Climate Cast - Analysis of Climate Change

Batch 16 - G Sumanth (160121771037) T Pavan Kumar (160121771062) T Sameekruth (160121771063)

Data Cleaning

This document contains code and docs for the data cleaning process performed on multiple datasets related to climate data. The cleaning process involves reading in various datasets, handling missing or duplicate values, renaming columns, merging datasets, and saving the cleaned data into tidy format.

Air Quality Dataset

The air quality dataset is read from a CSV file located at “data/original/Air Quality.csv”. Initially, the dataset is inspected by displaying its first few rows using the head() function. Duplicate columns with the same value in every row are identified and removed. The dataset is then further processed by renaming columns, selecting relevant columns, and cleaning column names using the clean_names() function from the janitor package. The cleaned dataset is finally saved into a CSV file named “air_quality.csv” in the “data/tidy” directory.

```
air_quality <- read.csv("Climate-main/data/original/Air Quality.csv")
head(air_quality)
```

```
##   COU   Country SMALL_SUBNATIONAL_REGION Small.subnational.region LARGE_SUBNATIONAL_REGION
## 1 AUS Australia                       NA           Not applicable                   TOTAL
## 2 AUS Australia                       NA           Not applicable                   TOTAL
## 3 AUS Australia                       NA           Not applicable                   TOTAL
## 4 AUS Australia                       NA           Not applicable                   TOTAL
## 5 AUS Australia                       NA           Not applicable                   TOTAL
## 6 AUS Australia                       NA           Not applicable                   TOTAL
##   Large.subnational.region    VAR                          Variable  YEA Year Unit.Code
## 1                    Total PWM_EX Mean population exposure to PM2.5 1990 1990  MICRO_M3
## 2                    Total PWM_EX Mean population exposure to PM2.5 1995 1995  MICRO_M3
## 3                    Total PWM_EX Mean population exposure to PM2.5 2000 2000  MICRO_M3
## 4                    Total PWM_EX Mean population exposure to PM2.5 2001 2001  MICRO_M3
## 5                    Total PWM_EX Mean population exposure to PM2.5 2002 2002  MICRO_M3
## 6                    Total PWM_EX Mean population exposure to PM2.5 2003 2003  MICRO_M3
##                       Unit PowerCode.Code PowerCode Reference.Period.Code Reference.Period Value
## 1 Micrograms per cubic metre             0     Units                   NA               NA  7.76
## 2 Micrograms per cubic metre             0     Units                   NA               NA  7.46
## 3 Micrograms per cubic metre             0     Units                   NA               NA  7.45
## 4 Micrograms per cubic metre             0     Units                   NA               NA  7.58
## 5 Micrograms per cubic metre             0     Units                   NA               NA  7.60
## 6 Micrograms per cubic metre             0     Units                   NA               NA  7.91
##   Flag.Codes           Flags
## 1          E Estimated value
## 2          E Estimated value
## 3          E Estimated value
## 4          E Estimated value
## 5          E Estimated value
## 6          E Estimated value
```

```
same_value_columns <-
  apply(air_quality, 2, function(x) {
    length(unique(x)) == 1
  })
same_value_column_names <- names(air_quality)[same_value_columns]
print(same_value_column_names)
```

```
##  [1] "SMALL_SUBNATIONAL_REGION" "Small.subnational.region" "LARGE_SUBNATIONAL_REGION"
##  [4] "Large.subnational.region" "VAR"                      "Variable"
##  [7] "Unit.Code"                "Unit"                     "PowerCode.Code"
## [10] "PowerCode"                "Reference.Period.Code"    "Reference.Period"
```

```
air_quality <- air_quality[!same_value_columns]
```

```
air_quality <- air_quality |>
  rename(iso3 = "COU", air_quality = "Value") |>
  select(-c("Flag.Codes", "YEA", "Flags")) |>
  clean_names()
head(air_quality)
```

```
##   iso3   country year air_quality
## 1  AUS Australia 1990        7.76
## 2  AUS Australia 1995        7.46
## 3  AUS Australia 2000        7.45
## 4  AUS Australia 2001        7.58
## 5  AUS Australia 2002        7.60
## 6  AUS Australia 2003        7.91
```

```
air_quality <- air_quality[order(air_quality$iso3), ]
write_csv(air_quality, file = "Climate-main/data/tidy/air_quality.csv")
```

Annual Mean Global Surface Temp

The annual mean global surface temperature dataset is read from a CSV file located at “data/original/Annual Mean Global Surface Temp.csv”. Similar to the air quality dataset, the initial inspection is performed, and duplicate columns with the same value in every row are removed. The dataset is then reshaped from wide to long format using the pivot_longer() function to have a single column for the year. Column names are cleaned, and the resulting tidy dataset is saved into a CSV file named “annual_mean_global_surface_temp.csv” in the “data/tidy” directory.

```
annual_mean_global_surface_temp <-
  read_delim(
    "Climate-main/data/original/Annual Mean Global Surface Temp.csv",
    show_col_types = FALSE
  )
head(annual_mean_global_surface_temp)
```

```
## # A tibble: 6 × 72
##   ObjectId Country ISO2  ISO3  Indicator Unit  Source `CTS Code` `CTS Name` `CTS Full Descriptor` `1961`
##      <dbl> <chr>   <chr> <chr> <chr>     <chr> <chr>  <chr>      <chr>      <chr>                   <dbl>
## 1        1 Afghan… AF    AFG   Temperat… Degr… Food … ECCS       Surface T… Environment, Climate… -0.113
## 2        2 Albania AL    ALB   Temperat… Degr… Food … ECCS       Surface T… Environment, Climate…  0.627
## 3        3 Algeria DZ    DZA   Temperat… Degr… Food … ECCS       Surface T… Environment, Climate…  0.164
## 4        4 Americ… AS    ASM   Temperat… Degr… Food … ECCS       Surface T… Environment, Climate…  0.079
## 5        5 Andorr… AD    AND   Temperat… Degr… Food … ECCS       Surface T… Environment, Climate…  0.736
## 6        6 Angola  AO    AGO   Temperat… Degr… Food … ECCS       Surface T… Environment, Climate…  0.041
## # ℹ 61 more variables: `1962` <dbl>, `1963` <dbl>, `1964` <dbl>, `1965` <dbl>, `1966` <dbl>,
## #   `1967` <dbl>, `1968` <dbl>, `1969` <dbl>, `1970` <dbl>, `1971` <dbl>, `1972` <dbl>, `1973` <dbl>,
## #   `1974` <dbl>, `1975` <dbl>, `1976` <dbl>, `1977` <dbl>, `1978` <dbl>, `1979` <dbl>, `1980` <dbl>,
## #   `1981` <dbl>, `1982` <dbl>, `1983` <dbl>, `1984` <dbl>, `1985` <dbl>, `1986` <dbl>, `1987` <dbl>,
## #   `1988` <dbl>, `1989` <dbl>, `1990` <dbl>, `1991` <dbl>, `1992` <dbl>, `1993` <dbl>, `1994` <dbl>,
## #   `1995` <dbl>, `1996` <dbl>, `1997` <dbl>, `1998` <dbl>, `1999` <dbl>, `2000` <dbl>, `2001` <dbl>,
## #   `2002` <dbl>, `2003` <dbl>, `2004` <dbl>, `2005` <dbl>, `2006` <dbl>, `2007` <dbl>, `2008` <dbl>, …
```

```
same_value_columns <-
  apply(annual_mean_global_surface_temp, 2, function(x) {
    length(unique(x)) == 1
  })
same_value_column_names <- names(annual_mean_global_surface_temp)[
  same_value_columns
]
print(same_value_column_names)
```

```
## [1] "Indicator"           "Unit"                "Source"              "CTS Code"
## [5] "CTS Name"            "CTS Full Descriptor"
```

```
annual_mean_global_surface_temp <- annual_mean_global_surface_temp[
  !same_value_columns
]
```

```
annual_mean_global_surface_temp <- annual_mean_global_surface_temp |>
  pivot_longer(
    cols = matches("^\\d{4}$"),
    names_to = "year",
    values_to = "mean_global_surface_temp"
  ) |>
  select(-"ObjectId") |>
  clean_names()
head(annual_mean_global_surface_temp)
```

```
## # A tibble: 6 × 5
##   country                     iso2  iso3  year  mean_global_surface_temp
##   <chr>                       <chr> <chr> <chr>                    <dbl>
## 1 Afghanistan, Islamic Rep. of AF    AFG   1961                    -0.113
## 2 Afghanistan, Islamic Rep. of AF    AFG   1962                    -0.164
## 3 Afghanistan, Islamic Rep. of AF    AFG   1963                     0.847
## 4 Afghanistan, Islamic Rep. of AF    AFG   1964                    -0.764
## 5 Afghanistan, Islamic Rep. of AF    AFG   1965                    -0.244
## 6 Afghanistan, Islamic Rep. of AF    AFG   1966                     0.226
```

```
annual_mean_global_surface_temp <-
  annual_mean_global_surface_temp[order(annual_mean_global_surface_temp$iso3), ]
write_csv(
  annual_mean_global_surface_temp,
  file = "Climate-main/data/tidy/annual_mean_global_surface_temp.csv"
)
```

CO2 Concentration

The CO2 concentration dataset is read from a CSV file located at “data/original/Carbon Dioxide Concentration.csv”. After initial inspection and removal of duplicate columns, column names are renamed for clarity, and column names are cleaned. The resulting tidy dataset is saved into a CSV file named “carbon_dioxide_concentration.csv” in the “data/tidy” directory.

```
carbon_dioxide_concentration <-
  read_csv("Climate-main/data/original/Carbon Dioxide Concentration.csv",
    show_col_types = FALSE
  )
```

```
## New names:
## • `` -> `...5`
## • `` -> `...6`
```

```
head(carbon_dioxide_concentration)
```

```
## # A tibble: 6 × 6
##   Entity      Code   Year `Annual CO₂ emissions` ...5  ...6
##   <chr>       <chr> <dbl>                  <dbl> <lgl> <chr>
## 1 Afghanistan AFG    1949                  14656 NA    tonnes
## 2 Afghanistan AFG    1950                  84272 NA    <NA>
## 3 Afghanistan AFG    1951                  91600 NA    <NA>
## 4 Afghanistan AFG    1952                  91600 NA    <NA>
## 5 Afghanistan AFG    1953                 106256 NA    <NA>
## 6 Afghanistan AFG    1954                 106256 NA    <NA>
```

```
carbon_dioxide_concentration <- carbon_dioxide_concentration |>
  rename(
    iso3 = "Code",
    country = "Entity",
    co2_emissions = "Annual CO₂ emissions"
  ) |>
  clean_names()
head(carbon_dioxide_concentration)
```

```
## # A tibble: 6 × 6
##   country     iso3   year co2_emissions x5    x6
##   <chr>       <chr> <dbl>         <dbl> <lgl> <chr>
## 1 Afghanistan AFG    1949         14656 NA    tonnes
## 2 Afghanistan AFG    1950         84272 NA    <NA>
## 3 Afghanistan AFG    1951         91600 NA    <NA>
## 4 Afghanistan AFG    1952         91600 NA    <NA>
## 5 Afghanistan AFG    1953        106256 NA    <NA>
## 6 Afghanistan AFG    1954        106256 NA    <NA>
```

```
carbon_dioxide_concentration <-
  carbon_dioxide_concentration[order(carbon_dioxide_concentration$iso3), ]
write_csv(
  carbon_dioxide_concentration,
  file = "Climate-main/data/tidy/carbon_dioxide_concentration.csv"
)
```

Annual Greenhouse Gas Emission

The annual greenhouse gas emission dataset is read from an Excel file located at “data/original/Annual Greenhouse Gas Emission.xlsx”. After renaming columns and reshaping the dataset to long format to have a single column for the year, the dataset is cleaned, and the resulting tidy dataset is saved into a CSV file named “annual_greenhouse_gas_emission.csv” in the “data/tidy” directory.

```
annual_greenhouse_gas_emission <-
  read_excel("Climate-main/data/original/Annual Greenhouse Gas Emission.xlsx",
    sheet = "GHG_totals_by_country"
  )
```

```
annual_greenhouse_gas_emission <- annual_greenhouse_gas_emission |>
  rename(iso3 = "EDGAR Country Code", ) |>
  pivot_longer(
    cols = matches("^\\d{4}$"),
    names_to = "year",
    values_to = "ghg_emissions"
  ) |>
  clean_names()
```

```
annual_greenhouse_gas_emission <-
  annual_greenhouse_gas_emission[order(annual_greenhouse_gas_emission$iso3), ]
write_csv(
  annual_greenhouse_gas_emission,
  file = "Climate-main/data/tidy/annual_greenhouse_gas_emission.csv"
)
```

Merging all the datasets

Finally, all the cleaned datasets are merged into a single dataset by common columns such as “iso3” and “year”. Various datasets are read from CSV files located in the “data/tidy” directory. After merging and cleaning column names, the resulting merged dataset is saved into a CSV file named “merged.csv” in the “data/tidy” directory.

```
forestarea <- read_csv("Climate-main/data/tidy/forestarea.csv")
```

```
## Rows: 6815 Columns: 4
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (2): country, iso3
## dbl (2): year, forest_area
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
carbonstocks <- read_csv("Climate-main/data/tidy/carbonstocks.csv")
```

```
## Rows: 6032 Columns: 4
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (2): country, iso3
## dbl (2): year, carbonstock
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
landarea <- read_csv("Climate-main/data/tidy/landarea.csv")
```

```
## Rows: 6815 Columns: 4
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (2): country, iso3
## dbl (2): year, landarea
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
air_quality <- read_csv("Climate-main/data/tidy/air_quality.csv")
```

```
## Rows: 5497 Columns: 4
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (2): iso3, country
## dbl (2): year, air_quality
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
annual_greenhouse_gas_emission <-
  read_csv("Climate-main/data/tidy/annual_greenhouse_gas_emission.csv")
```

```
## Rows: 11342 Columns: 4
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (2): iso3, country
## dbl (2): year, ghg_emissions
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
carbon_dioxide_concentration <-
  read_csv("Climate-main/data/tidy/carbon_dioxide_concentration.csv")
```

```
## Rows: 30308 Columns: 6
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (2): country, iso3
## dbl (2): year, co2_emissions
## lgl (2): x5, x6
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
disaster_frequency <-
  read_csv("Climate-main/data/tidy/disaster_frequency.csv")
```

```
## Rows: 9245 Columns: 6
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (4): country, iso2, iso3, indicator
## dbl (2): year, disaster_count
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
annual_mean_global_surface_temp <-
  read_csv("Climate-main/data/tidy/annual_mean_global_surface_temp.csv")
```

```
## Rows: 13950 Columns: 5
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (3): country, iso2, iso3
## dbl (2): year, mean_global_surface_temp
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
merged_data <-
  merge(landarea,
    forestarea,
    by = c("iso3", "year"),
    all = FALSE
  ) |>
  select(-c("country.y")) |>
  rename("country" = "country.x") |>
  merge(carbonstocks, by = c("iso3", "year"), all = FALSE) |>
  select(-c("country.y")) |>
  rename("country" = "country.x") |>
  merge(air_quality, by = c("iso3", "year"), all = FALSE) |>
  select(-c("country.y")) |>
  rename("country" = "country.x") |>
  merge(annual_greenhouse_gas_emission,
    by = c("iso3", "year"),
    all = FALSE
  ) |>
  select(-c("country.y")) |>
  rename("country" = "country.x") |>
  merge(carbon_dioxide_concentration,
    by = c("iso3", "year"),
    all = FALSE
  ) |>
  select(-c("country.y")) |>
  rename("country" = "country.x") |>
  merge(disaster_frequency,
    by = c("iso3", "year"),
    all = FALSE
  ) |>
  select(-c("country.y")) |>
  rename("country" = "country.x") |>
  merge(annual_mean_global_surface_temp,
    by = c("iso3", "year"),
    all = FALSE
  ) |>
  select(-c("country.y", "iso2.y", "x5", "x6", "indicator")) |>
  rename(
    "country" = "country.x",
    "iso2" = "iso2.x",
    "land_area (in 1000 HA)" = "landarea",
    "forest_area (in 1000 HA)" = "forest_area",
    "carbon_stocks (in millions tonnes)" = "carbonstock",
    "air_quality (in  ug per m3)" = "air_quality",
    "ghg_emissions (in million metric tonnes of CO2 equivalents per yr)" = "ghg_emissions",
    "co2_emissions (in tonnes)" = "co2_emissions",
    "disaster_count (in Number of)" = "disaster_count",
    "mean_global_surface_temp (in celsius)" = "mean_global_surface_temp"
  ) |>
  select("iso2", "iso3", "country", "year", everything()) |>
  filter(year > 1999) |>
  clean_names()

merged_data$mean_global_surface_temp_in_celsius <- na.approx(
  merged_data$mean_global_surface_temp_in_celsius
)
write_csv(merged_data, file = "Climate-main/data/tidy/merged.csv")
head(merged_data)
```

```
##   iso2 iso3 country year land_area_in_1000_ha forest_area_in_1000_ha carbon_stocks_in_millions_tonnes
## 1   AO  AGO  Angola 2000               124670               77708.61                         1342.805
## 2   AO  AGO  Angola 2001               124670               77153.55                         1333.213
## 3   AO  AGO  Angola 2002               124670               76598.49                         1323.622
## 4   AO  AGO  Angola 2003               124670               76043.43                         1314.030
## 5   AO  AGO  Angola 2004               124670               75488.37                         1304.439
## 6   AO  AGO  Angola 2005               124670               74933.30                         1294.848
##   air_quality_in_ug_per_m3 ghg_emissions_in_million_metric_tonnes_of_co2_equivalents_per_yr
## 1                    23.57                                                          71.46134
## 2                    24.27                                                          69.30239
## 3                    23.83                                                          68.05226
## 4                    23.61                                                          68.30399
## 5                    23.56                                                          74.42711
## 6                    24.41                                                          73.53390
##   co2_emissions_in_tonnes disaster_count_in_number_of mean_global_surface_temp_in_celsius
## 1                15995108                           5                               0.169
## 2                15908461                           3                               0.295
## 3                16080002                           1                               0.735
## 4                17484428                           2                               0.889
## 5                17015688                           4                               0.414
## 6                15386819                           1                               1.021
```

The final merged dataset to our project is created.

