STAT30340: Data Programming with R

Assignment 3

Anthony Doyle (19758695)

1 Task: Analysis

1.1 Data Set Cleaning

```
library(fable)
library(ggplot2)
library(maps)
library(rio)
library(tidyverse)
library(tsibble)

pwt1001 <- import("pwt1001.dta", setclass = "tibble")
print(missing <- sum(is.na(pwt1001)))
```

```
[1] 176681
```

The data set, *pwt1001*, used in this analysis is the *Penn World Table (PWT) 10.01*, which contains national accounts data across 183 countries from 1950 to 2019. It comprises 12810 observations across 52 variables, which includes measures such as real GDP, population and average annual hours worked by employed persons. The variables are mostly numeric, but there are also characters (e.g. Country, Currency Unit). We note that are a significant number of missing observations (76681); they are not removed and we do not impute values. Moreover, we specify the data set as a tibble for improved data handling.

```
pwt1001 <- pwt1001 |>
  select(
    countrycode, country, currency_unit, year, rgdpe, rgdpo, pop, emp, avh, hc,
    rgdpna, rconna, rdana, rnna, rkna, rtfpna, rwtfpna
  ) |>
  group_by(country) |>
  mutate(
    emprate = (emp / pop) * 100, # Employment rate
    avhw = avh / 52, # Average hours worked weekly
    rgdpna_pc = rgdpna / pop, # Real GDP per capita
    rgdpna_pc_g = (rgdpna_pc / lag(rgdpna_pc) - 1) * 100,
```

```
    # Real GDP per capita growth rate
  )
```

In this paper, we analyse national accounting data in the European Union (EU) from 1950 to 2019.

We begin by filtering the data set to our variables of interest: 3-letter ISO country code (*countrycode*); Country name (*country*); Currency unit (*currency_unit*); Year (*year*); Expenditure-side real GDP at chained PPPs (in mil. 2017US$) (*rgdpe*); Output-side real GDP at chained PPPs (in mil. 2017US$) (*rgdpo*); Population (in millions) (*pop*); Number of persons engaged (in millions) (*emp*); Average annual hours worked by persons engaged (source: The Conference Board) (*avh*); Human capital index (*hc*)[1]; Real GDP at constant 2017 national prices (in mil. 2017US$) (*rgdpna*); Real consumption at constant 2017 national prices (in mil. 2017US$) (*rconna*); Real domestic absorption at constant 2017 national prices (in mil. 2017US$) (*rdana*); Capital stock at constant 2017 national prices (in mil. 2017US$) (*rnna*); Capital services at constant 2017 national prices (2017=1) (*rkna*); TFP at constant national prices (2017=1) (*rtfpna*); and Welfare-relevant TFP at constant national prices (2017=1) (*rwtfpna*). (All other variables are removed; please see Appendix 1 for an exhaustive list of all variables in the original data set.)

We also produce new variables for per capita national accounting meausures. These are the employment rate (*emprate*); real GDP per capita (*rdgpna_pc*); and real GDP per capita growth rate (*rgdpna_pc_g*).

```
eu <- c("Austria", "Belgium", "Bulgaria", "Croatia", "Cyprus", "Czech Republic",
        "Denmark", "Estonia", "Finland", "France", "Germany", "Greece",
        "Hungary", "Iceland", "Ireland", "Italy", "Latvia", "Lithuania",
        "Luxembourg", "Malta", "Netherlands", "Poland", "Portugal", "Romania",
        "Slovakia", "Slovenia", "Spain", "Sweden")

pwt1001 <- pwt1001 |>
  mutate(
    eu = ifelse(country %in% eu, 1, 0)
  )

pwt1001 <- pwt1001 |>
  relocate(eu, .after = country)
```

The European Union is a supranational political and economic union, which consisted of 28 Member States in 2019. To identify EU Member States in this data set, we create a vector of states who held full EU Membership status in 2019. We then a create a dummy variable for EU (*eu*), and assign values (where 1 = EU Member (in 2019); 0 = not a EU Member (in 2019), to the countries accordingly. We also use the `relocate()` function to place the dummy variable for EU Membership (in 2019) (*eu*) after the variable for Country Name.

```
eu_avg <- pwt1001 |>
  filter(eu == 1) |>
  group_by(year) |>
  summarise(
    country = "EU",
```

[1]The Human Capital Index (HCI), calculated annually by the World Bank, "quantifies the contribution of health and education to the productivity of the next generation of workers".

```
    countrycode = "EU",
    across(where(is.numeric), ~ mean(.x, na.rm = TRUE))
  )

pwt1001 <- bind_rows(pwt1001, eu_avg)
```

To compute yearly averages for the EU, we create a new data frame, *eu_avg*, which filters for EU Member States; then, groups (via the `group_by()` function) all rows corresponding to the same year; and computes the yearly mean average (via `summarise()` function). For example, the real GDP value for the EU in 1975 is the mean average of all EU Member States (per 2019 EU Membership status) real GDP in that year. We specify the calculation to include numeric values only and for all missing observations ('NA') to be excluded. We then merge the *eu_avg* with the original data set using the `bind_rows()` function.[2]

[2]We find that the function `bind_rows()` accounts better for discrepancies when merging data sets, versus the alternative base R function `rbind()`.

1.2 Data Analysis

```
eu_1950 <- pwt1001 |>
  filter(eu == 1, year == 1950) |>
  select(country, rgdpna_pc, emprate, avhw, hc) |>
  arrange(desc(rgdpna_pc))

knitr::kable(
  eu_1950,
  caption = "National Accounting Figures for EU Countries in 1950
  (per 2019 EU Membership Status). Source: pwt1001",
  col.names = c("Country",
                "GDP per Capita",
                "Employment Rate",
                "Avg. Weekly Hrs. Worked",
                "HCI"),
  digits = 2
)
```

Table 1: National Accounting Figures for EU Countries in 1950 (per 2019 EU Membership Status). Source: pwt1001

