel(
        selectInput(inputId = "zoom",
                label="Zoom by Region",
                choices=c("World" = "World",
                        "Africa" = "Africa",
                        "Asia" = "Asia",
                        "Europe" = "Europe",
                        "South East Asia" = "SEAsia",
                        "North America" = "NAmerica",
                        "Central America" = "SAmerica",
                        "South America" = "SAmerica",
                        "Oceania" = "Oceania"
                        ),
                selected="World")
        ),
        mainPanel(
        plotOutput("map")
        )
        )
        ),      tabPanel("Forecast",
                        sidebarLayout(
                        sidebarPanel(
                        selectInput(inputId = "region",
                                label="Select Country",
                                choices=unique(data$country),
                                selected="Afghanistan"
                        ),
                        sliderInput(inputId = "future",
                                label="Select Length of Forecast",
                                min = 1,
                                max = 10,
                                step = T,
                                post = " years ahead",
                                value = 5
                        )

                        ),
                        mainPanel(
                        plotOutput("forecast"),
```

```
                                tableOutput("diagnostic")
                                )
                                )
                    )
                    )
        )


# Define server logic required to draw a histogram ----
server <- function(input, output) {

  output$trends <- renderPlot({
        data_country <- data %>%
        filter(country==input$place)
        p1 <-ggplot(data=data_country, aes_string(x="time", y="carbonfp_pcap")) +
        geom_smooth() +
        labs(subtitle="Carbon Footprint Per Capita 1980-2020",
        y="Carbon Footprint per Capita",
        x="Year",
        title="Carbon Production Over Time",
        caption = "Source: Gapminder")+
        theme_economist()

        p2 <-ggplot(data=data_country, aes_string(x="time", y="gdp_pcap" )) +
        geom_smooth()+
        labs(subtitle="Gross Domestic Product Per Capita 1980-2020",
        y="GDP per Capita",
        x="Year",
        title="GDP Over Time",
        caption = "Source: Gapminder")+
        theme_economist()


        p3 <-ggplot(data=data_country, aes_string(x="time", y="gini" )) +
        geom_smooth() +
        labs(subtitle="Gini Coefficient 1980-2020",
        y="Gini coefficient Capita",
        x="Year",
        title="Inequality Over Time",
        caption = "Source: Gapminder") +
        theme_economist()


        p <- p1+p2+p3
        p
  })

  output$map <- renderPlot({
```

```
        zoom <- input$zoom
        world <- map_data("world")
        quick_map(correlation_df, plot_col = "correlations", reverse_palette = F, zoom =
zoom)
        })

  output$forecast <- renderPlot({


        ts_plotter <- function(region, h){
        region <- input$region
        model_data <- subset(data, data$country==region)
        carbon <- ts(model_data$carbonfp_pcap, start = c(1980,1), frequency = 1)
        gdp <- ts(model_data$gdp_pcap, start = c(1980,1), frequency = 1)
        gini <- ts(model_data$gini, start = c(1980,1), frequency = 1)

        ts_data <- cbind(carbon, gdp)
        ts_data <- cbind(ts_data, gini)

        aic <- VARselect(ts_data, lag.max=8, type="trend")$selection["AIC(n)"]
        model <- VAR(ts_data, p=aic, type="both")

        future = forecast(model, h = h)
        p <-autoplot(future) + ggtitle("Multivariate Forecast")
        p <- p + theme_economist()
        p


        }
        ts_plotter(input$region, input$future)

  })

  output$diagnostic <- renderTable({

        region <- input$region
        model_data <- subset(data, data$country==region)
        carbon <- ts(model_data$carbonfp_pcap, start = c(1980,1), frequency = 1)
        gdp <- ts(model_data$gdp_pcap, start = c(1980,1), frequency = 1)
        gini <- ts(model_data$gini, start = c(1980,1), frequency = 1)

        ts_data <- cbind(carbon, gdp)
        ts_data <- cbind(ts_data, gini)

        aic <- VARselect(ts_data, lag.max=8, type="trend")$selection["AIC(n)"]
        model <- VAR(ts_data, p=aic, type="both")