Normalization & Visualization

To ensure comparability across different variables, normalization techniques such as min-max scaling or z-score standardization can be applied. Additionally, visualizing the time series data and forecasts can provide insights into trends and patterns. This can be done using line plots, bar plots, or interactive visualizations.

Below section of the document focuses on the Normalization and Visualisation of the raw merged dataset created in the above section.

Normalisation

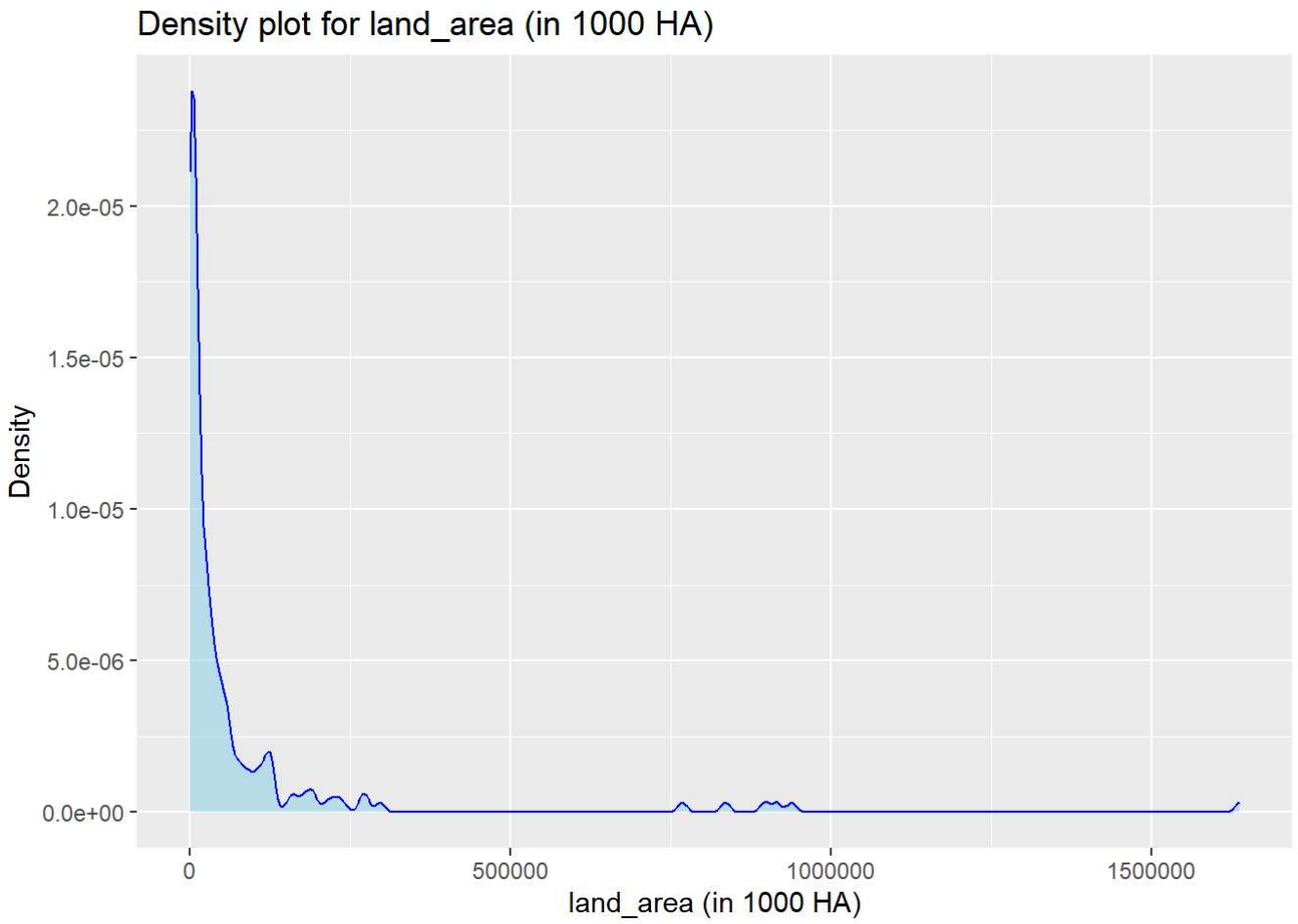
Normalization is a data transformation process that aligns data values to a common scale or distribution of values so that.

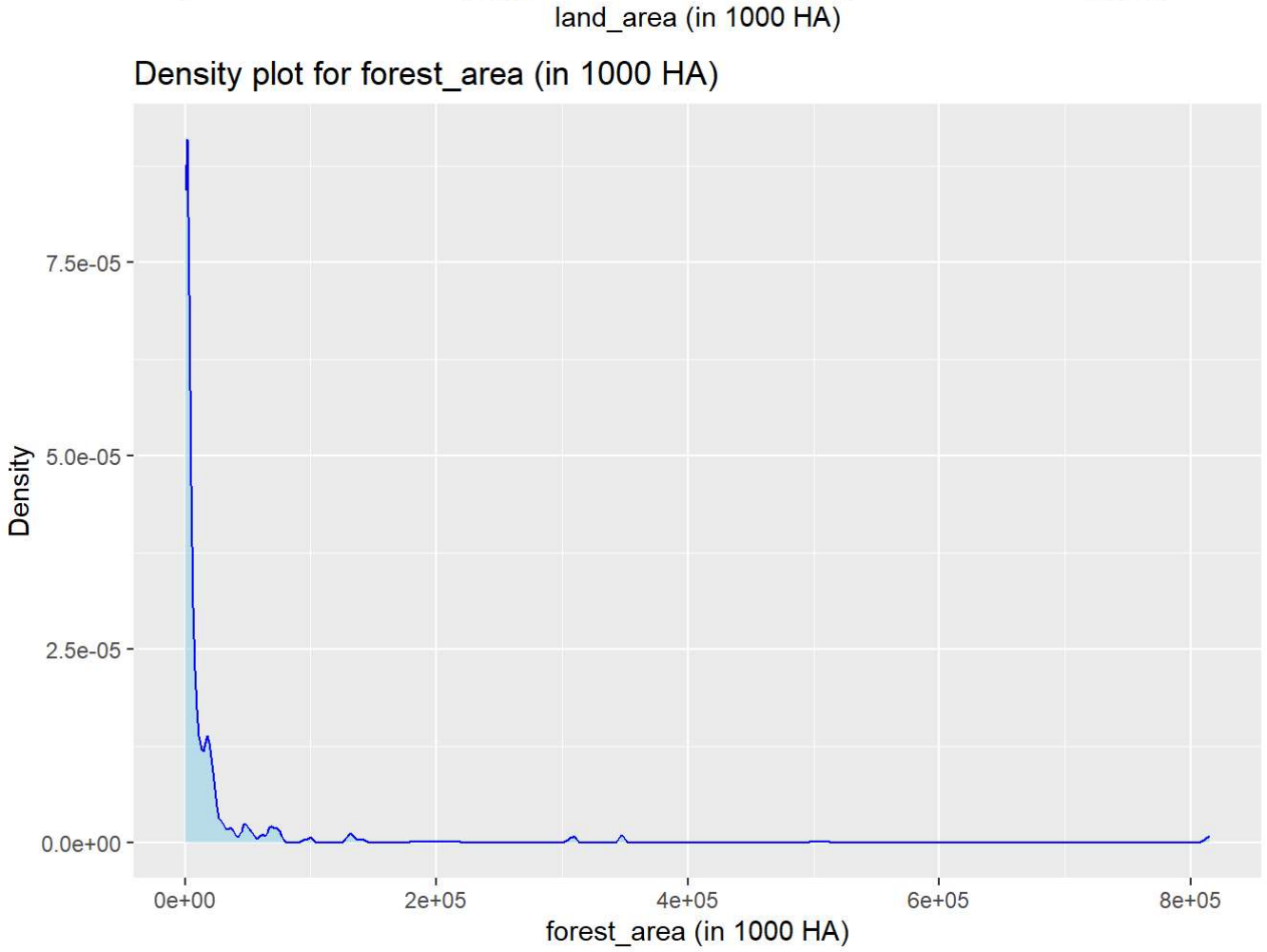
Data Before Normalisation

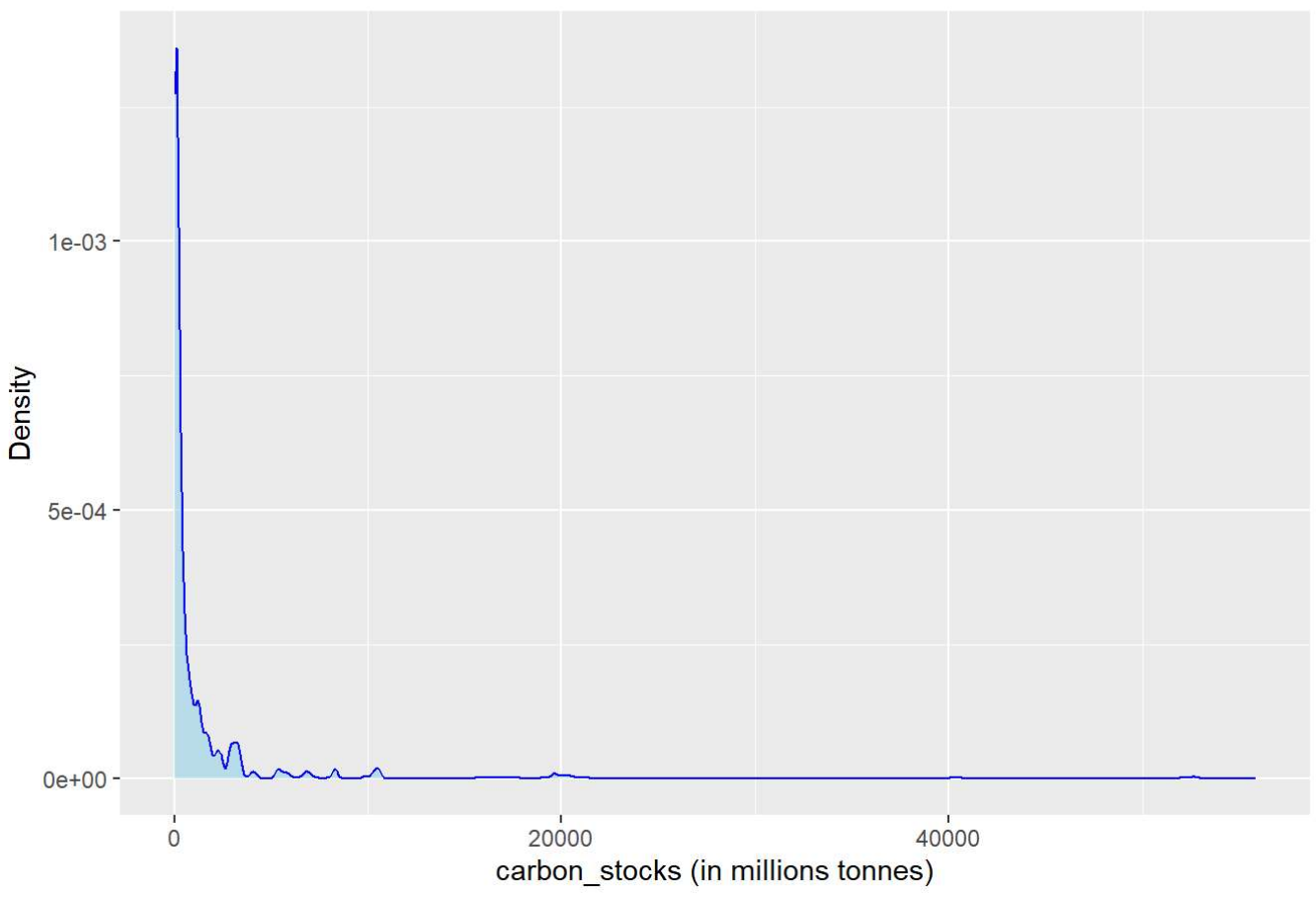
```
library(ggplot2)
merged_data <- read_csv("merged.csv")
```

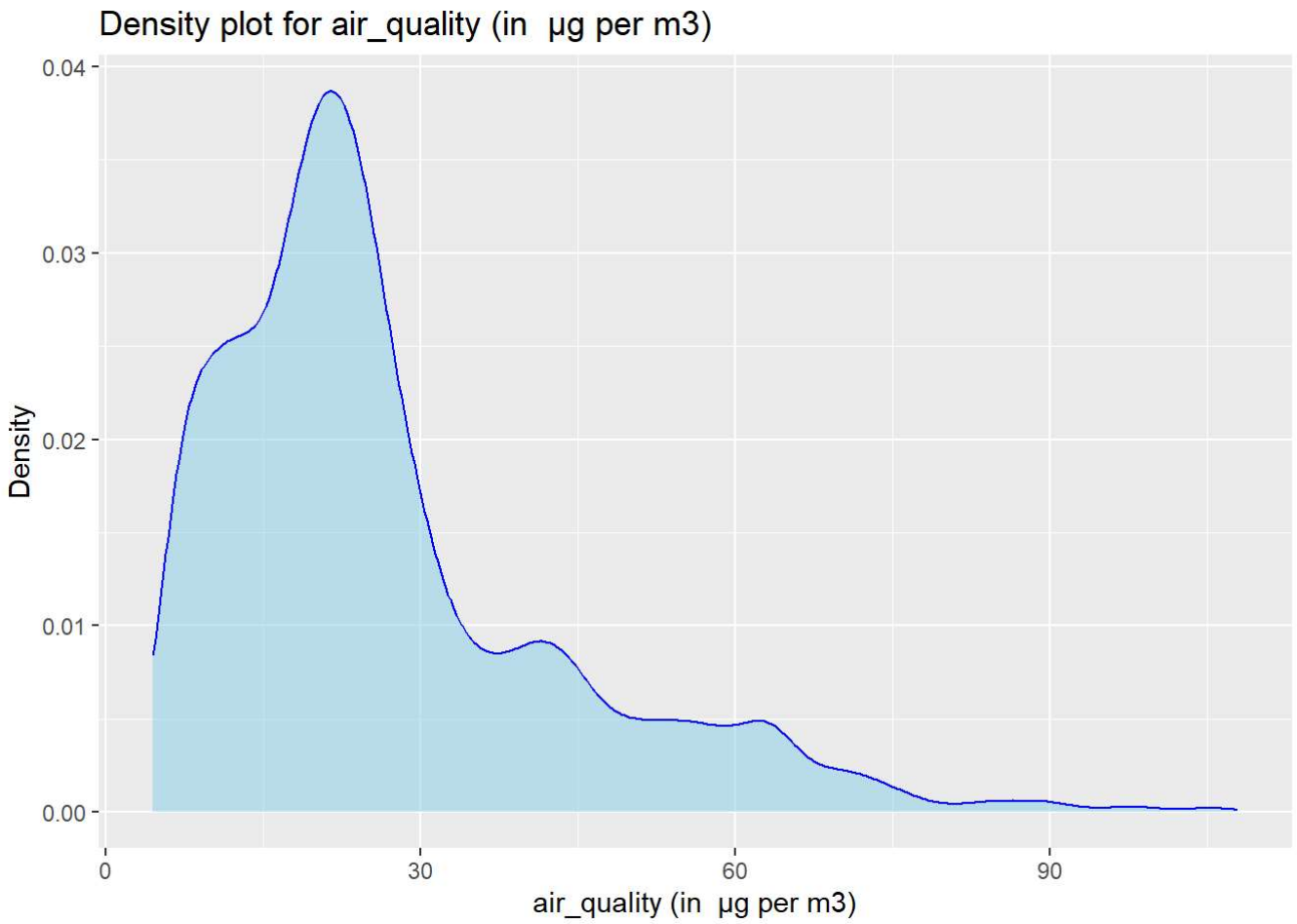
```
## Rows: 3561 Columns: 12
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (3): iso2, iso3, country
## dbl (9): year, land_area (in 1000 HA), forest_area (in 1000 HA), carbon_stocks (in millions tonnes),...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

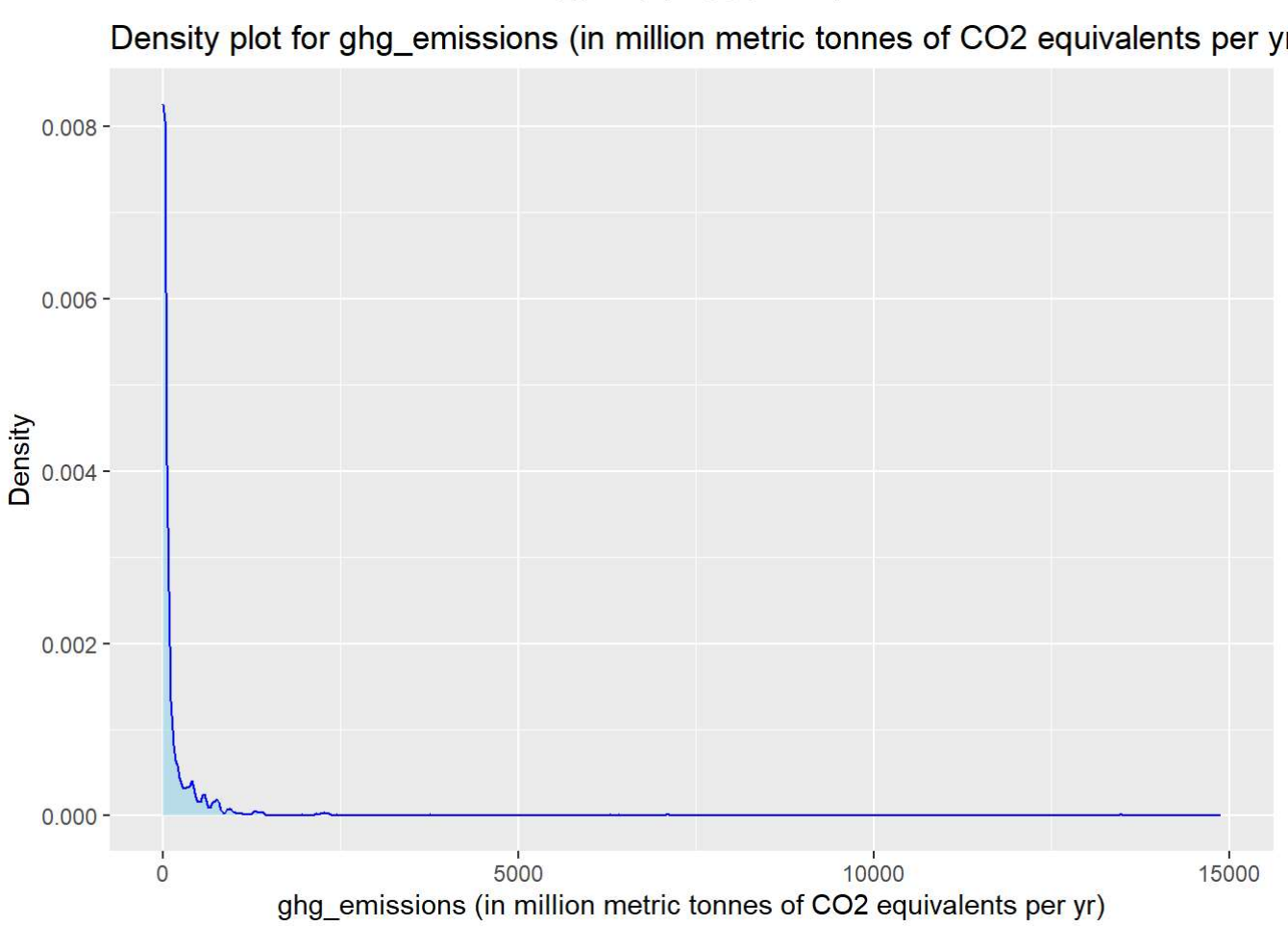
```
# Define relevant columns for the first set of density plots
relevant_cols <- c("land_area (in 1000 HA)", "forest_area (in 1000 HA)",
                   "carbon_stocks (in millions tonnes)", "air_quality (in  µg per m3)",
                   "ghg_emissions (in million metric tonnes of CO2 equivalents per yr)",
                   "co2_emissions (in tonnes)")