Country	GDP per Capita	Employment Rate	Avg. Weekly Hrs. Worked	HCI
Luxembourg	18803.05	45.89	NA	1.43
Denmark	12823.84	46.19	39.41	2.84
Sweden	12447.34	48.06	38.56	2.32
Netherlands	11244.24	42.11	38.41	2.42
Belgium	9844.31	40.06	40.31	2.20
EU	8926.65	43.01	41.94	2.07
Ireland	8870.91	42.95	46.53	2.16
Iceland	8764.58	45.73	NA	1.97
France	8752.23	45.95	45.21	2.18
Austria	8157.60	42.04	40.11	2.55
Finland	7465.49	54.52	38.93	2.12
Italy	7249.37	42.39	40.38	1.79
Germany	7204.65	44.89	46.70	2.43
Spain	5153.95	41.48	42.49	1.87
Portugal	4046.43	37.74	46.25	1.24
Cyprus	3071.70	25.10	NA	1.52
Bulgaria	NA	NA	NA	NA
Czech Republic	NA	NA	NA	NA
Estonia	NA	NA	NA	NA
Greece	NA	NA	NA	NA
Croatia	NA	NA	NA	NA
Hungary	NA	NA	NA	NA
Lithuania	NA	NA	NA	NA
Latvia	NA	NA	NA	NA

Country	GDP per Capita	Employment Rate	Avg. Weekly Hrs. Worked	HCI
Malta	NA	NA	NA	NA
Poland	NA	NA	NA	NA
Romania	NA	NA	NA	NA
Slovakia	NA	NA	NA	NA
Slovenia	NA	NA	NA	NA

Table 1 displays the national accounting data for EU countries in 1950 on a per-country basis. It is arranged in descending order on the basis of GDP per capita. Luxembourg had the highest real GDP per capita in 1950, estimated as 18803.05 in 2017US$. whereas, Cyprus had the lowest at 3071.70 in 2017US$. Moreover, The EU average was 8926.65 in 2017US$. Employment rates, average hours worked weekly and the Human Capital Index figures are also provided in Table 1. Interestingly, Luxembourg has a relatively low HCI score (1.43) (the second lowest amongst the recorded countries in 1950).[3] [4]: However, we note that there is a considerable amount of missing data for this year, particularly for many Eastern European countries and former Soviet states.

```
eu_2019 <- pwt1001 |>
  filter(eu == 1, year == 2019) |>
  select(country, rgdpna_pc, emprate, avhw, hc) |>
  arrange(desc(rgdpna_pc))

knitr::kable(
  eu_2019,
  caption = "National Accounting Figures for EU Countries in 2019
  (per 2019 EU Membership Status). Source: pwt1001",
  col.names = c("Country",
                "GDP per Capita",
                "Employment Rate",
                "Average Weekly Hrs. Worked",
                "Human Capital Index"),
  digits = 2
)
```

Table 2: National Accounting Figures for EU Countries in 2019 (per 2019 EU Membership Status). Source: pwt1001

Country	GDP per Capita	Employment Rate	Average Weekly Hrs. Worked	Human Capital Index
Ireland	96812.32	46.30	34.08	3.19
Luxembourg	91864.90	74.82	28.95	3.62
Netherlands	56194.86	55.31	27.69	3.40
Denmark	53927.64	51.49	26.55	3.60

[3]We find that the function `bind_rows()` accounts better for discrepancies when merging data sets, versus the alternative base R function `rbind()`.

[4]We find that the function `bind_rows()` accounts better for discrepancies when merging data sets, versus the alternative base R function `rbind()`.

Country	GDP per Capita	Employment Rate	Average Weekly Hrs. Worked	Human Capital Index
Austria	53234.83	50.81	30.99	3.38
Sweden	53012.51	49.84	30.87	3.44
Iceland	52016.95	56.73	27.97	3.28
Germany	51654.93	53.64	26.66	3.68
Belgium	46285.39	42.65	30.51	3.15
Finland	44832.87	48.33	30.59	3.50
France	44027.97	42.36	28.94	3.23
EU	42559.18	48.64	32.63	3.34
Italy	40756.57	42.27	33.04	3.16
Spain	40574.37	42.52	32.42	2.99
Malta	39273.25	50.00	36.83	3.16
Czech Republic	37399.15	51.28	34.38	3.67
Slovenia	33993.17	50.36	30.63	3.57
Estonia	33786.15	50.77	34.56	3.66
Lithuania	33358.36	49.99	36.27	3.30
Cyprus	33043.13	42.21	34.72	2.90
Portugal	32191.94	48.52	35.86	2.51
Poland	32047.85	42.65	38.90	3.46
Hungary	29800.27	48.65	33.18	3.42
Latvia	29506.63	47.02	35.86	3.16
Slovakia	28053.99	45.24	32.60	3.85
Romania	28048.31	44.83	34.46	3.27
Greece	27640.66	40.44	39.16	3.14
Croatia	27148.82	44.01	35.22	3.62
Bulgaria	21169.20	48.86	31.64	3.19

Table 2 displays the national accounting data for EU countries in 2019 on a per-country basis. Again, it is arranged in descending order on the basis of GDP per capita. Ireland had the highest GDP per capita, estimated as 96812.32 in 2017US$, which was substantially higher than the EU average of 42559.18 in 2017US$. On the other hand, Bulgaria had the lowest GDP per capita at 21169.20 in 2017US$.

Interestingly, Greece, which had the third lowest GDP per capita in 2019 (27640.66 in 2017US$), had the highest average weekly hours worked, 39.16 hours. This contrasts with high GDP per capita countries. For example, the Netherlands, which had the second highest real GDP per capita (56194.86 in 2017US$), had the lowest average weekly hours worked, 27.69 hours.[5]

```
eu_1950_2019 <- pwt1001 |>
  filter(year %in% c(1950, 2019), eu == 1) |>
  select(country, year, rgdpna_pc)

country_order <- eu_1950_2019 |>
  filter(year == 2019) |>
```