##########################################################################
```

```
# Diagnostics

##############################################################################
# True Return always means fail

autocor_check <- function(model){
# Checking for autocorrelation
serial1 <- serial.test(model)
p_autocor <- serial1$serial[[3]]
if(p_autocor < 0.05){
return(TRUE) # autocorrelation present - failed test
}else{
return(FALSE) # autocorrelation not present - passed
}
}

het_ked_check <- function(model){
# Checking for Heteroskedasticity
arch1 <-arch.test(model)
p_hetero <- arch1$arch.mul[[3]]
if(p_hetero < 0.05){
return(TRUE) # Heteroskedasticity is present - fail
}else{
return(FALSE) # No heteroskedasticity is present - pass
}
}

normality_check <- function(model){
norm1 <- normality.test(model)
p_norm <- norm1$jb.mul$JB[[3]] # normal - high p
p_skew <- norm1$jb.mul$Skewness[[3]] # no-skewness - high p
p_kurt <- norm1$jb.mul$Kurtosis[[3]] # no-kurtosis - high p
if(p_norm < 0.05 || p_skew < 0.05 || p_kurt < 0.05){
return(TRUE) # not normal - fail
}else{
return(FALSE) # normal - pass
}
}

diagnostic <- c("Autocorrelation", "heteroscedasticity", "Normality")
result <- c(autocor_check(model),het_ked_check(model),normality_check(model))
statement <- replicate(n=3, NA)

for (i in 1:3){
if(result[i]==T){
statement[i] = "Fail"
} else {
statement[i] = "Pass"
```

```
        }
        }

        diagnostic_df <- cbind(diagnostic, result)
        View(diagnostic_df)
        diagnostic_df <- cbind(diagnostic_df, statement)
        diagnostic_df
  })
}

shinyApp(ui = ui, server = server)

#model_data <- subset(data, data$country=="Germany")
#View(model_data)

#model <- lm(model_data$gini~model_data$carbonfp_pcap+
#         model_data$gdp_pcap +
#         model_data$carbonfp_pcap+
#         model_data$gdp_pcap*model_data$carbonfp_pcap)

#summary(model)
#p_carb_gdp <- ggplot(data = model_data, aes(y=carbonfp_pcap, x = gdp_pcap)) +
#  geom_point() +
#  ggtitle("Carbon vs GDP")
#p_carb_gdp
#p_carb_gini <- ggplot(data = model_data, aes(y=carbonfp_pcap, x = gini)) +
#  geom_point() +
#  ggtitle("Carbon vs Gini")#p_carb_gini
#p_gdp_gini <- ggplot(data = model_data, aes(y=gdp_pcap, x = gini)) +
#  geom_point() +
#  ggtitle("GDP vs Gini")
#p_gdp_gini

#p <- (p_carb_gdp + p_carb_gini) / p_gdp_gini
#p
#plot(model_data$gdp_pcap~model_data$gini)

#model_data <- subset(data, data$country=="Germany")
#View(model_data)

#plot(model_data$carbonfp_pcap~model_data$gdp_pcap)
#plot(model_data$carbonfp_pcap~model_data$gini)
#plot(model_data$gdp_pcap~model_data$gini)
```

[2] R Script for Power BI

[2.1] Diagnostic Summary in Forecast Section

```
# The following code to create a dataframe and remove duplicated rows
is always executed and acts as a preamble for your script:

# dataset <- data.frame(country, time, carbonfp_pcap, gdp_pcap, gini)
# dataset <- unique(dataset)

# Paste or type your script code here:
library(forecast,
lib.loc="C:\\Users\\jverl\\AppData\\Local\\R\\win-library\\4.2")
library(vars,
lib.loc="C:\\Users\\jverl\\AppData\\Local\\R\\win-library\\4.2")
library(gridExtra,
lib.loc="C:\\Users\\jverl\\AppData\\Local\\R\\win-library\\4.2")
data <- dataset

carbon <- ts(data$carbonfp_pcap, start = c(1980,1), frequency = 1)
gdp <- ts(data$gdp_pcap, start = c(1980,1), frequency = 1)
gini <- ts(data$gini, start = c(1980,1), frequency = 1)

ts_data <- cbind(carbon, gdp)
ts_data <- cbind(ts_data, gini)

colnames(ts_data) <- c("Carbon Footpring per Capita", "GDP per Capita",
"Gini Coefficient")

aic <- VARselect(ts_data, lag.max=8, type="trend")$selection["AIC(n)"]
model <- VAR(ts_data, p=aic, type="both")