merged_data$`land_area (in 1000 HA)`
# Create a list to store density plots
density_plots <- list()

# Loop through relevant columns and create density plots
for (col in relevant_cols) {
    # Create a density plot for the current column
    density_plots[[col]] <- ggplot(merged_data, aes(x = !!sym(col))) +
        geom_density(fill = "skyblue", color = "blue", alpha = 0.5) +
        labs(title = paste("Density plot for", col), x = col, y = "Density")
}

# Display density plots
for (i in seq_along(relevant_cols)) {
    print(density_plots[[i]])
}
```

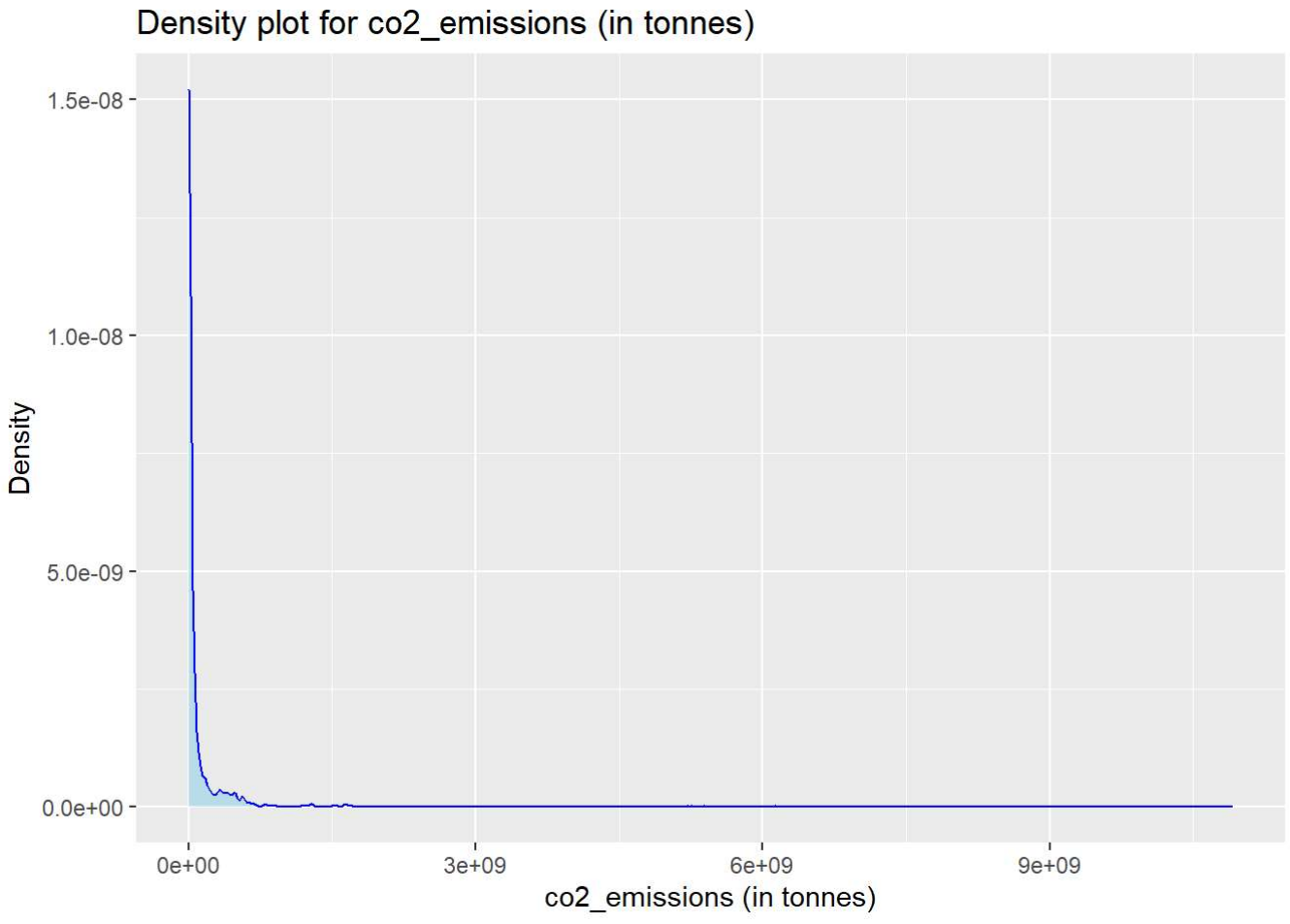

Density plot for land_area (in 1000 HA)
Density
2.0e-05
1.5e-05
1.0e-05
5.0e-06
0.0e+00
0
500000
1000000
1500000
land_area (in 1000 HA)


Density plot for forest_area (in 1000 HA)
Density
7.5e-05
5.0e-05
2.5e-05
0.0e+00
0e+00
2e+05
4e+05
6e+05
8e+05
forest_area (in 1000 HA)


Density plot for carbon_stocks (in millions tonnes)
Density
1e-03
5e-04
0e+00
0
20000
40000
carbon_stocks (in millions tonnes)



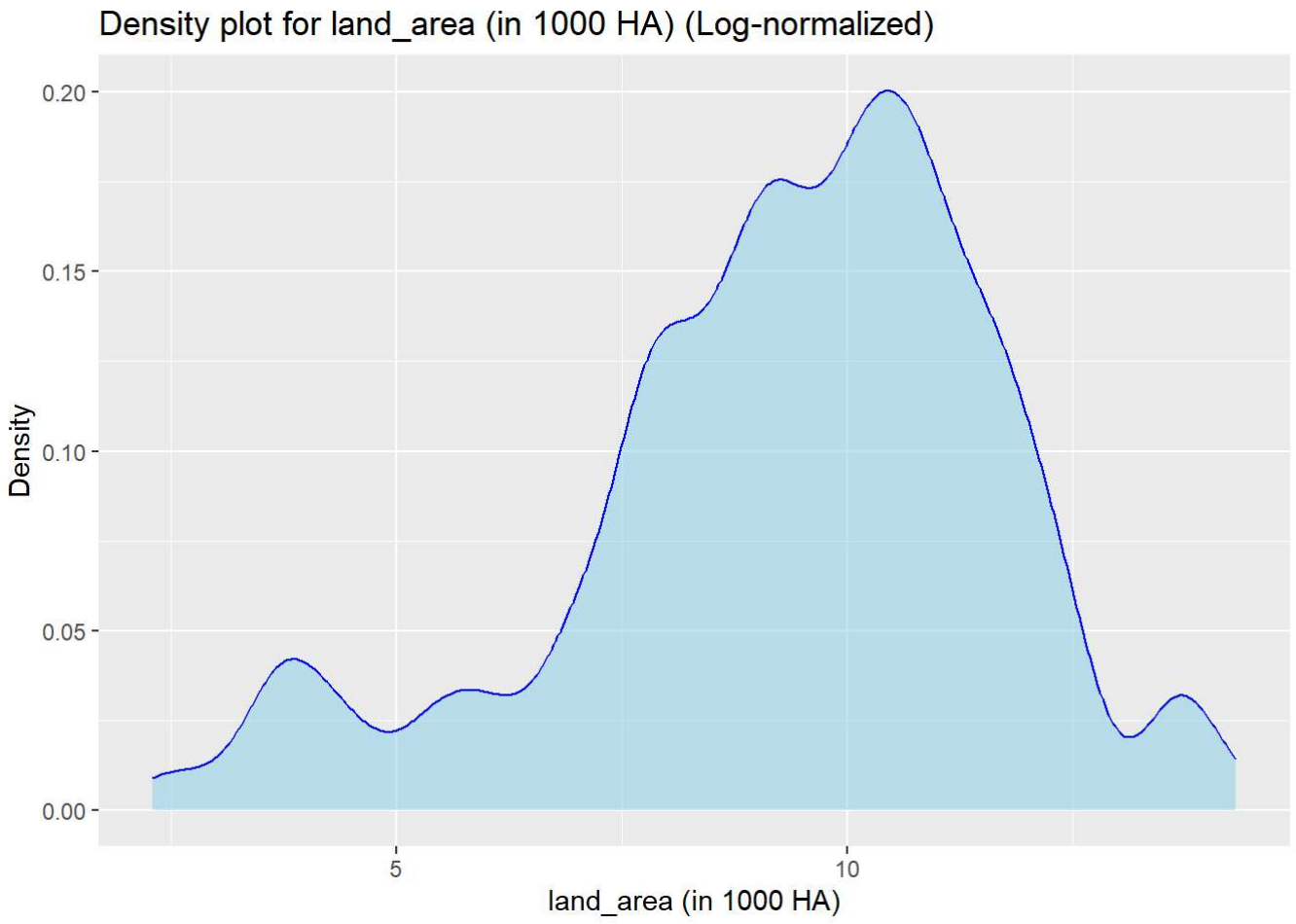


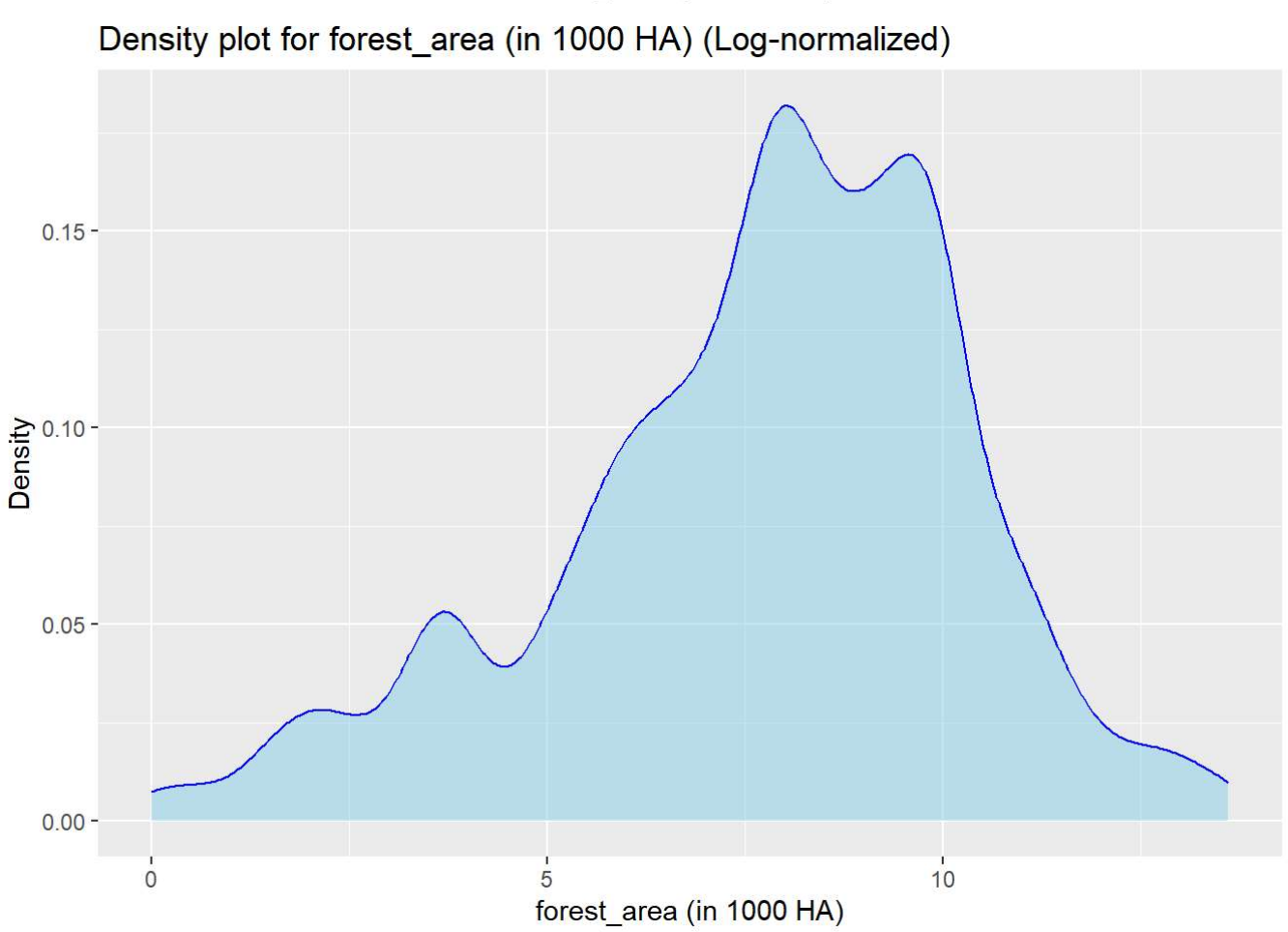


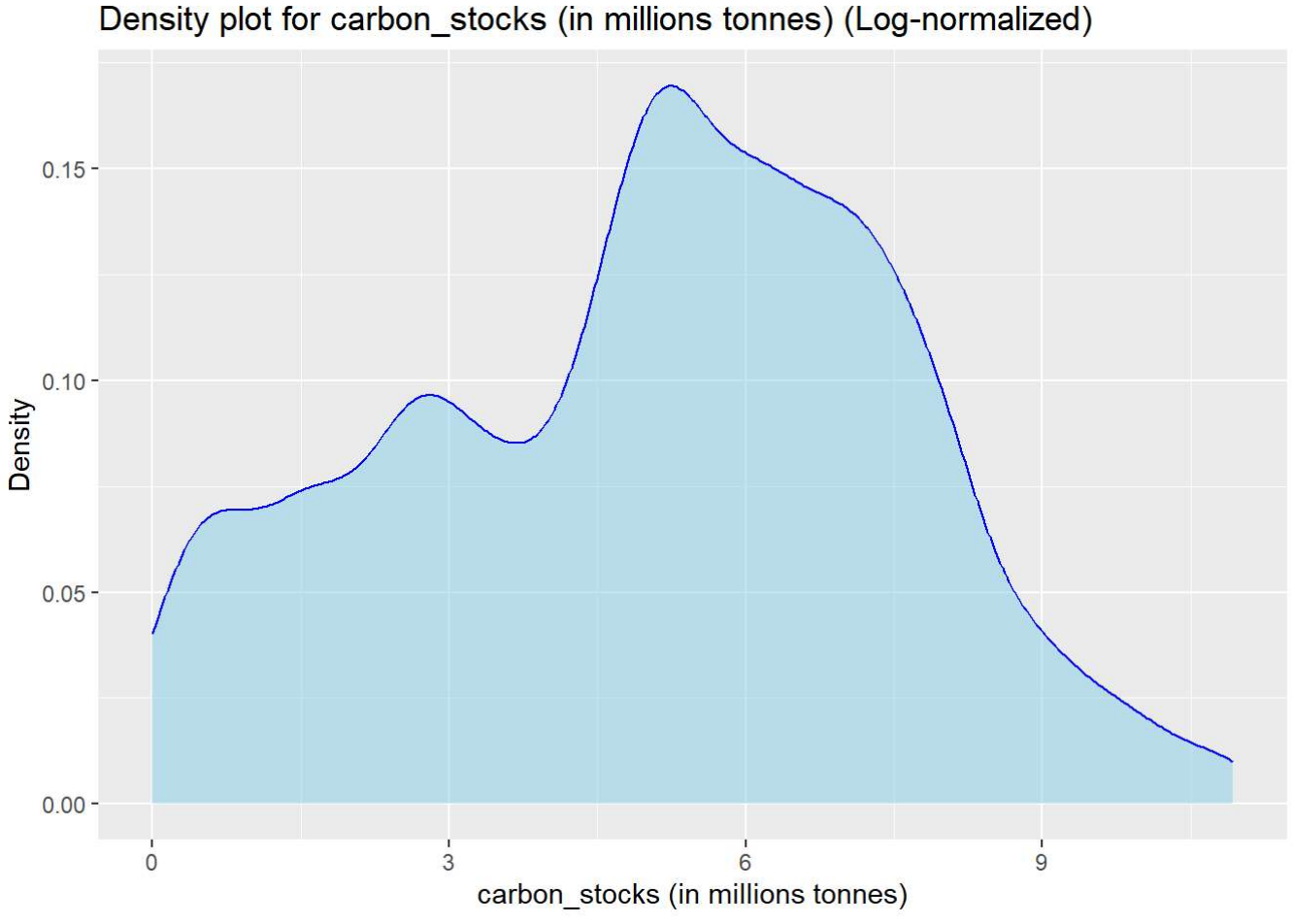
Data After Log Normalisation

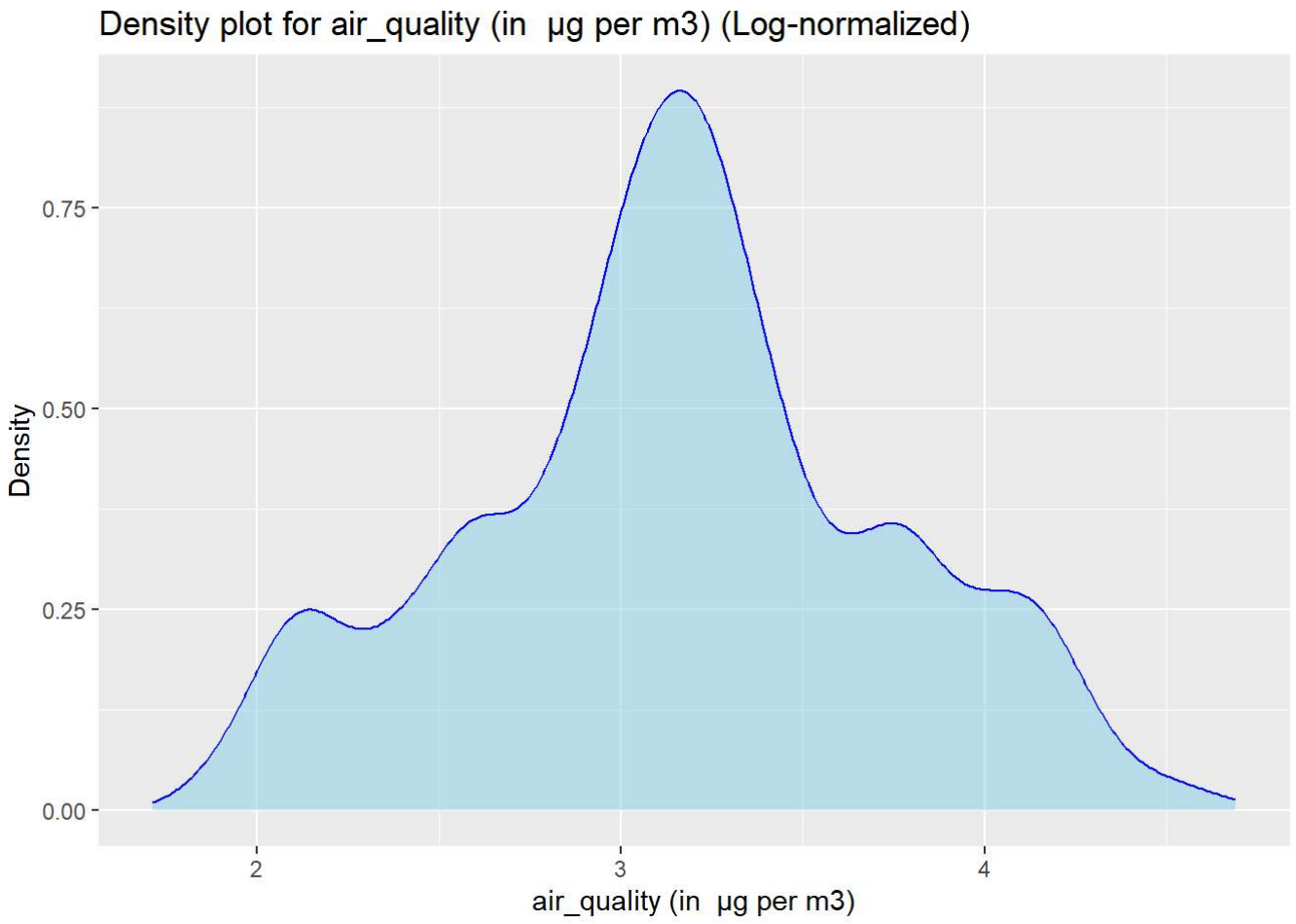
Log normalization is the process of converting each log data field or entry to a standardized data representation and categorizing it consistently.