[5]See Figure 3: Average Hours Worked Weekly vs Real GDP Per Capita for EU Countries in 2019, which explores this relationship in more detail.

```
  arrange(desc(rgdpna_pc)) |> # Arrange in descending order for 2019 rgdpna_pc
  pull(country)

eu_1950_2019 <- eu_1950_2019 |>
  mutate(country = factor(country, levels = country_order)) |>
  mutate(
    fill_color = case_when(
      country == "EU" & year == 1950 ~ "EU 1950",
      country == "EU" & year == 2019 ~ "EU 2019",
      year == 1950 ~ "1950",
      year == 2019 ~ "2019"
    )
  )

# Custom fills are implemented to highlight the EU Average in 1950 and 2019
custom_colors <- c(
  "EU 1950" = "grey",
  "EU 2019" = "black",
  "1950" = "#b3cde3",
  "2019" = "#011f4b"
)

figure1 <- ggplot(eu_1950_2019, aes(x = country,
                                    y = rgdpna_pc, fill = fill_color)) +
  geom_bar(stat = "identity",
           position = position_dodge(width = 1)) +  # Group bars side-by-side
  scale_fill_manual(
    values = custom_colors
  ) +
  labs(
    y = "GDP per Capita (2017US$)",
    caption = "Source: pwt1001"
  ) +
  coord_flip() +
  theme_bw() +
  theme(
    legend.title = element_blank(),
    legend.text = element_text(family = "serif", size = 8),
    legend.key.size = unit(0.75, "lines"),
    axis.title.y = element_blank(),
    axis.title = element_text(family = "serif"),
    axis.text = element_text(family = "serif"),
    plot.title = element_text(family = "serif"),
    plot.caption = element_text(family = "serif")
  )
figure1
```

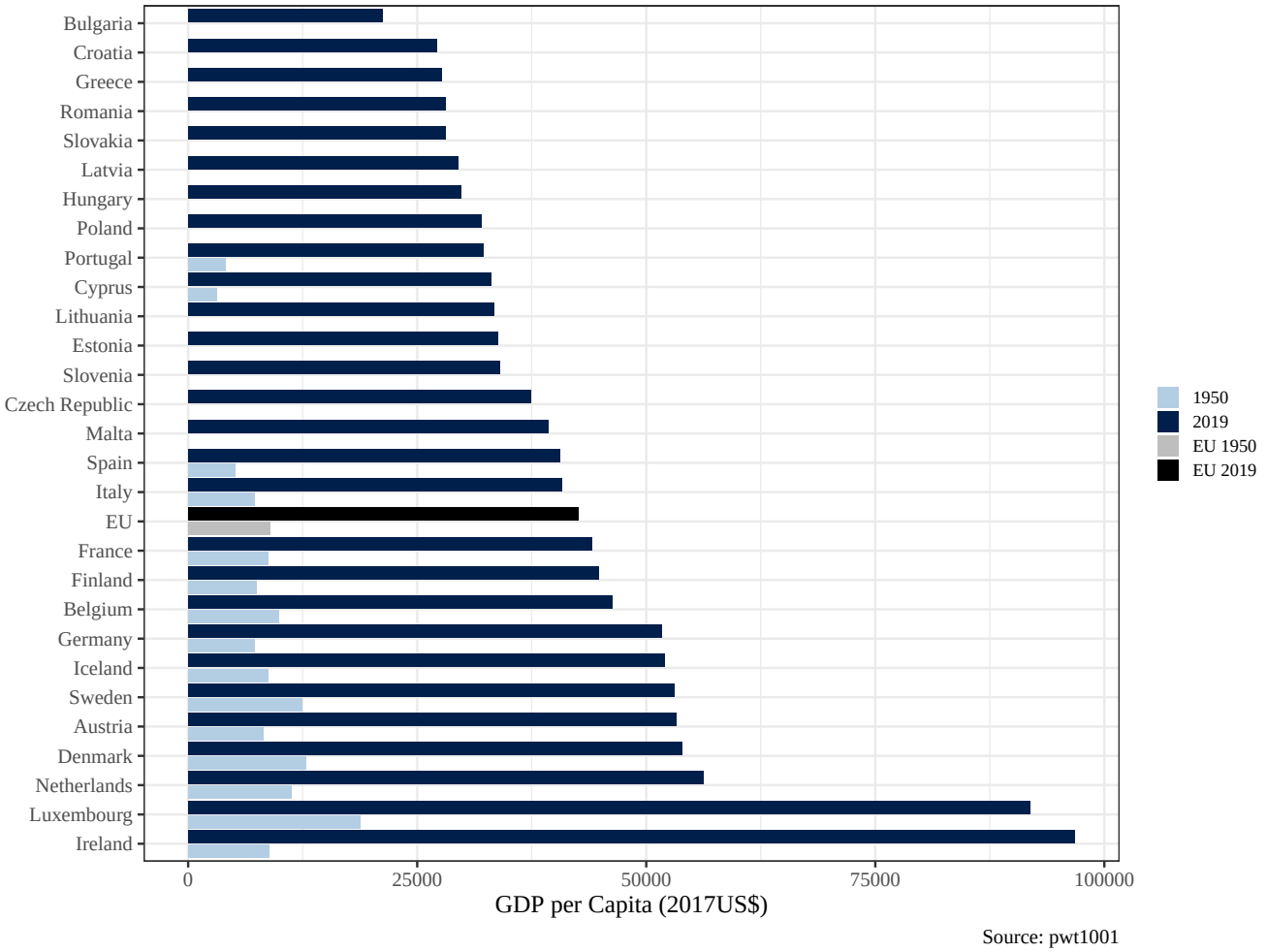

Figure 1: Real GDP per Capita for EU Countries (1950 and 2019)