################################################################################
# Diagnostics
################################################################################
# True Return always means fail

autocor_check <- function(model){
    # Checking for autocorrelation
    serial1 <- serial.test(model)
    p_autocor <- serial1$serial[[3]]
    if(p_autocor < 0.05){
        return(TRUE) # autocorrelation present - failed test
      }else{
```

```
      return(FALSE) # autocorrelation not present - passed
    }
  }

  het_ked_check <- function(model){
    # Checking for Heteroskedasticity
    arch1 <-arch.test(model)
    p_hetero <- arch1$arch.mul[[3]]
    if(p_hetero < 0.05){
      return(TRUE) # Heteroskedasticity is present - fail
    }else{
      return(FALSE) # No heteroskedasticity is present - pass
    }
  }

  normality_check <- function(model){
    norm1 <- normality.test(model)
    p_norm <- norm1$jb.mul$JB[[3]] # normal - high p
    p_skew <- norm1$jb.mul$Skewness[[3]] # no-skewness - high p
    p_kurt <- norm1$jb.mul$Kurtosis[[3]] # no-kurtosis - high p
    if(p_norm < 0.05 || p_skew < 0.05 || p_kurt < 0.05){
      return(TRUE) # not normal - fail
    }else{
      return(FALSE) # normal - pass
    }
  }

  diagnostic <- c("Autocorrelation", "heteroscedasticity",
"Normality")
  result <-
c(autocor_check(model),het_ked_check(model),normality_check(model))
  statement <- replicate(n=3, NA)

  for (i in 1:3){
    if(result[i]==T){
      statement[i] = "Fail"
    } else {
      statement[i] = "Pass"
    }
  }

  diagnostic_df <- cbind(diagnostic, result)
  diagnostic_df <- cbind(diagnostic_df, statement)
  grid.table(diagnostic_df)
```

[2.2] Forecast plot in Power BI

```
# The following code to create a dataframe and remove duplicated rows
is always executed and acts as a preamble for your script:

# dataset <- data.frame(country, time, carbonfp_pcap, gdp_pcap, gini)
# dataset <- unique(dataset)

# Paste or type your script code here:
library(forecast,
lib.loc="C:\\Users\\jverl\\AppData\\Local\\R\\win-library\\4.2")
library(vars,
lib.loc="C:\\Users\\jverl\\AppData\\Local\\R\\win-library\\4.2")

data <- dataset

carbon <- ts(data$carbonfp_pcap, start = c(1980,1), frequency = 1)
gdp <- ts(data$gdp_pcap, start = c(1980,1), frequency = 1)
gini <- ts(data$gini, start = c(1980,1), frequency = 1)

ts_data <- cbind(carbon, gdp)
ts_data <- cbind(ts_data, gini)

colnames(ts_data) <- c("Carbon Footpring per Capita", "GDP per Capita",
"Gini Coefficient")

aic <- VARselect(ts_data, lag.max=8, type="trend")$selection["AIC(n)"]
model <- VAR(ts_data, p=aic, type="both")

future = forecast(model, h = 10)
plot(future, main = "Multivariate Forecast")
```

[2.3] Correlations Table in Power Query Editor

```
R.Execute("
# 'dataset' holds the input data for this script
#(lf)
data <- dataset
#(lf)
#(lf)
countries <- unique(data$country)
#(lf)
num_countries <- length(countries)
#(lf
)#(lf)
correlations <- replicate(num_countries,NA)
```

```
#(lf)
#(lf)
for (i in 1:num_countries){
        #(lf)
        place <- countries[i]#(lf)
        correlations[i] <- cor(data$carbonfp_pcap[data$country==place],
        #(lf)
        data$gdp_pcap[data$country==place])#(lf)}#(lf)correlation_df<-data.frame(countries,
        correlations)
#(lf)
#(lf)
correlation_df$countries <- as.factor(correlation_df$countries)
#(lf)
#(lf)
Correlation_df",
[dataset=#"Changed Type"])
```