```r
relevant_cols <- c("land_area (in 1000 HA)", "forest_area (in 1000 HA)",
                   "carbon_stocks (in millions tonnes)", "air_quality (in  µg per m3)",
                   "ghg_emissions (in million metric tonnes of CO2 equivalents per yr)",
                   "co2_emissions (in tonnes)")

# Log normalization function
log_normalization <- function(x) {
    return(log(x + 1))  # Adding 1 to avoid logarithm of 0
}

# Apply log normalization to relevant columns
log_normalized_data <- merged_data
log_normalized_data[relevant_cols] <- lapply(merged_data[relevant_cols], log_normalization)

# Create a list to store density plots for log-normalized data
log_density_plots <- list()

# Loop through relevant columns and create density plots for log-normalized data
for (col in relevant_cols) {
    # Create a density plot for the current column
    log_density_plots[[col]] <- ggplot(log_normalized_data, aes(x = !!sym(col))) +
        geom_density(fill = "skyblue", color = "blue", alpha = 0.5) +
        labs(title = paste("Density plot for", col, "(Log-normalized)"), x = col, y = "Density")
}

# Display density plots for log-normalized data
for (i in seq_along(relevant_cols)) {
    print(log_density_plots[[i]])
}
```

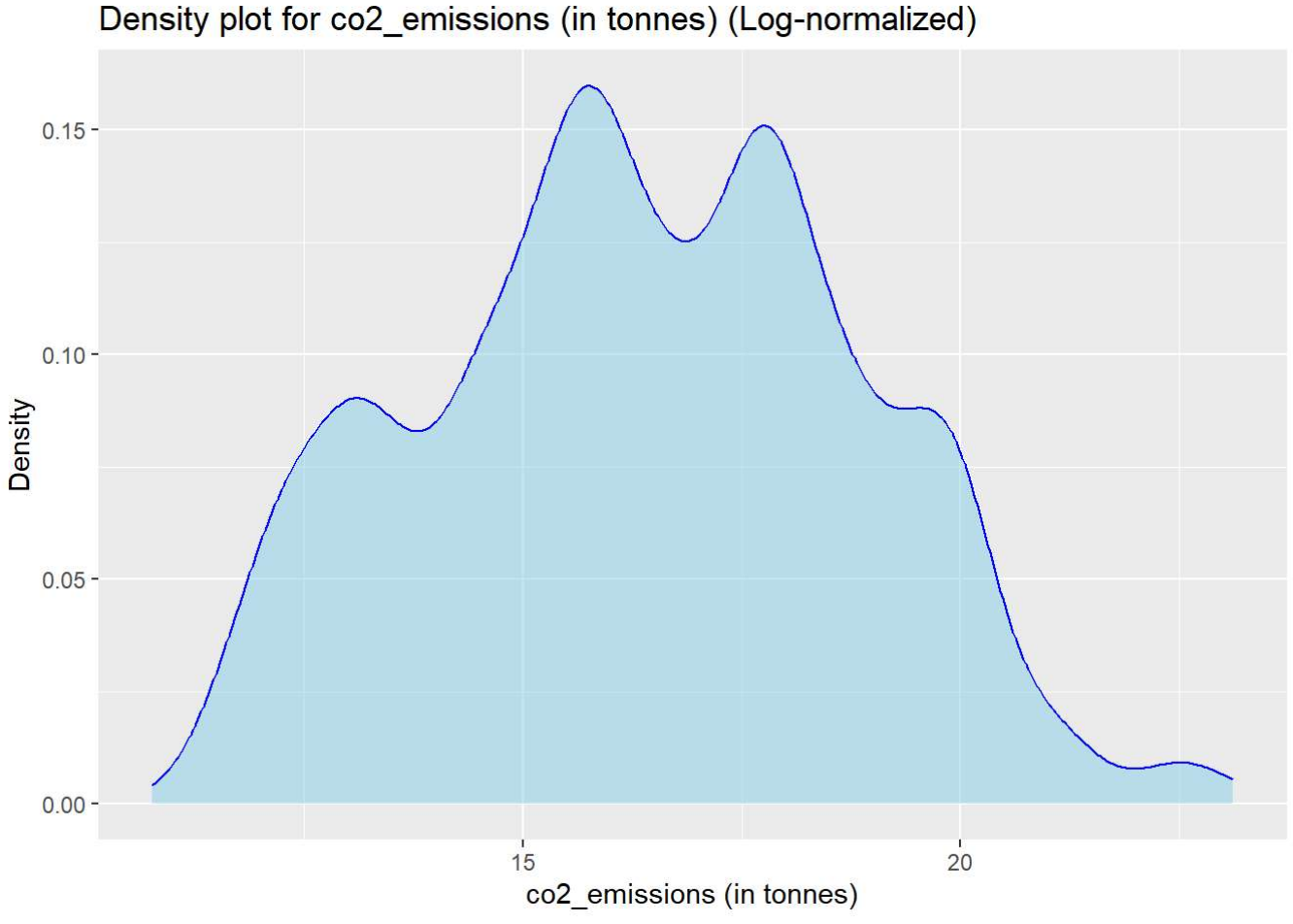

Density plot for land_area (in 1000 HA) (Log-normalized)
Density
0.20
0.15
0.10
0.05
0.00
5
10
land_area (in 1000 HA)


Density plot for forest_area (in 1000 HA) (Log-normalized)
Density
0.15
0.10
0.05
0.00
0
5
10
forest_area (in 1000 HA)


Density plot for carbon_stocks (in millions tonnes) (Log-normalized)
Density
0.15
0.10
0.05
0.00
0
3
6
9
carbon_stocks (in millions tonnes)


Density plot for air_quality (in µg per m3) (Log-normalized)
Density
0.75
0.50
0.25
0.00
2
3
4
air_quality (in µg per m3)

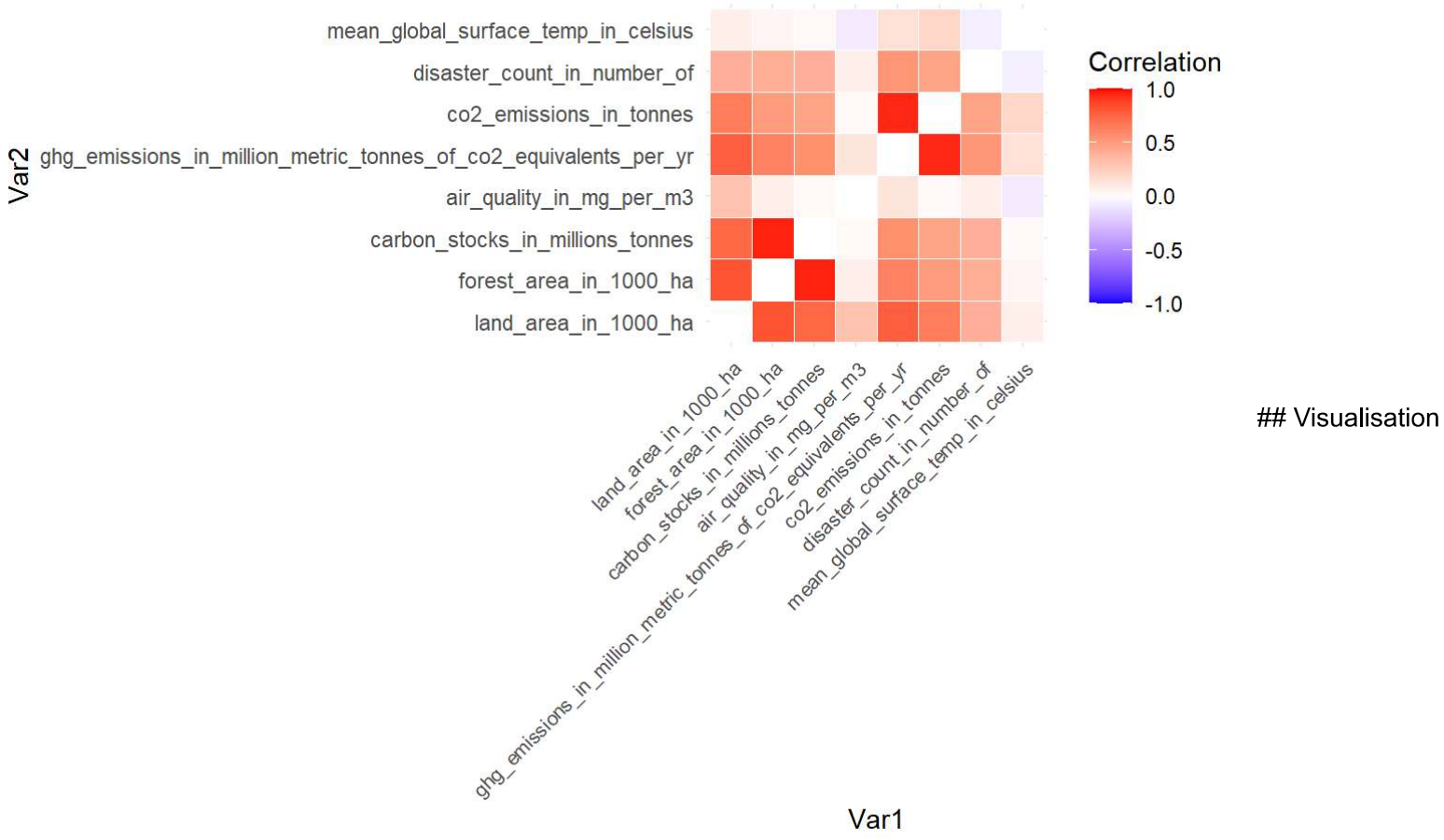
Density plot for ghg_emissions (in million metric tonnes of CO2 equivalents per yr)
Density
0.20
0.15
0.10
0.05
0.00
0.0
2.5
5.0
7.5
10.0
ghg_emissions (in million metric tonnes of CO2 equivalents per yr)


Density plot for co2_emissions (in tonnes) (Log-normalized)
Density
0.15
0.10
0.05
0.00
15
20
co2_emissions (in tonnes)

Corelation between Parameters

Correlation refers to the statistical relationship between the two entities. It measures the extent to which two variables are linearly related.

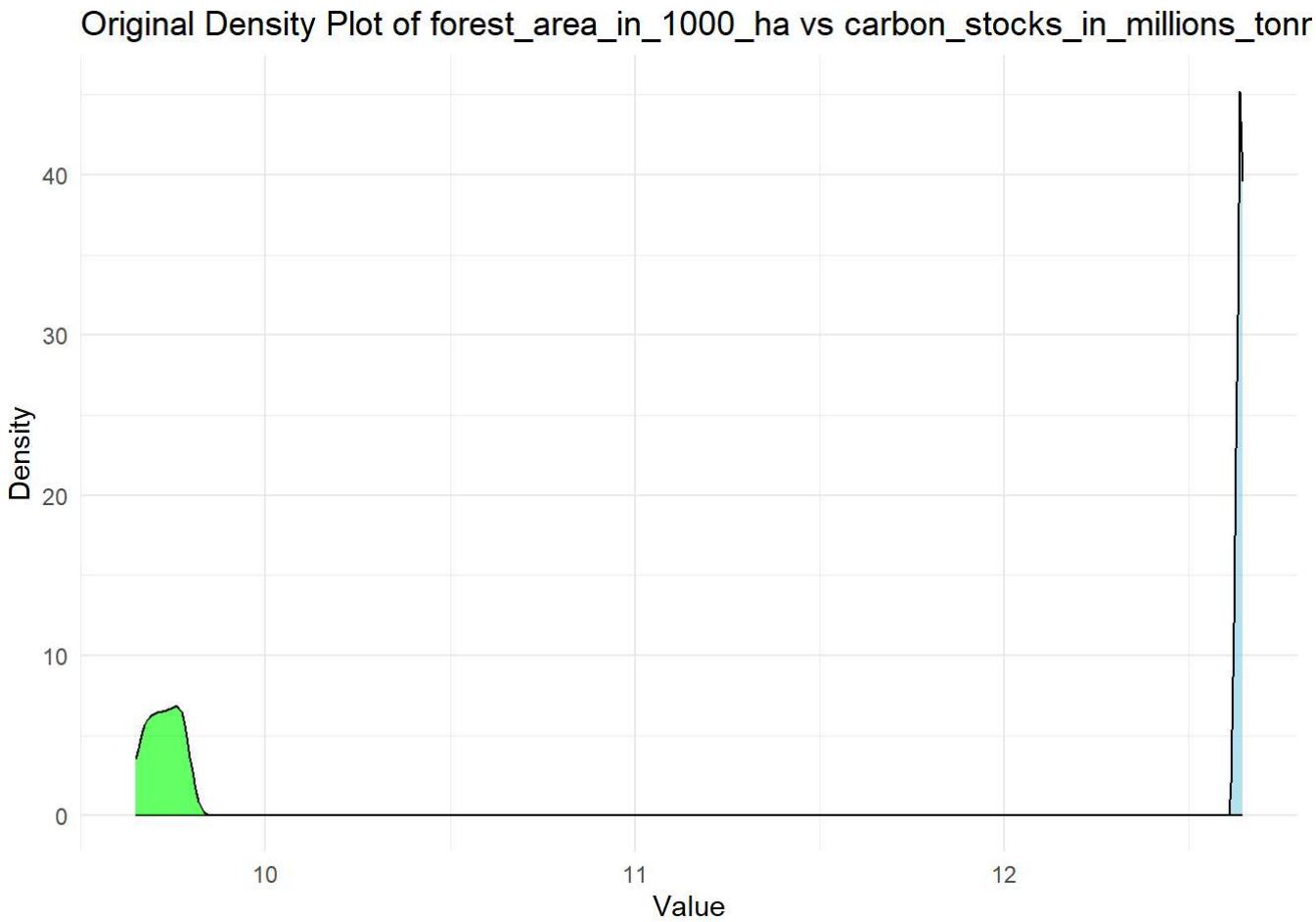
```
column_names <- colnames(log_normalized_data)
column_names
install.packages("janitor")
library(janitor)

log_normalized_data<-clean_names(log_normalized_data)
correlation_matrix_world <- cor(log_normalized_data[, -c(1:4)])  # Exclude non-numeric columns like iso2, iso3, country, yea
r