Figure 1 displays GDP per capita for EU Member States in 1950 and 2019 compared. The figure is arranged in descending order on the basis of GDP per capita in 2019. In all cases, where GDP per capita data existed in 1950, GDP per capita has increased.

```
eu_countries <- c("Austria", "Belgium", "Bulgaria", "Croatia", "Cyprus",
                  "Czech Republic", "Denmark", "Estonia", "Finland", "France",
                  "Germany", "Greece", "Hungary", "Iceland", "Ireland", "Italy",
                  "Latvia", "Lithuania", "Luxembourg", "Malta", "Netherlands",
                  "Poland", "Portugal", "Romania", "Slovakia", "Slovenia",
                  "Spain", "Sweden")

pwt_eu <- pwt1001 |>
  filter(country %in% eu_countries, year == 2019) |>
  select(country, rgdpna_pc)

# Rename the column for consistency
europe_map <- map_data("world")
europe_map <- europe_map |>
```

```r
  mutate(is_eu = ifelse(region %in% eu_countries, "EU", "Non-EU")) |>
  left_join(pwt_eu, by = c("region" = "country"))

figure2 <- ggplot() +
  geom_polygon(data = europe_map |> filter(is_eu == "Non-EU"),
               aes(x = long, y = lat, group = group),
               fill = "grey80", color = "white", size = 0.2) +

  geom_polygon(data = europe_map |> filter(is_eu == "EU"),
               aes(x = long, y = lat, group = group, fill = rgdpna_pc),
               color = "white", size = 0.2) +
  scale_fill_gradient(name = "Real GDP PC (2017 US$)",
                      low = "#b3cde3", high = "#011f4b", na.value = "grey80",
                      labels = scales::comma) + # Normal form values
  labs(caption = "Source: pwt1001") +
  coord_quickmap(xlim = c(-25, 45), ylim = c(35, 72)) + # For Europe only
  theme_minimal() +
  theme(
    axis.title = element_blank(), # Remove longitude-latitude axes
    axis.text = element_blank(),
    axis.ticks = element_blank(),
    plot.caption = element_text(family = "serif"),
    legend.position = "right",
    legend.title = element_text(size = 10, family = "serif"),
    legend.text = element_text(size = 8, family = "serif"),
    legend.key.size = unit(0.75, "lines")
  ) +
  guides(fill = guide_legend(keywidth = 0.8, keyheight = 0.8))
figure2
```

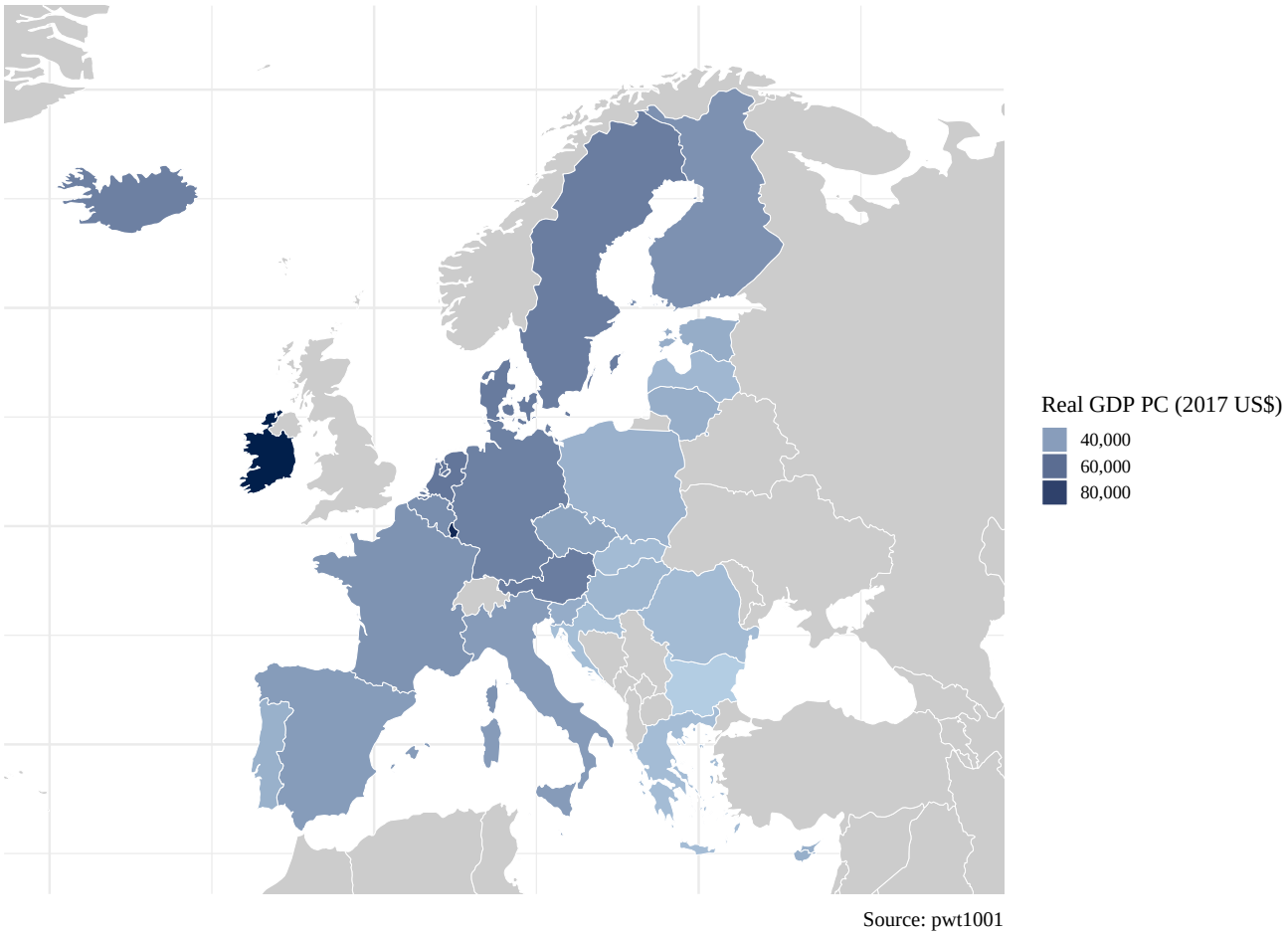

Figure 2: Real GDP per Capita Map for EU Countries in 2019

Figure 2 displays a heat map for real GDP per capita for EU Member States in 2019. Evidently, Ireland has the highest real GDP per capita. Moreover, there seems to be a regional relationship with GDP per capita: Eastern European countries appear to have lower GDP per capita than Western EU Member States.

```
eu_scatter <- pwt1001 |>
  filter(year == 2019, eu == 1) |>
  select(country, rgdpna_pc, avhw)

figure3 <- ggplot(eu_scatter, aes(x = avhw, y = rgdpna_pc)) +
  geom_point(color = "#011f4b", size = 3) +
  geom_text(aes(label = country),
            vjust = -0.5,
            hjust = 0.5,
            size = 3,
            family = "serif") +
  geom_smooth(method = "lm", se = FALSE, color = "#b3cde3") +
  scale_y_continuous(labels = scales::comma) +
```

```
  labs(
    x = "Average Hours Worked Weekly",
    y = "Real GDP per Capita (2017 US$)"
  ) +
  theme_bw() +
  theme(
    legend.position = "none",
    axis.title.y = element_blank(),
    axis.title = element_text(family = "serif"),
    axis.text = element_text(family = "serif"),
    plot.title = element_text(family = "serif"),
    plot.caption = element_text(family = "serif")
  )
figure3
```

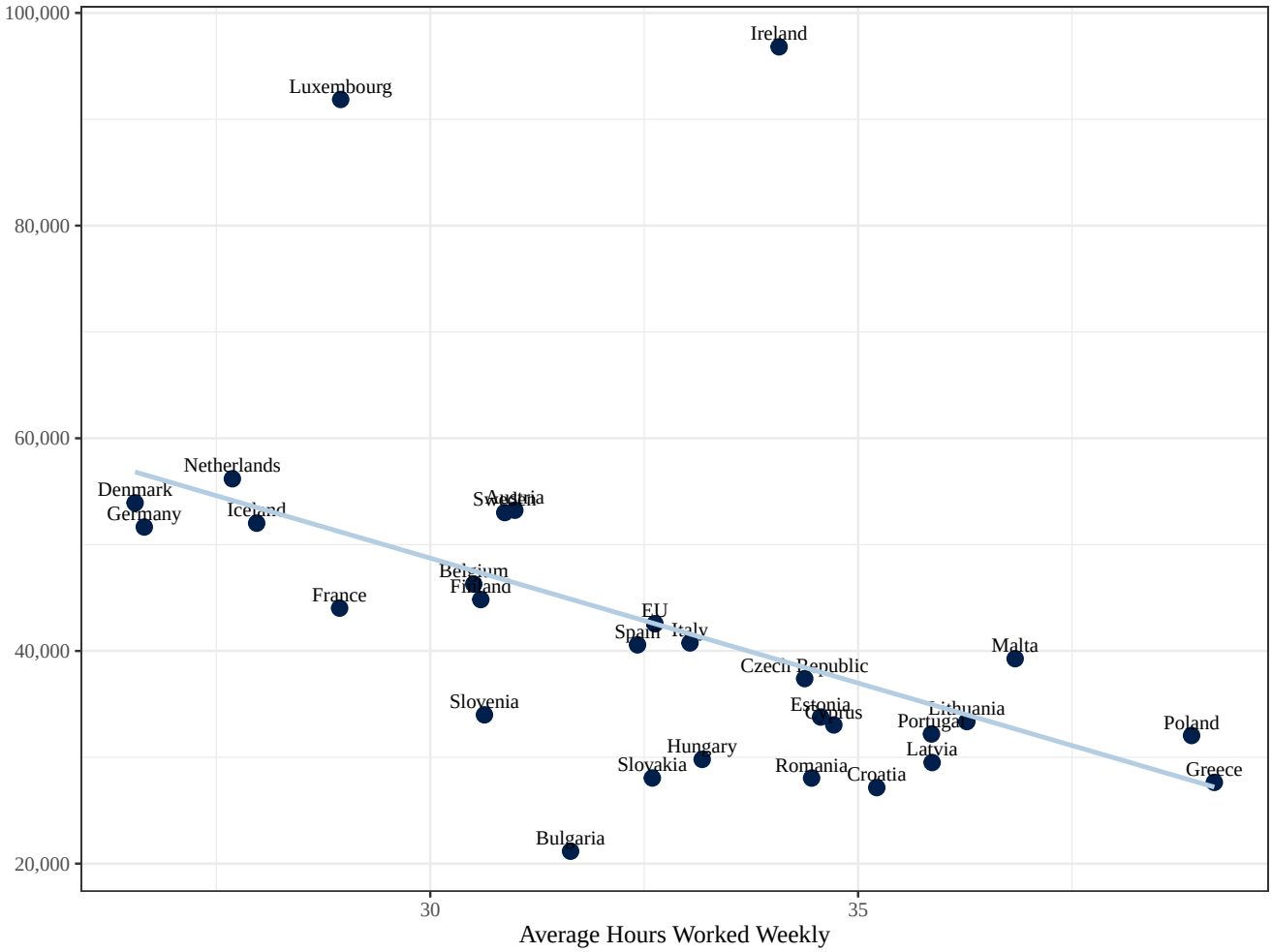

Figure 3: Average Hours Worked Weekly vs Real GDP Per Capita for EU Countries in 2019

Figure 2 displays the relationship between real GDP per capita and the average hours worked weekly across EU Member States in 2019. A linear regression line is fitted. There appears to be a negative

relationship, which postulates that more "productive" workers work less.

2 Task: R Package

2.1 Real GDP Per Capita Growth

```
ts_world <- pwt1001 |>
  filter(
    year >= 1950 & year <= 2019,
    country %in% c("EU",
                   "United States",
                   "China",
                   "India",
                   "Russian Federation")
  ) |>
  select(country, year, rgdpna_pc)

figure3 <- ggplot(ts_world, aes(x = year, y = rgdpna_pc, color = country)) +
  geom_line(size = 0.5) +
  labs(
    y = "Real GDP Per Capita",
    caption = "Source: pwt1001"
  ) +
  scale_color_manual(
    values = c(
      "EU" = "#011f4b",
      "United States" = "#E0E0E0",
      "China" = "#B0B0B0",
      "India" = "#808080",
      "Russia" = "#404040"
    )
  ) +
  theme_bw() +
  theme(
    legend.position = "right",
    legend.title = element_blank(),
    legend.text = element_text(family = "serif", size = 8),
    legend.key.size = unit(0.75, "lines"),
    axis.title.x = element_blank(),
    axis.title = element_text(family = "serif"),
    axis.text = element_text(family = "serif"),
    plot.title = element_text(family = "serif"),
    plot.caption = element_text(family = "serif")
  )
figure3
```