# Function to identify constant columns
find_constant_columns <- function(data) {
    constant_columns <- sapply(data, function(col) length(unique(col)) == 1)
    constant_column_names <- names(constant_columns)[constant_columns]
    return(constant_column_names)
}

# Get constant column names
constant_columns <- find_constant_columns(log_normalized_data)

# Exclude constant columns and non-numeric columns before normalization
columns_to_exclude <- c(1:4, which(names(log_normalized_data) %in% constant_columns))
normalized_world_data <- scale(log_normalized_data[, -columns_to_exclude])
head(normalized_world_data)
# Calculate correlation matrix for normalized India data
correlation_matrix_world <- cor(normalized_world_data)

# Print correlation matrix
print(correlation_matrix_world)
library(readr)
# Write correlation matrix to a CSV file
write.csv(correlation_matrix_world, file = "correlation_matrix_world_log.csv")

diag(correlation_matrix_world) <- 0



# Load necessary libraries
library(reshape2)
library(ggplot2)

# Plot correlation matrix heatmap
p <- ggplot(data = melt(correlation_matrix_world), aes(Var1, Var2, fill = value)) +
    geom_tile(color = "white") +
    scale_fill_gradient2(low = "blue", high = "red", mid = "white",
                         midpoint = 0, limit = c(-1,1), space = "Lab",
                         name="Correlation") +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 45, vjust = 1, size = 8, hjust = 1)) +
    coord_fixed()

p
```

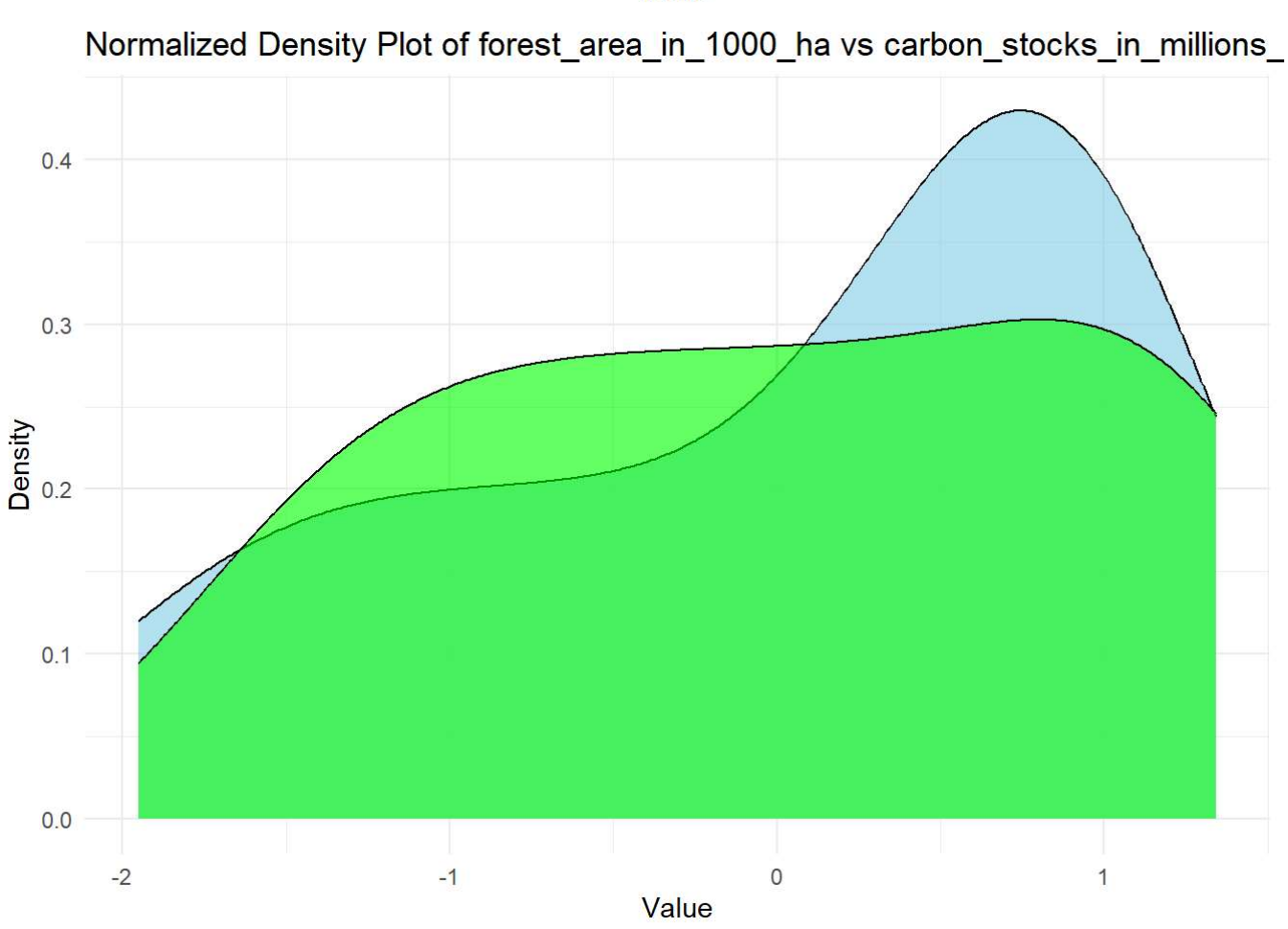


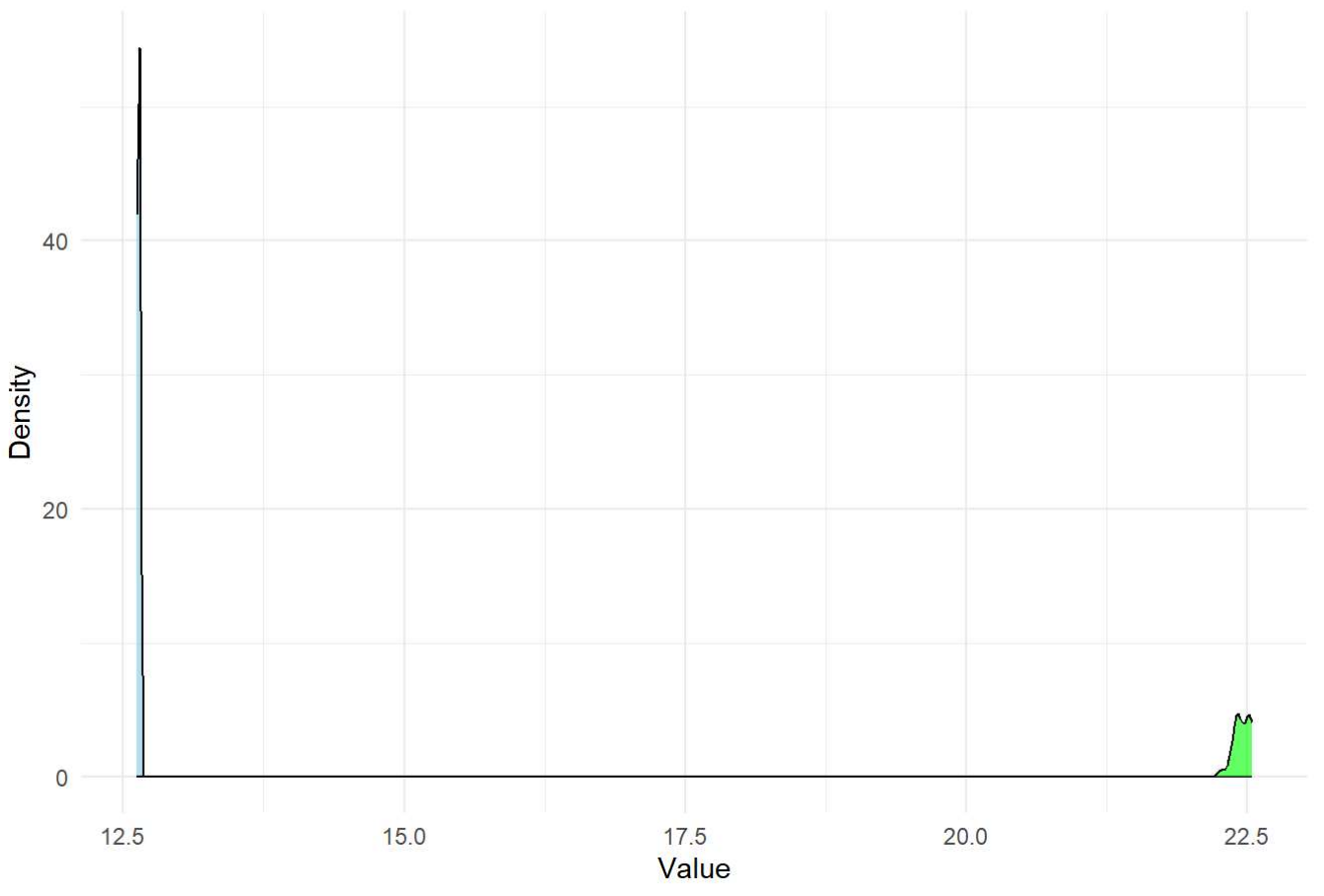
Visualisation

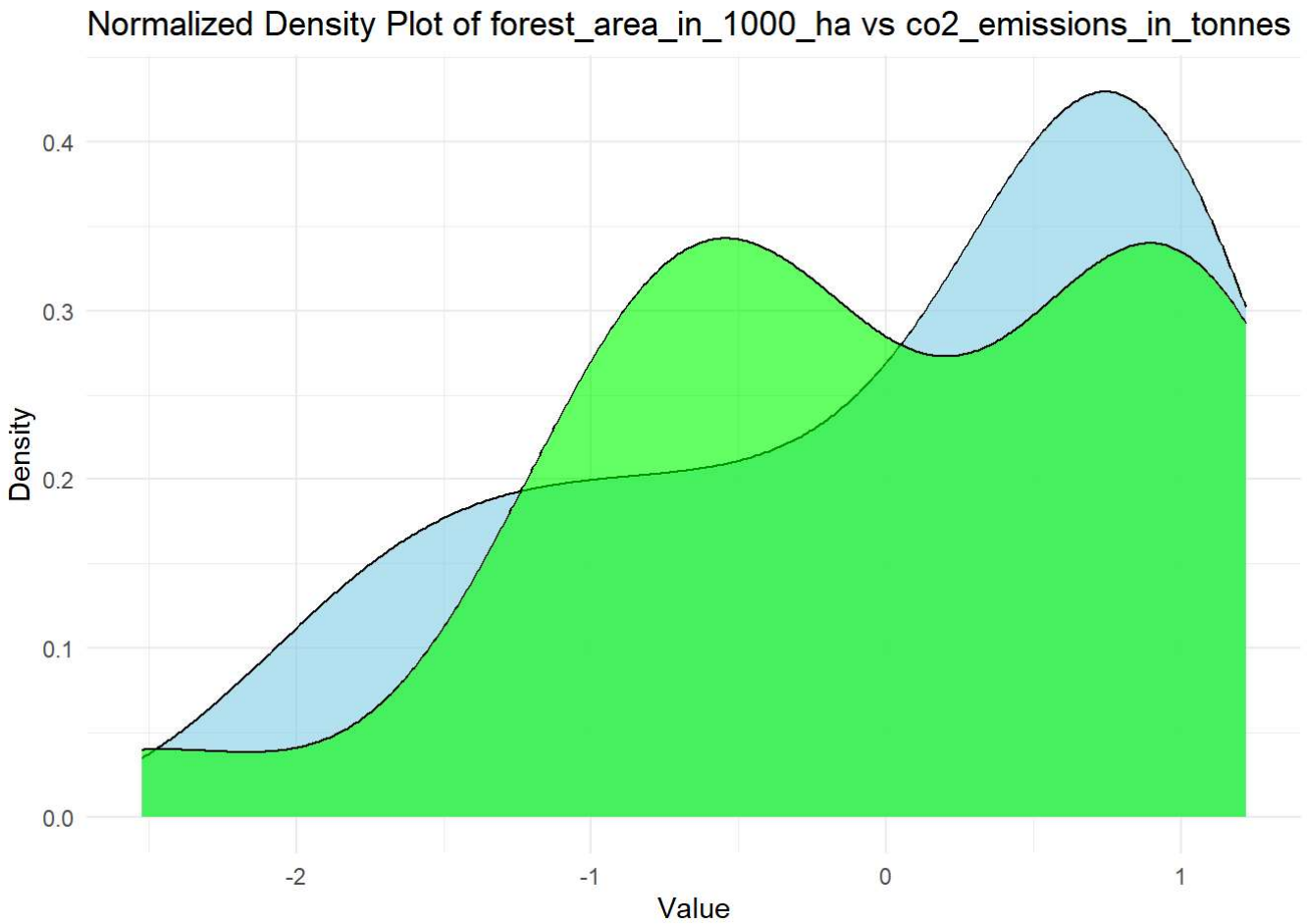
Data visualization is the process of using visual elements like charts, graphs, or maps to represent data. It translates complex, high-volume, or numerical data into a visual representation that is easier to process.

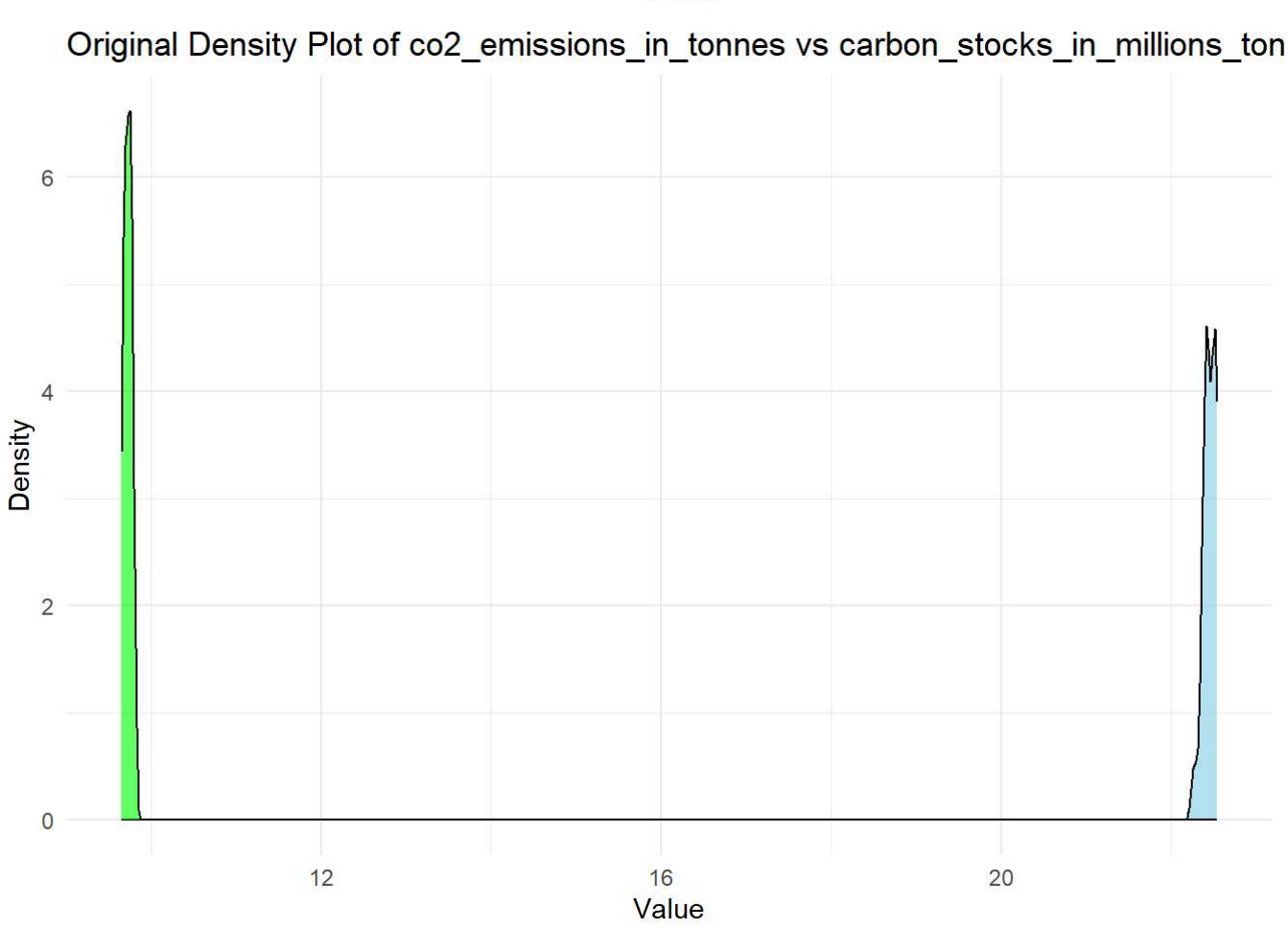
```r
# Load necessary libraries
library(ggplot2)
library(dplyr)
library(gridExtra)