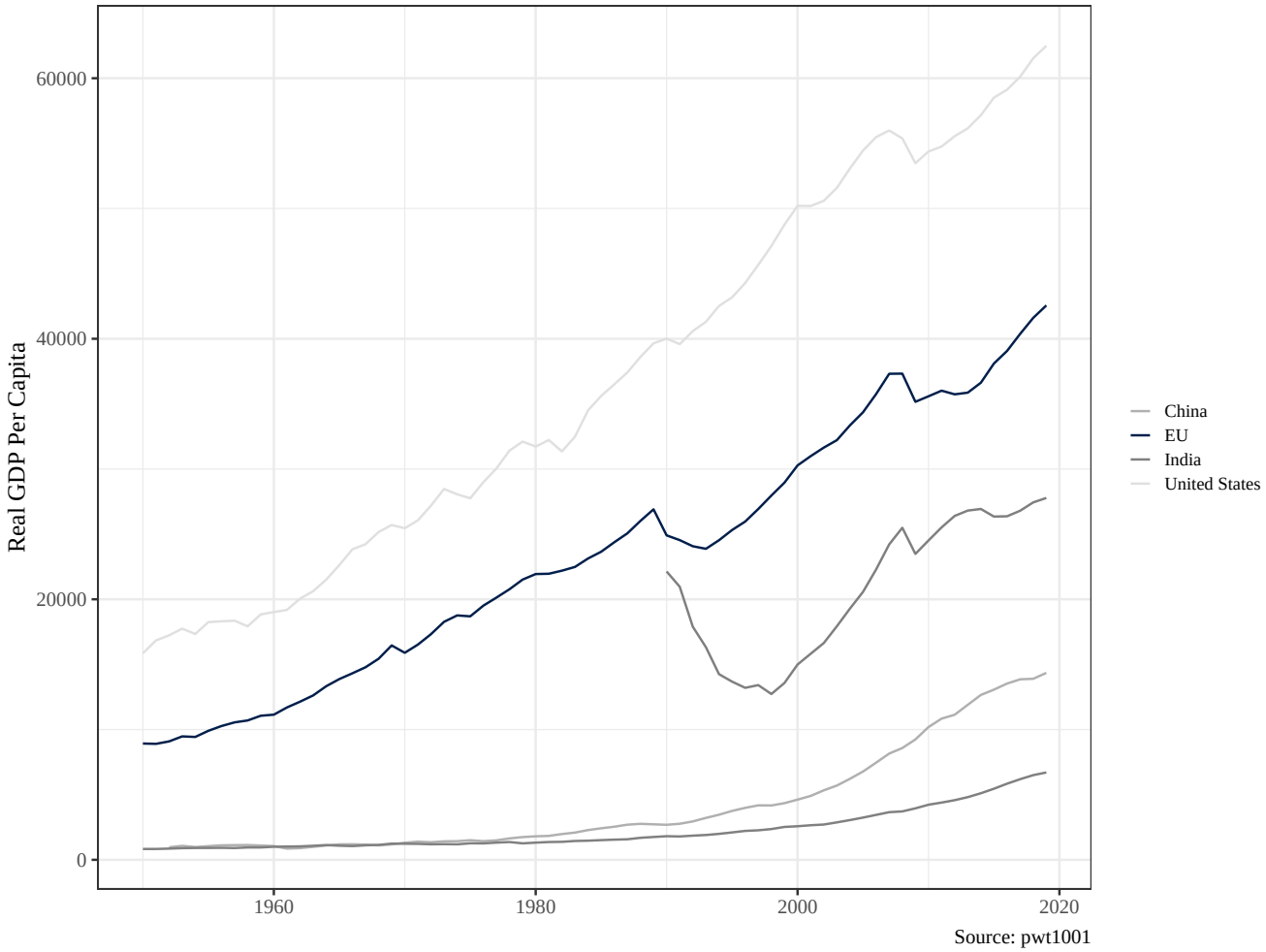

Figure 4: Real GDP Per Capita (1950-2019)

National accounts provide key historical time series data for forecasting future economic growth projections. Figure 4 displays a time series of real GDP per capita for the EU, the United States, China, Russia and India from 1950 to 2019.[^4] The EU has notably experienced strong positive gains over this period; but what does this say about future growth projections? To answer this question - although, one can never truly predict the future - we can create 10-year forecasts, which avails of the time series data on GDP figures in *pwt1001* to generate future projections. To create a simple 10-year forecasting model, we require the use of two packages: `tsibble` and `fable`.

2.2 `tsibble`

```
if (!requireNamespace("tsibble", quietly = TRUE)) {
  install.packages("tsibble")
}

eu_data <- pwt1001 |>
  filter(country == "EU") |>
```

```
  drop_na(rgdpna_pc, rgdpna_pc_g)

eu_ts <- eu_data |> # Convert to a tsibble
  as_tsibble(index = year) |>
  fill_gaps()
```

The package `tsibble` provides a data infrastructure for tidy temporal data with wrangling tools, which is essential for the use of forecasting models (such as in the `fable` package). First, we create a data frame, *eu_data*, which contains data for real GDP per capita and real GDP per capita growth rates for the EU from the 1950 to 2019. Using the `as_tsibble()` function, the data frame is converted to a 'tssible' (i.e., a time series tibble). The index specified is year, and we also request for any gaps in the data (i.e., missing data for years) to be filled with NA values.[^4] [^4] This is a precautionary specification, considering there are actually no missing years in the *eu_data* set.

2.3 `fable`

```
if (!requireNamespace("fable", quietly = TRUE)) {
  install.packages("fable")
}

fit <- eu_ts |>
  model(ARIMA(rgdpna_pc))

forecast <- fit |>
  forecast(h = 10) # GDP per capita 10 year forecast
```

The package `fable` provides a selection of tools for univariate and multivariate time series forecasting models. It is commonly used in econometric analysis, and is particularly useful for our analysis of real GDP per capita projections. We use the `model()` function to estimate an ARIMA (AutoRegressive Integrated Moving Average) model for real GDP per capita, using EU-level data.[6] We then employ the `forecast()` function to produce a 10 year forecast for real gdp per capita in the EU (specifying the horizon of the forecast as 10 (h = 10)).

```
forecast |>
  autoplot(eu_ts) +  # Includes the confidence interval
  labs(
    y = "Real GDP Per Capita",
    caption = "Source: pwt1001"
  ) +
  theme_bw() +
  theme(
    legend.position = "right",
    legend.title = element_blank(),
```

[6]ARIMA uses historical time series data to forecast future trends, combining unit root tests, minimisation of the AIC and MLE to obtain an ARIMA model.

```
    legend.text = element_text(family = "serif", size = 8),
    legend.key.size = unit(0.75, "lines"),
    axis.title.x = element_text(family = "serif"),
    axis.title = element_text(family = "serif"),
    axis.text = element_text(family = "serif"),
    plot.title = element_text(family = "serif"),
    plot.caption = element_text(family = "serif")
  )
```

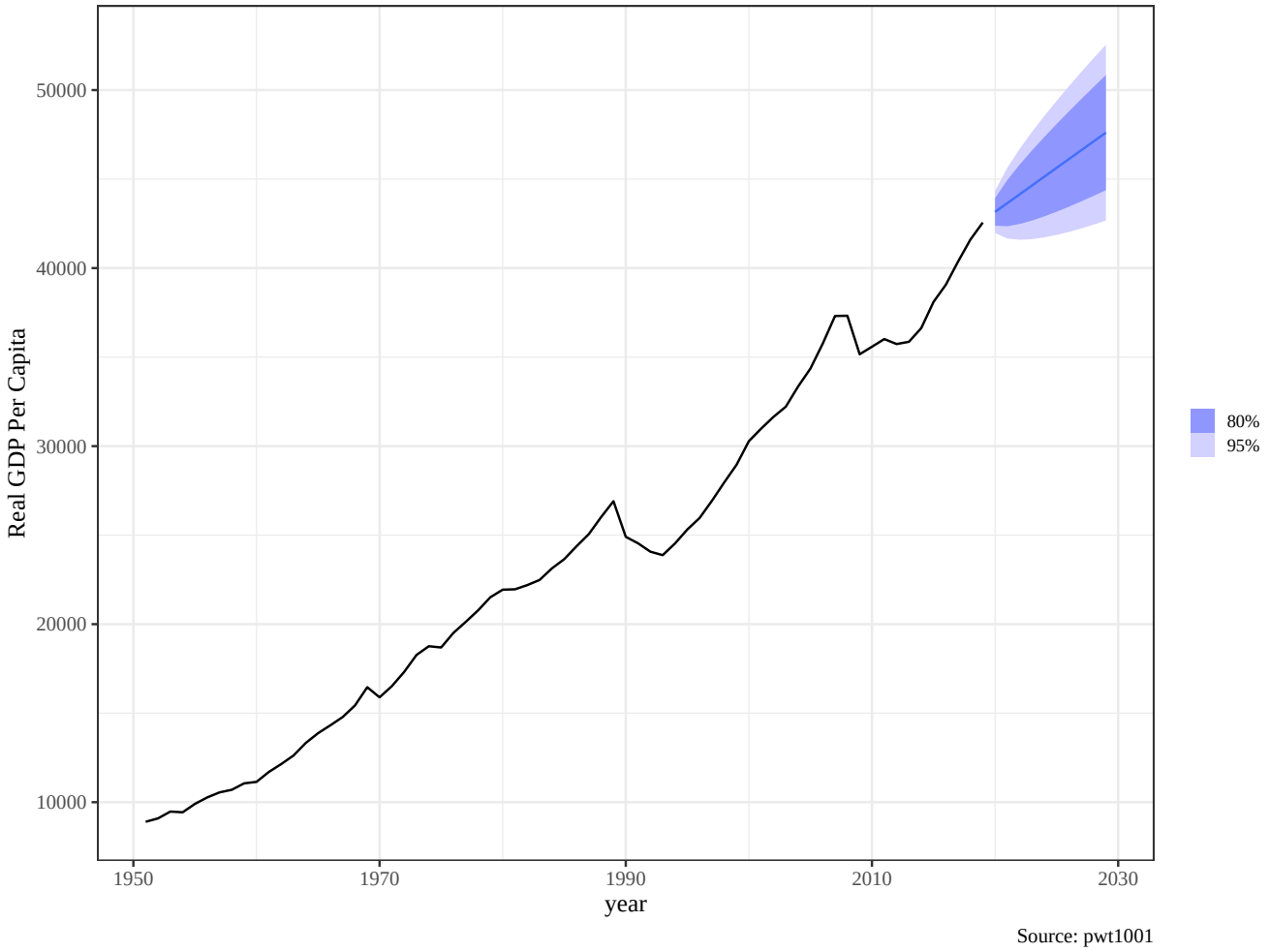

Figure 5: Real GDP per Capita 10 Year Projection for the EU

Figure 5 displays a basic plot, produced with the `autplot()` function, of GDP per capita 10 Year Forecasts for the EU. The thin blue line is the projection line, while the surrounding shaded areas are the 80 and 95 per cent confidence intervals respectively. Evidently, EU real GDP per capita looks set to increase over the next ten years.

3 Task: Functions/Programming

The Solow residual, also known as Total Factor Productivity (TFP), can be described as the improvement in labour productivity that derives from technological advances. It is essentially the the productivity growth that is calculated when all other factors of production (capital and labour) are subtracted from the Exogeneous/Solow growth model.

We define our function as follows:

1. **Function Name and Arguments**
   The function is named `solow_residual` and accepts four arguments: the relevant dataset `det` and the variables for capital, output, labor, and alpha.

2. **Conditional Check**
   A conditional check is implemented: the function will terminate if the specified variables are not present in the dataset, displaying an error message:
   `"One or more specified columns are not in the dataset."`

3. **Adding the New Variable**
   A new variable, *solow*, is added to the dataset. This variable is calculated as the product of:
   `capital^alpha * labor^(1 - alpha)`
   divided by output.

4. **Using `!!rlang::sym()`**
   The `!!rlang::sym()` construct is used to convert strings into symbols, enabling their use in `tidyverse` functions like `mutate()` or `filter()`.

   - The `!!` operator unquotes symbols for tidyverse evaluation.

5. **Creating an S3 Object**
   A list containing the relevant variables is created and then converted into an S3 object with the class attribute `"SolowResidual"`.

   - This object-oriented system in R allows for specific methods such as `summary.SolowResidual`.

6. **Defining a Print Method**
   A print method is defined to enhance the function's readability in R:

   - The `cat()` function outputs text to the console.

   - The `invisible()` function ensures the object is returned without additional output.