# Read dataset from CSV file
merged <- log_normalized_data

# Define ISO3 value
iso3_value <- "USA"

# Function to create density plot for given column pairs with and without Z-score normalization
create_density_plot <- function(df, col1, col2) {
  # Subset the data for the specified ISO3 value and the columns
  col1_subset <- df %>%
    filter(iso3 == iso3_value) %>%
    pull(col1) %>%
    as.numeric()

  col2_subset <- df %>%
    filter(iso3 == iso3_value) %>%
    pull(col2) %>%
    as.numeric()

  # Remove NA values
  col1_subset <- col1_subset[!is.na(col1_subset)]
  col2_subset <- col2_subset[!is.na(col2_subset)]

  # Function to normalize using Z-score
  normalize_z_score <- function(x) {
    (x - mean(x)) / sd(x)
  }

  # Normalize the data using Z-score
  col1_normalized <- normalize_z_score(col1_subset)
  col2_normalized <- normalize_z_score(col2_subset)

  # Plot the density
  p_original <- ggplot() +
    geom_density(aes(x = col1_subset), fill = "skyblue", alpha = 0.6, color = "black") +
    geom_density(aes(x = col2_subset), fill = "green", alpha = 0.6, color = "black") +
    labs(title = paste("Original Density Plot of", col1, "vs", col2),
         x = "Value",
         y = "Density") +
    scale_fill_manual(values = c("skyblue", "green")) +
    theme_minimal()

  p_normalized <- ggplot() +
    geom_density(aes(x = col1_normalized), fill = "skyblue", alpha = 0.6, color = "black") +
    geom_density(aes(x = col2_normalized), fill = "green", alpha = 0.6, color = "black") +
    labs(title = paste("Normalized Density Plot of", col1, "vs", col2),
         x = "Value",
         y = "Density") +
    scale_fill_manual(values = c("skyblue", "green")) +
    theme_minimal()

  return(list(p_original, p_normalized))
}

# Create density plots for specified column pairs with and without Z-score normalization
plots_list <- list(
  create_density_plot(merged, "forest_area_in_1000_ha", "carbon_stocks_in_millions_tonnes"),
  create_density_plot(merged, "forest_area_in_1000_ha", "co2_emissions_in_tonnes"),
  create_density_plot(merged, "co2_emissions_in_tonnes", "carbon_stocks_in_millions_tonnes"),
  create_density_plot(merged, "co2_emissions_in_tonnes", "ghg_emissions_in_million_metric_tonnes_of_co2_equivalents_per_y
r"),
  create_density_plot(merged, "carbon_stocks_in_millions_tonnes", "forest_area_in_1000_ha"),
  create_density_plot(merged, "ghg_emissions_in_million_metric_tonnes_of_co2_equivalents_per_yr", "forest_area_in_1000_ha"),
  create_density_plot(merged, "ghg_emissions_in_million_metric_tonnes_of_co2_equivalents_per_yr", "co2_emissions_in_tonne
s"),
  create_density_plot(merged, "carbon_stocks_in_millions_tonnes", "ghg_emissions_in_million_metric_tonnes_of_co2_equivalents
_per_yr"),
  create_density_plot(merged, "mean_global_surface_temp_in_celsius", "air_quality_in_mg_per_m3")
)
# Display plots
for (i in 1:length(plots_list)) {
  print(plots_list[[i]][[1]])
  print(plots_list[[i]][[2]])
}
```

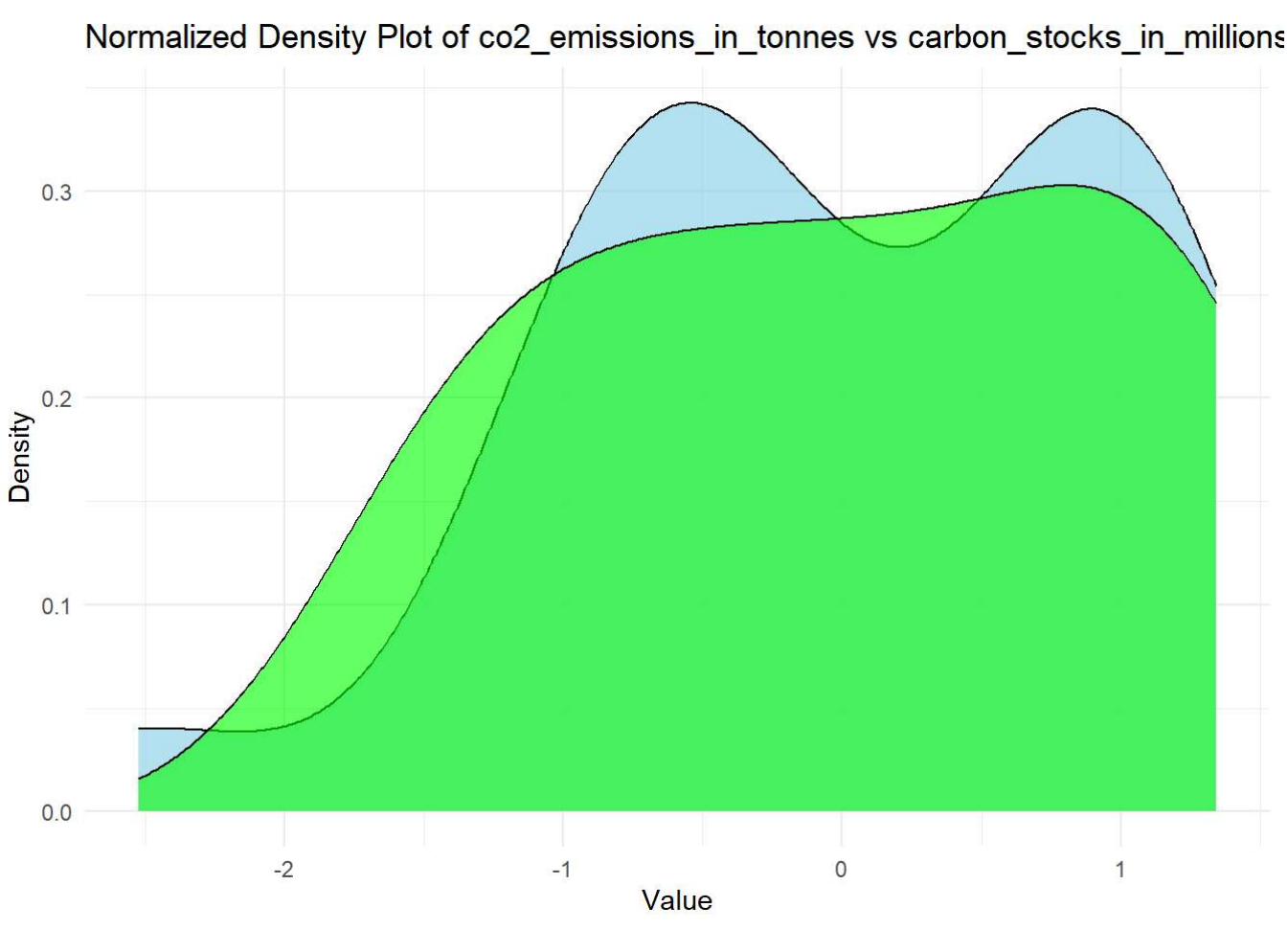

Original Density Plot of forest_area_in_1000_ha vs carbon_stocks_in_millions_tonr
Density
40
30
20
10
0
10
11
12
Value

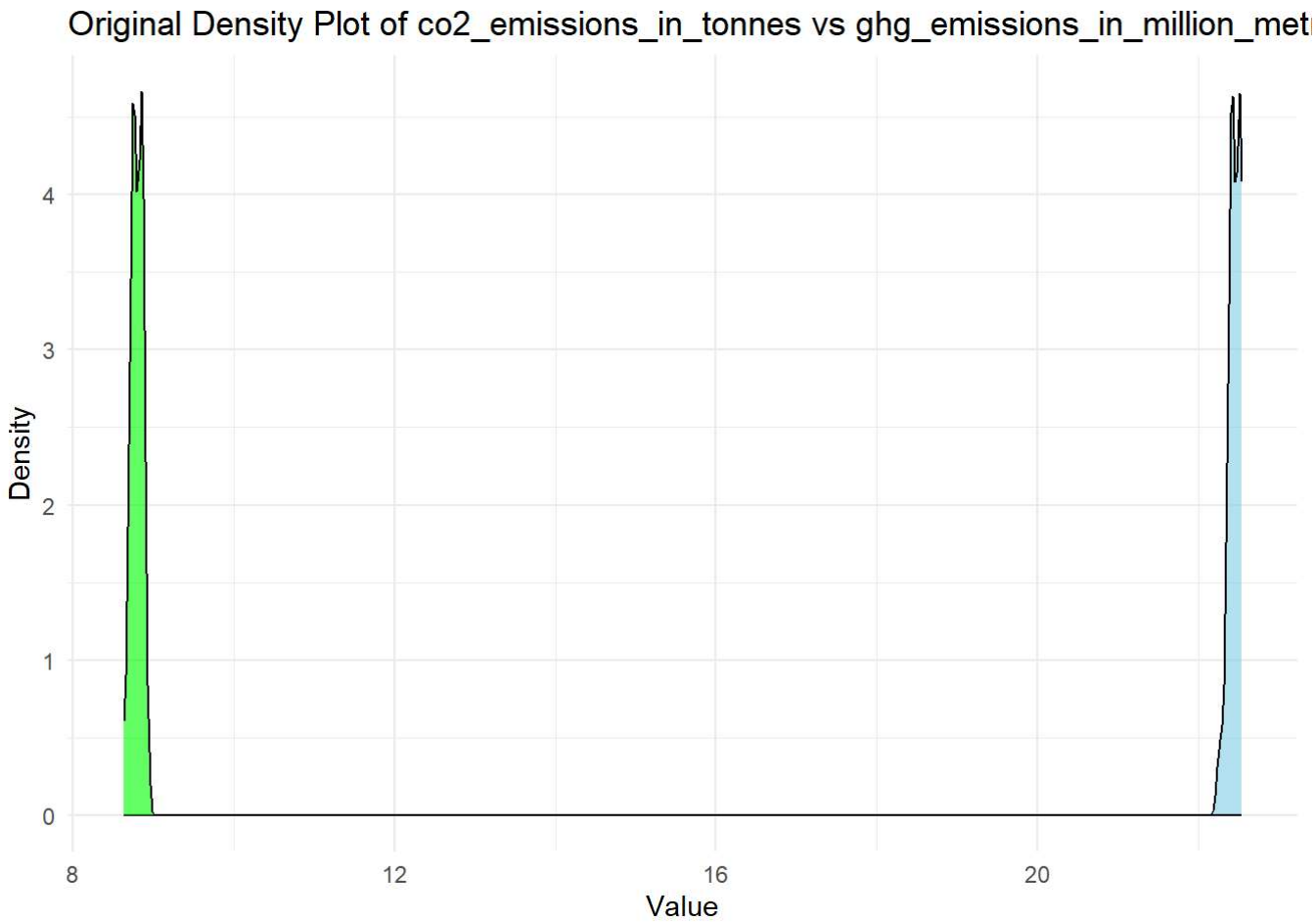

Normalized Density Plot of forest_area_in_1000_ha vs carbon_stocks_in_millions_
Density
0.4
0.3
0.2
0.1
0.0
-2
-1
0
1
Value

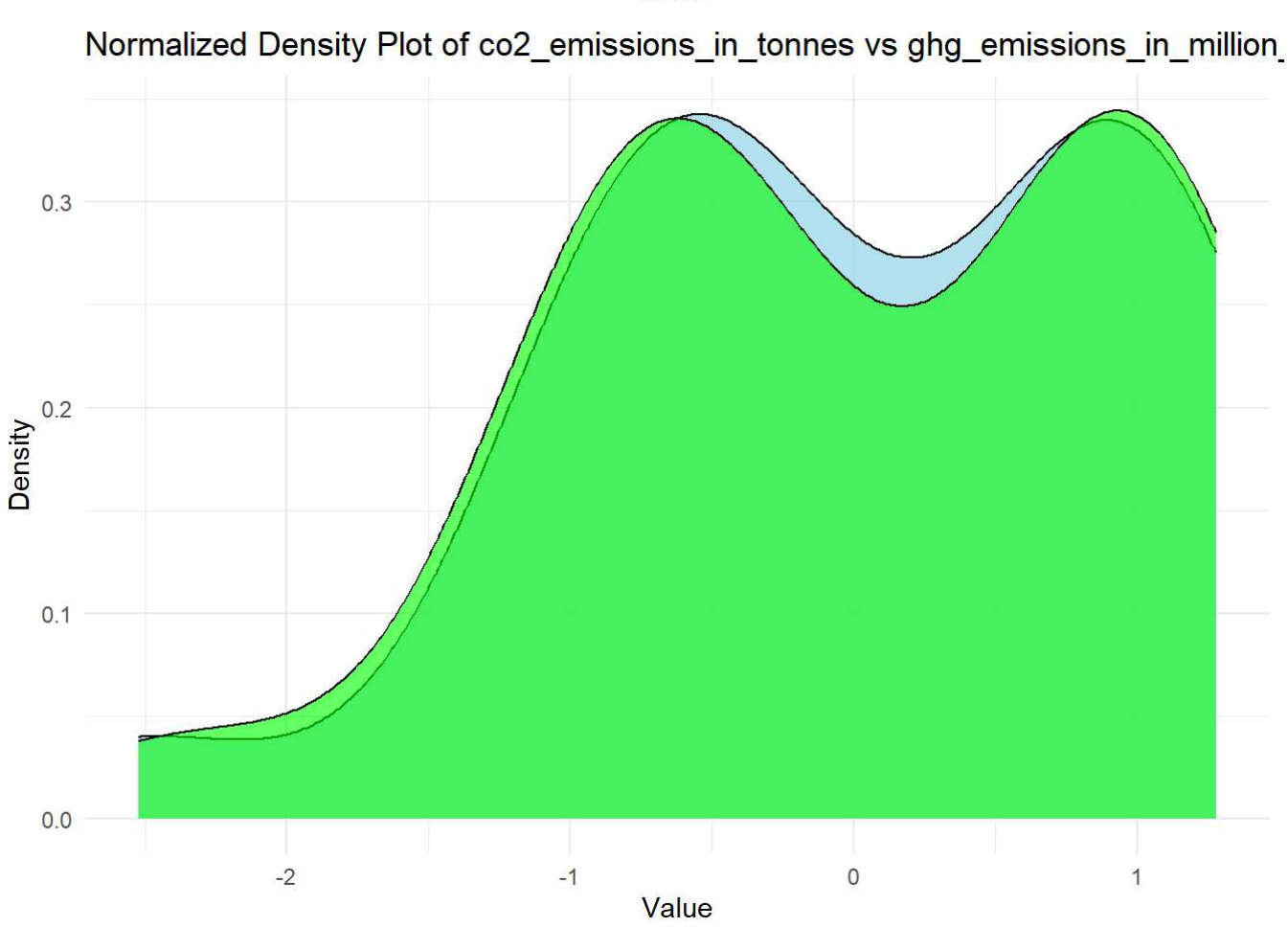