7. **Custom Summary Function**
   A custom function, `summary.SolowResidual`, is created to provide a formatted summary of the object.

8. **Calculating Descriptive Statistics**
   The summary function calculates descriptive statistics for the relevant variables in the dataset.

```r
solow_residual <- function(data, capital, output, labour, alpha = 0.3) {
  # Assume a constant capital share of output (alpha = 0.3)
  # Check if required columns are in the dataset
  if (!all(c(capital, output, labour) %in% colnames(data))) {
    stop("One or more specified columns are not in the data set")
  }

  data <- data |>
    mutate(
      # Calculate Solow residual based on the given formula
      residual = get(output) /
        (get(capital)^alpha * get(labour)^(1 - alpha))
    )

  # Returning an object of class "SolowResidual"
  structure(
    list(
      data = data,
      alpha = alpha,       # Capital share parameter
      capital = capital,   # Capital column name
      output = output,     # Output column name
      labour = labour      # Labour column name
    ),
    class = "SolowResidual"
  )
}

print.SolowResidual <- function(x, ...) {
  # Printing basic information about the SolowResidual object
  cat("Solow Residual Analysis:\n")
  cat("Alpha (Capital Share):", x$alpha, "\n")
  cat("Capital:", x$capital, "\n")
  cat("Output:", x$output, "\n")
  cat("Labour:", x$labour, "\n")
  invisible(x) # Ensuring no additional output is shown
}

summary.SolowResidual <- function(object, ...) { # Unspecified arguments
  # Print a summary of the SolowResidual object
  cat("Solow Residual Summary:\n")
  cat("Alpha (Capital Share):", object$alpha, "\n")
  cat("Capital:", object$capital, "\n")
  cat("Output:", object$output, "\n")
  cat("Labour:", object$labour, "\n")

  # Calculating and displaying summary statistics of the residual
  residual_stats <- object$data |>
    summarise(
```

```
      mean_residual = ifelse(all(is.na(residual)),
                             NA, mean(residual, na.rm = TRUE)),
      sd_residual = ifelse(all(is.na(residual)),
                           NA, sd(residual, na.rm = TRUE)),
      min_residual = ifelse(all(is.na(residual)),
                             NA, min(residual, na.rm = TRUE)),
      max_residual = ifelse(all(is.na(residual)),
                             NA, max(residual, na.rm = TRUE))
    )

  print(residual_stats) # Display summary statistics
  invisible(object) # To ensure the object is displayed once only
}

plot_solow_residual <- function(x) {
  ggplot(
    x$data %>% filter(!is.na(residual)), # Filter out rows with NA residuals
    aes(x = year, y = residual)
  ) +
    geom_line(color = "#011f4b", size = 1) +
    labs(
      title = "Solow Residual Plot",
      x = "Year",
      y = "Solow Residual"
    ) +
    theme_bw() +
    theme(
      axis.title = element_text(family = "serif"),
      axis.text = element_text(family = "serif"),
      plot.title = element_text(family = "serif"),
      plot.caption = element_text(family = "serif")
    )
}
```

```
eu_residual_data <- pwt1001 |> filter(country == "eu")

eu_residual <- solow_residual(
  eu_data,
  capital = "rnna",
  output = "rgdpna",
  labour = "emp"
)

print(eu_residual)
```

```
Solow Residual Analysis:
Alpha (Capital Share): 0.3
Capital: rnna
```

```
Output: rgdpna
Labour: emp
```

```
summary(eu_residual)
```

```
Solow Residual Summary:
Alpha (Capital Share): 0.3
Capital: rnna
Output: rgdpna
Labour: emp
# A tibble: 1 x 5
  country mean_residual sd_residual min_residual max_residual
  <chr>           <dbl>       <dbl>        <dbl>        <dbl>
1 EU              1321.        352.         630.        1817.
```

```
# plot(eu_residual) # Error: "Error in xy.coords(x, y, xlabel, ylabel, log):
# 'x' is a list, but does not have components 'x' and 'y'"
```

We calculate the Solow residual for the EU (using the *pwt1001* data) by specifying the relevant variables for capital ("rnna"), output ("rgdpna") and labour ("emp"), and calling the solow_residual function. This returns summary statistics such as the mean average residual for the years 1950-2019 (1320.693), as well the standard deviation (351.5298).

We also produce a basic plot, visualising the Solow residual/TFP growth from 2019 to 1950. (Note the error: "Error in xy.coords(x, y, xlabel, ylabel, log): 'x' is a list, but does not have components 'x' and 'y' "). To view the plot, run the code in RStudio.

References

- Mitchell O'Hara-Wild, Rob Hyndman, and Earo Wang (2024). *fable: Forecasting Models for Tidy Time Series*. R package version 0.4.1. Available at https://CRAN.R-project.org/package=fable.
- Hadley Wickham (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. ISBN: 978-3-319-24277-4. Available at https://ggplot2.tidyverse.org.
- Original S code by Richard A. Becker and Allan R. Wilks, R version by Ray Brownrigg, with enhancements by Thomas P. Minka and Alex Deckmyn (2023). *maps: Draw Geographical Maps*. R package version 3.4.2. Available at https://CRAN.R-project.org/package=maps.
- Chung-hong Chan, Thomas J. Leeper, Jason Becker, and David Schoch (2023). *rio: A Swiss-army knife for data file I/O*. Available at https://cran.r-project.org/package=rio.
- Hadley Wickham, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'Agostino McGowan, Romain François, Garrett Grolemund, Alex Hayes, Lionel Henry, Jim Hester, Max Kuhn, Thomas Lin Pedersen, Evan Miller, Stephan Milton Bache, Kirill Müller, Jeroen Ooms, David Robinson, Dana Paige Seidel, Vitalie Spinu, Kohske Takahashi, Davis Vaughan, Claus Wilke, Kara Woo, and Hiroaki Yutani (2019). "Welcome to the tidyverse." *Journal of Open Source Software*, 4(43), 1686. DOI: https://doi.org/10.21105/joss.01686.

- Earo Wang, Dianne Cook, and Rob J. Hyndman (2020). "A new tidy data structure to support exploration and modeling of temporal data." *Journal of Computational and Graphical Statistics*, 29(3), 466-478. DOI: https://doi.org/10.1080/10618600.2019.1695624.