Original Density Plot of forest_area_in_1000_ha vs co2_emissions_in_tonnes
Density
40
20
0
12.5
15.0
17.5
20.0
22.5
Value

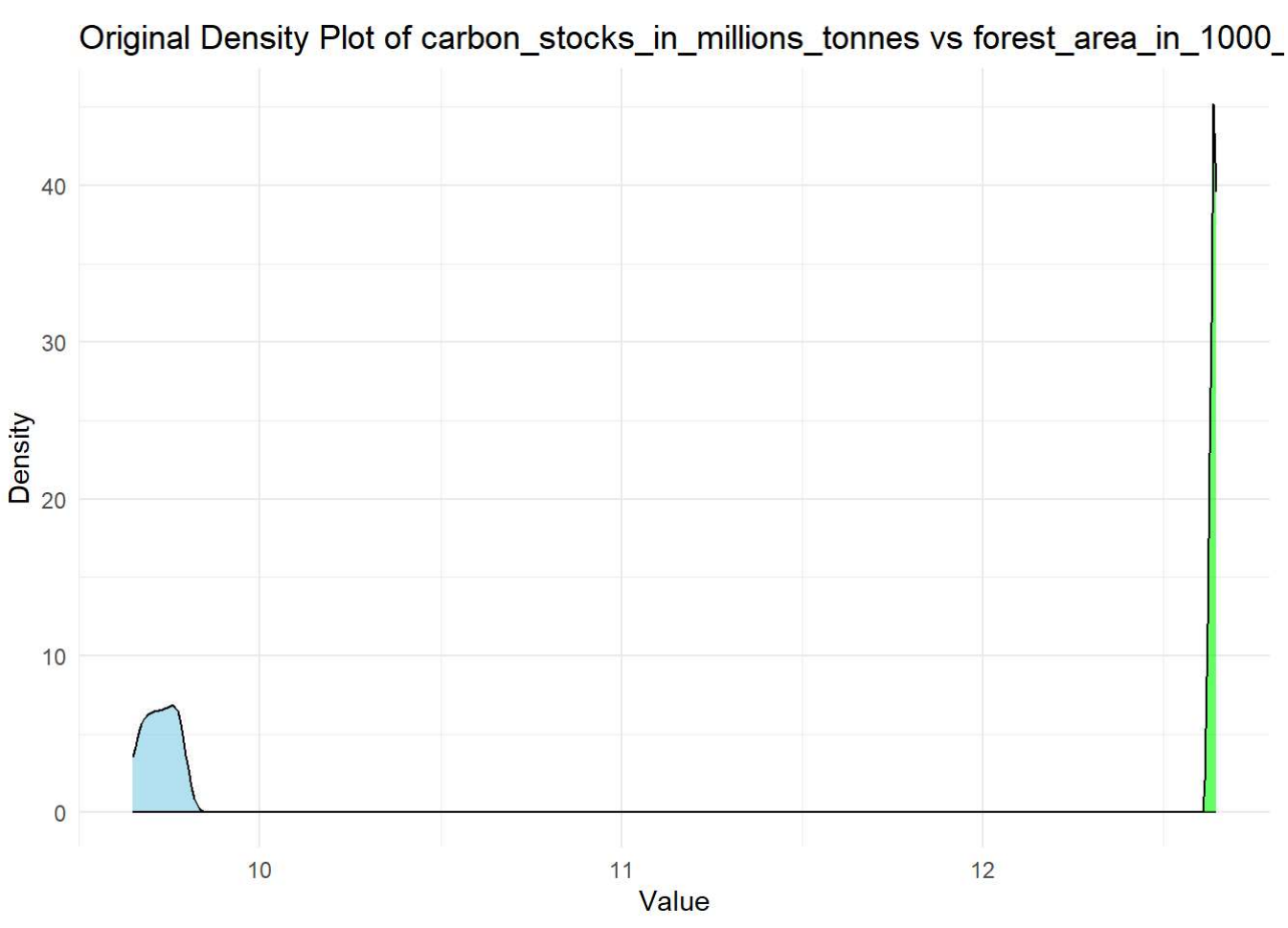

Normalized Density Plot of forest_area_in_1000_ha vs co2_emissions_in_tonnes
Density
0.0
0.1
0.2
0.3
0.4
-2
-1
0
1
Value

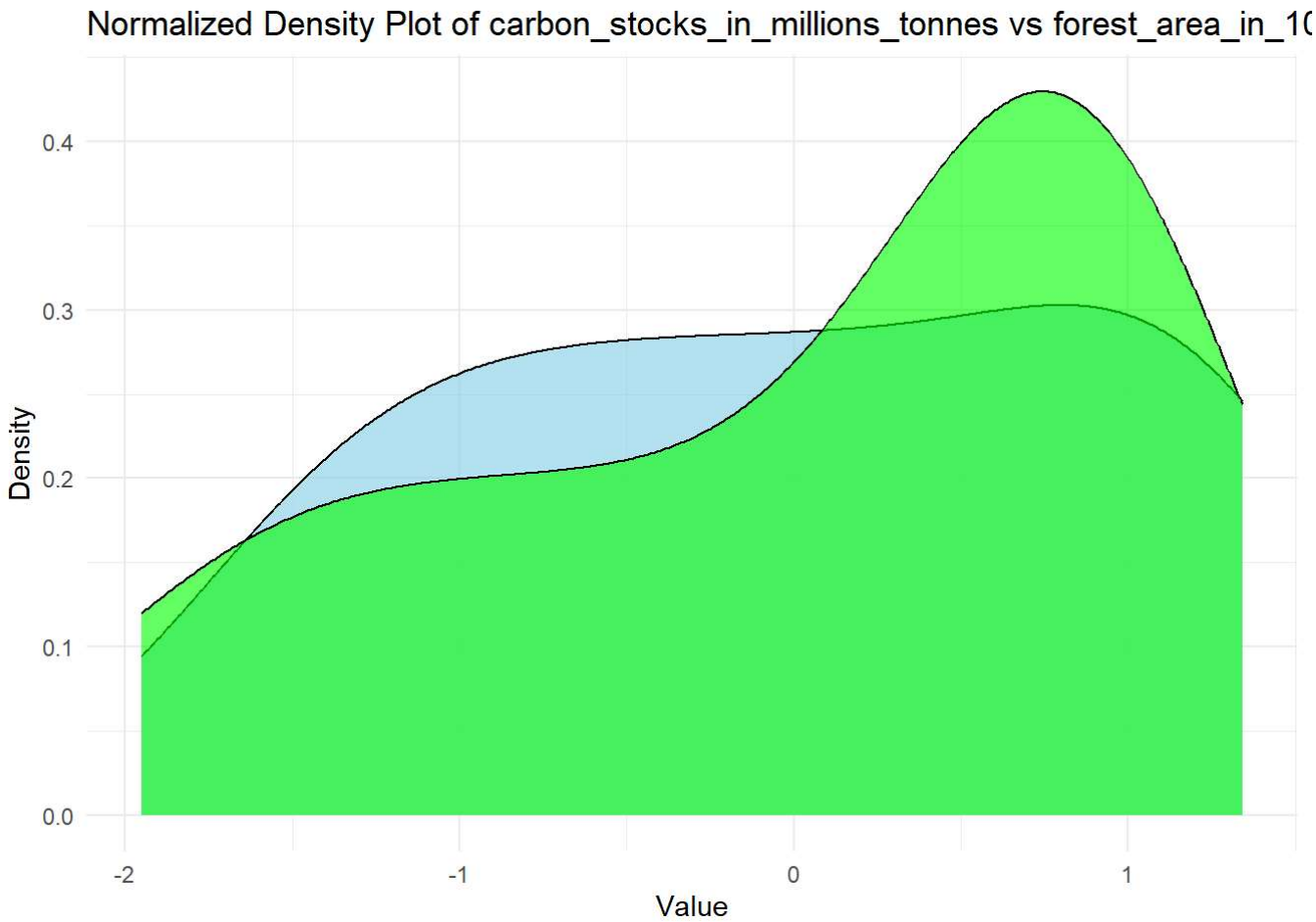

Original Density Plot of co2_emissions_in_tonnes vs carbon_stocks_in_millions_ton
Density
0
2
4
6
12
16
20
Value

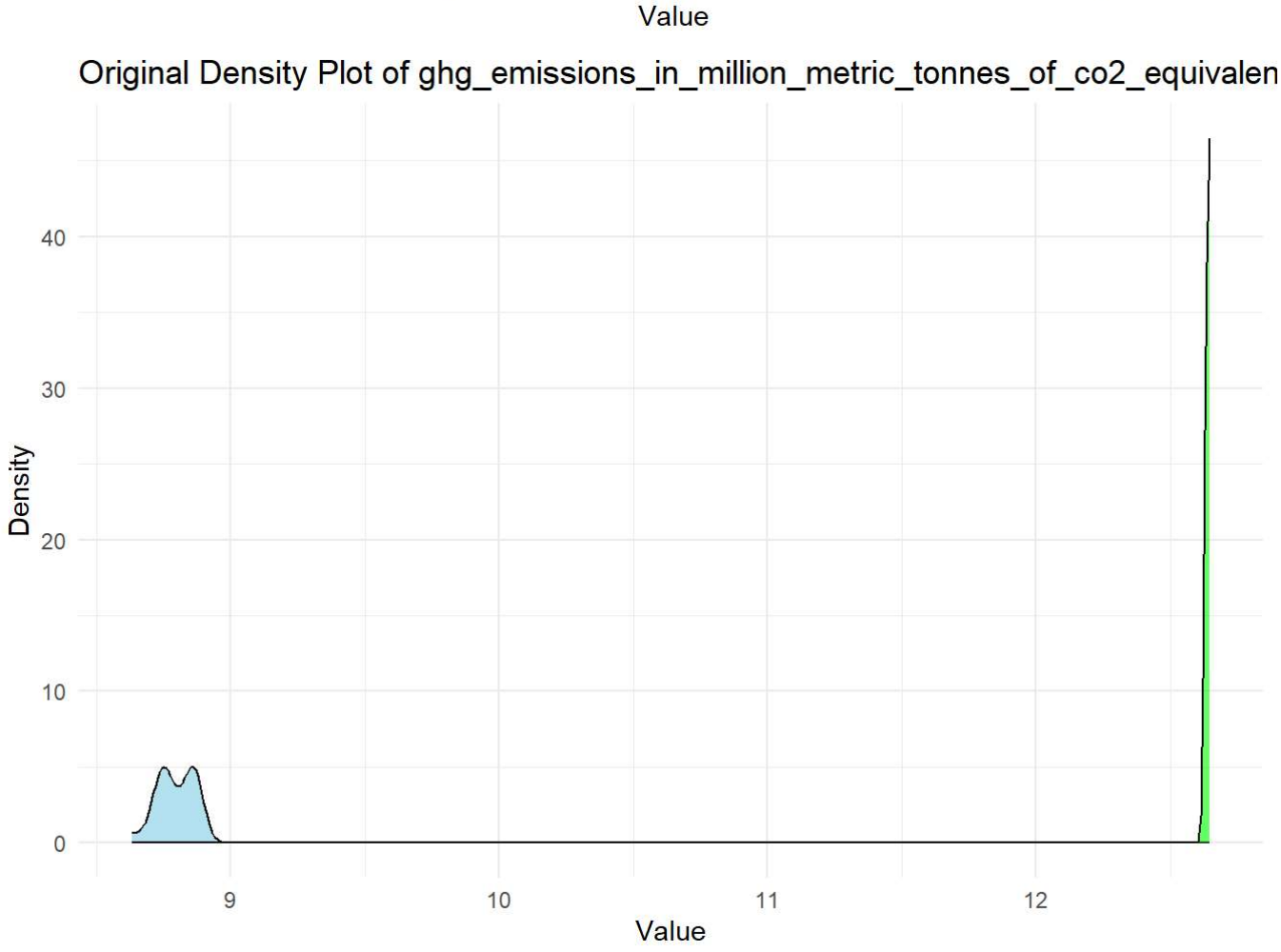

Normalized Density Plot of co2_emissions_in_tonnes vs carbon_stocks_in_millions
Density
0.0
0.1
0.2
0.3
-2
-1
0
1
Value

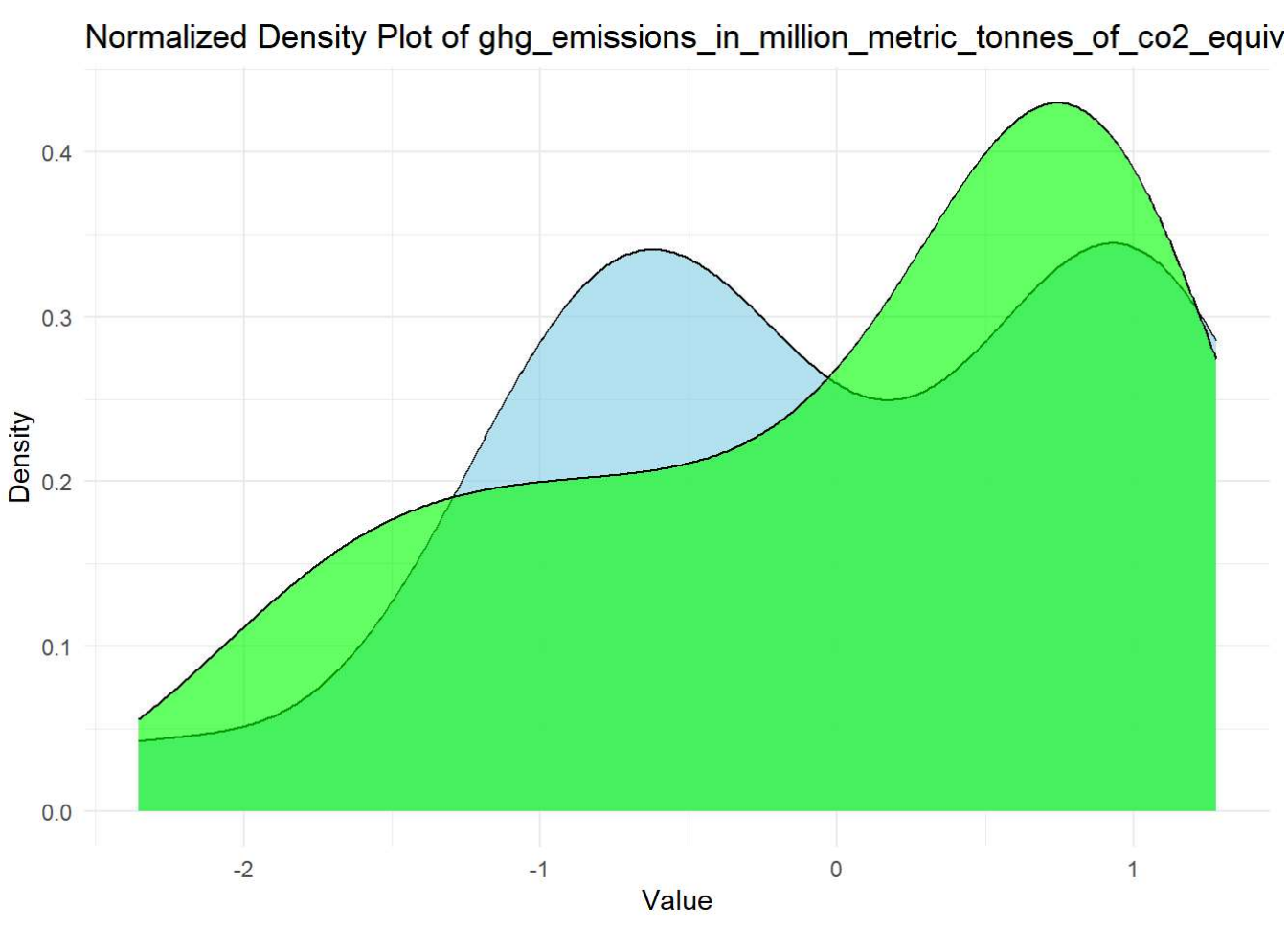

Original Density Plot of co2_emissions_in_tonnes vs ghg_emissions_in_million_met
Density
0
1
2
3
4
8
12
16
20
Value

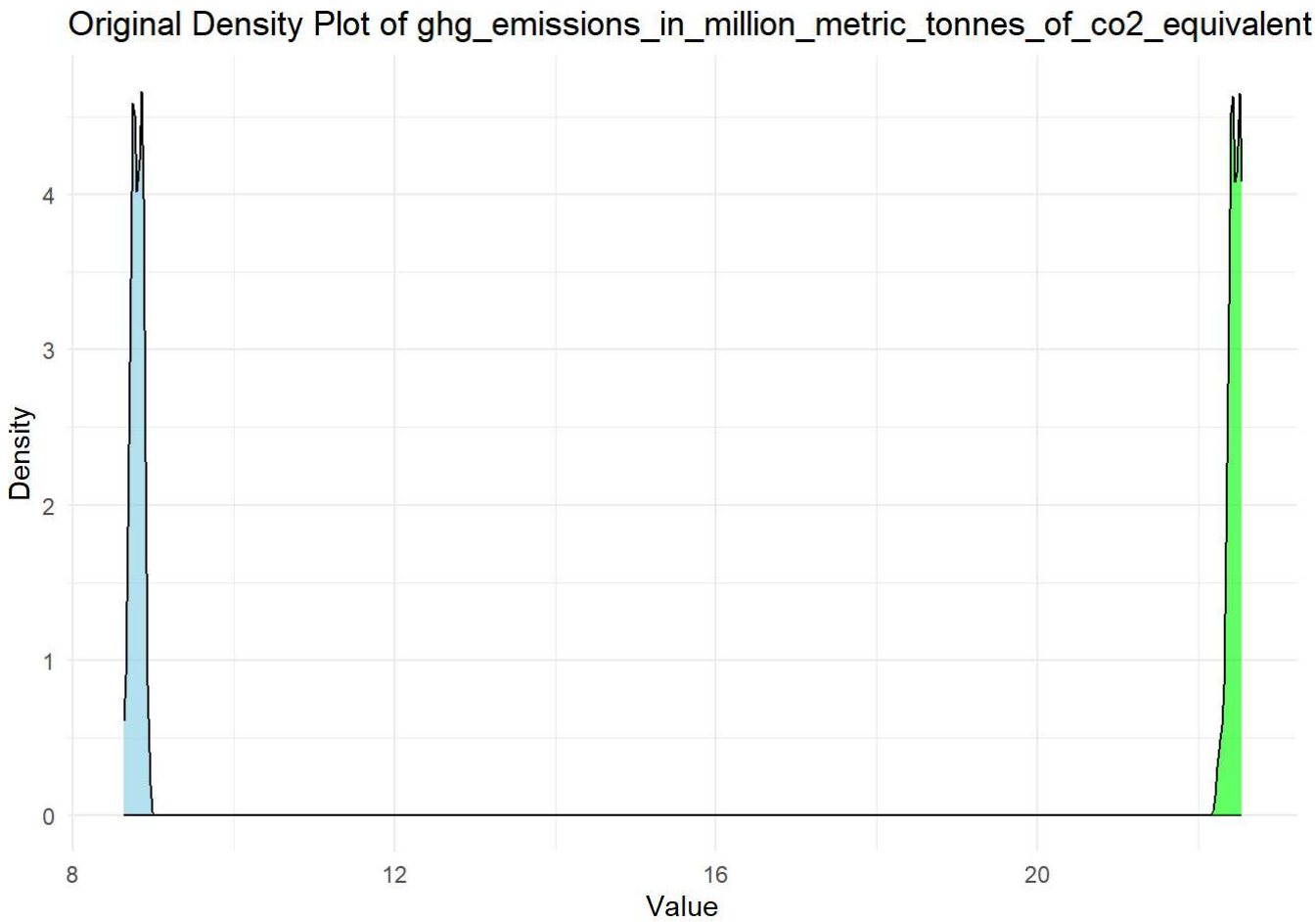

Normalized Density Plot of co2_emissions_in_tonnes vs ghg_emissions_in_million_
Density
0.0
0.1
0.2
0.3
-2
-1
0
1
Value

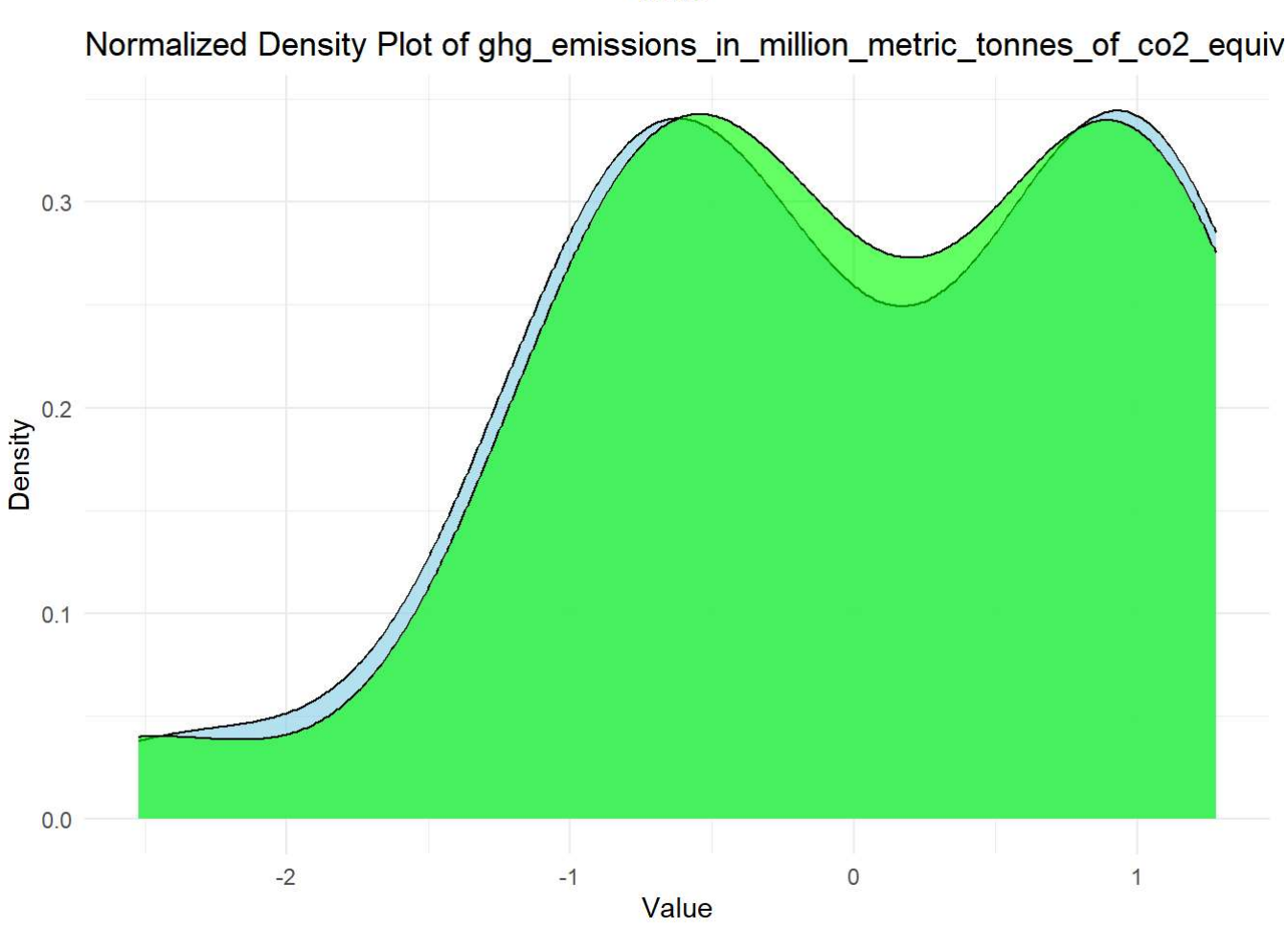

Original Density Plot of carbon_stocks_in_millions_tonnes vs forest_area_in_1000_
Density
0
10
20
30
40
10
11
12
Value

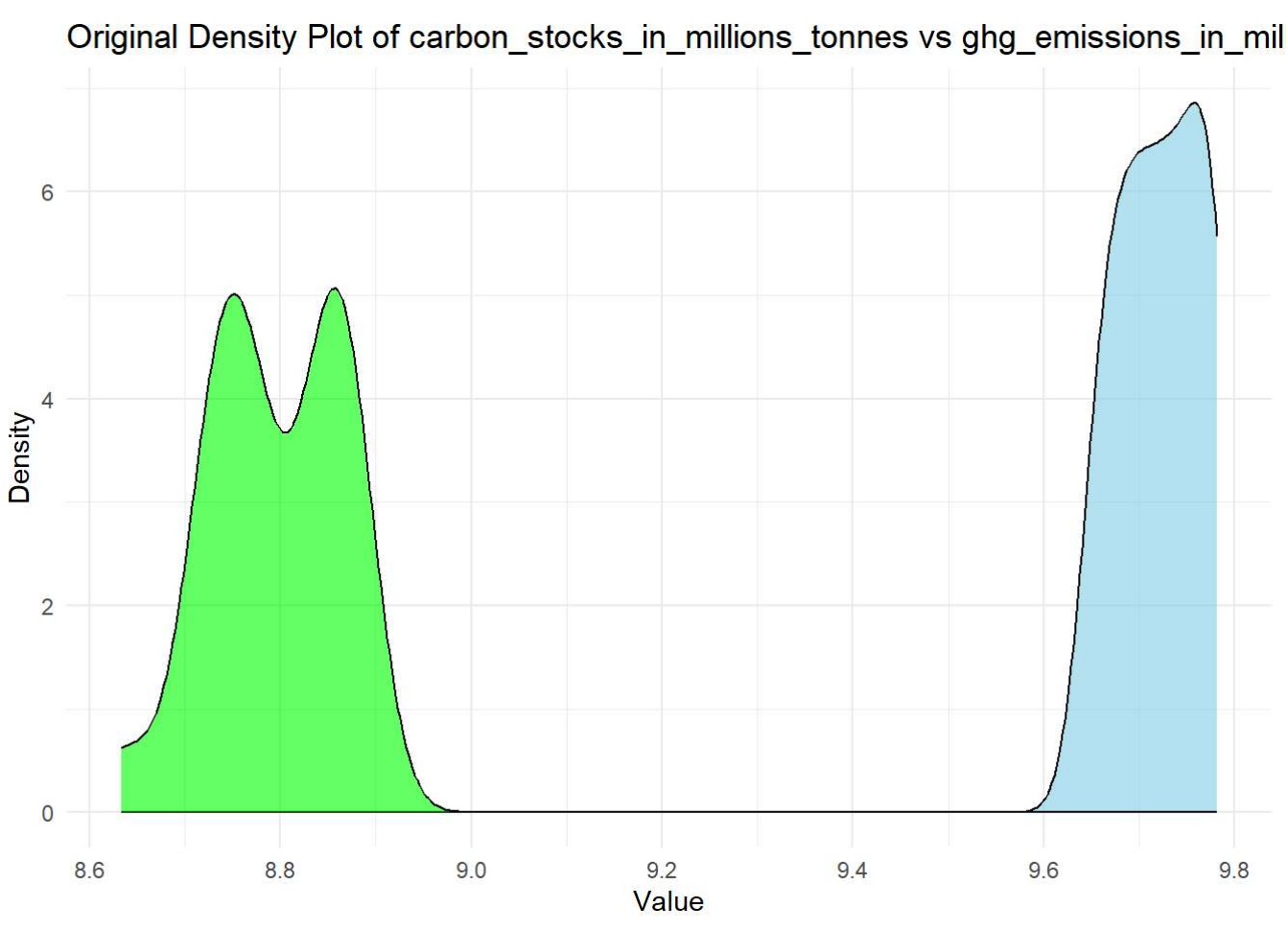

Normalized Density Plot of carbon_stocks_in_millions_tonnes vs forest_area_in_1(
Density
0.0
0.1
0.2
0.3
0.4
-2
-1
0
1
Value

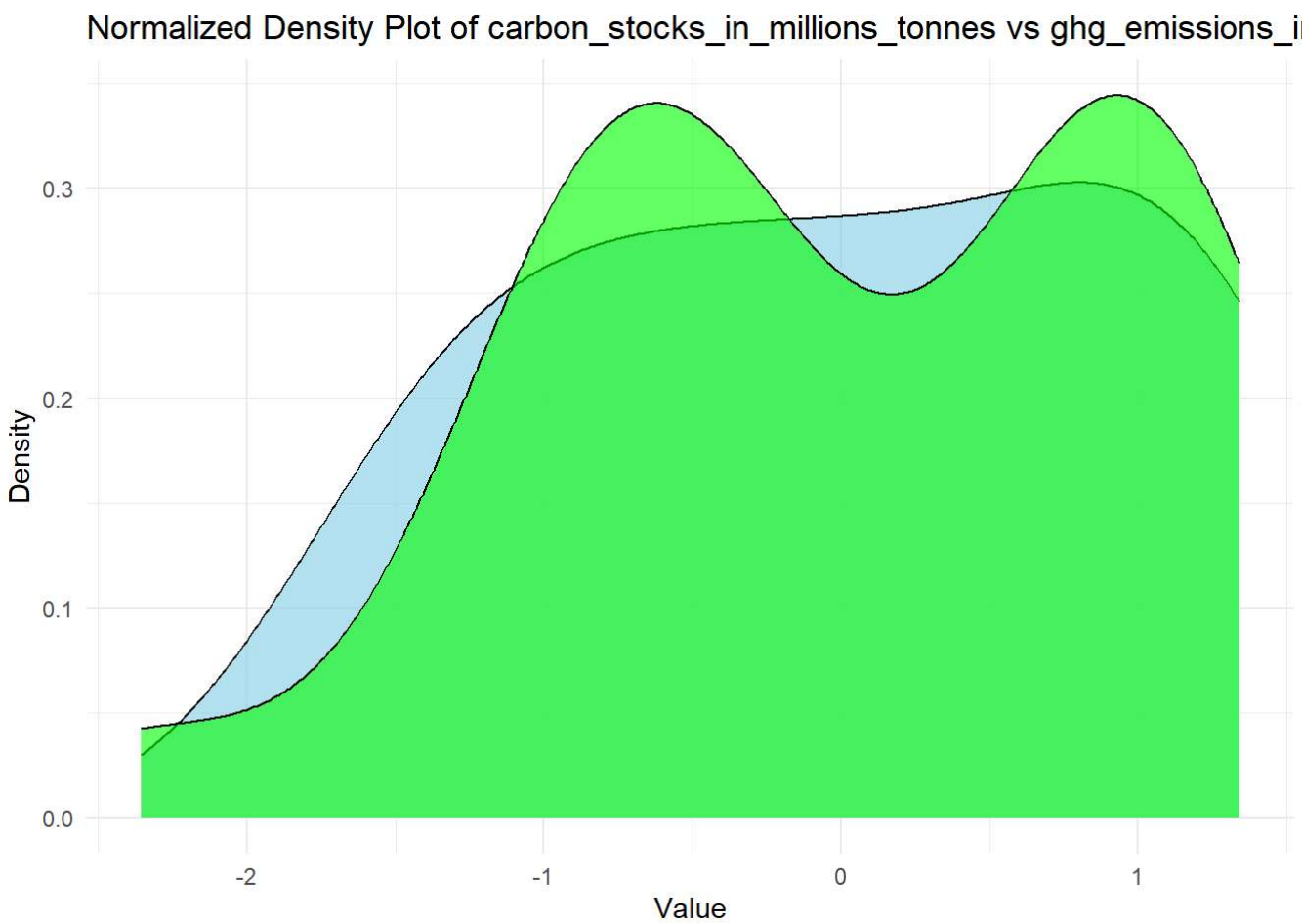

Original Density Plot of ghg_emissions_in_million_metric_tonnes_of_co2_equivalen
Density
0
10
20
30
40
9
10
11
12
Value

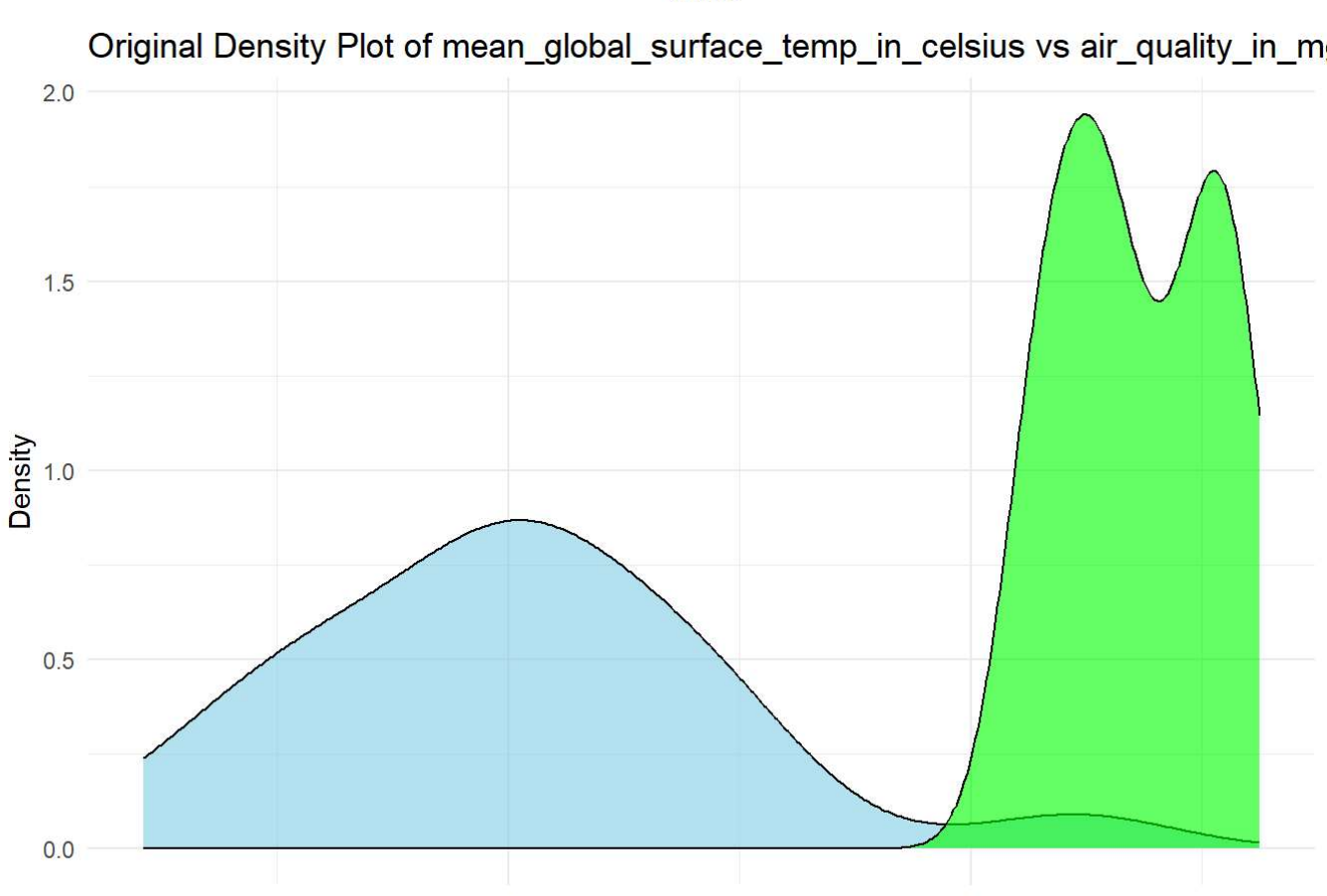

Normalized Density Plot of ghg_emissions_in_million_metric_tonnes_of_co2_equiv
Density
0.0
0.1
0.2
0.3
0.4
-2
-1
0
1
Value

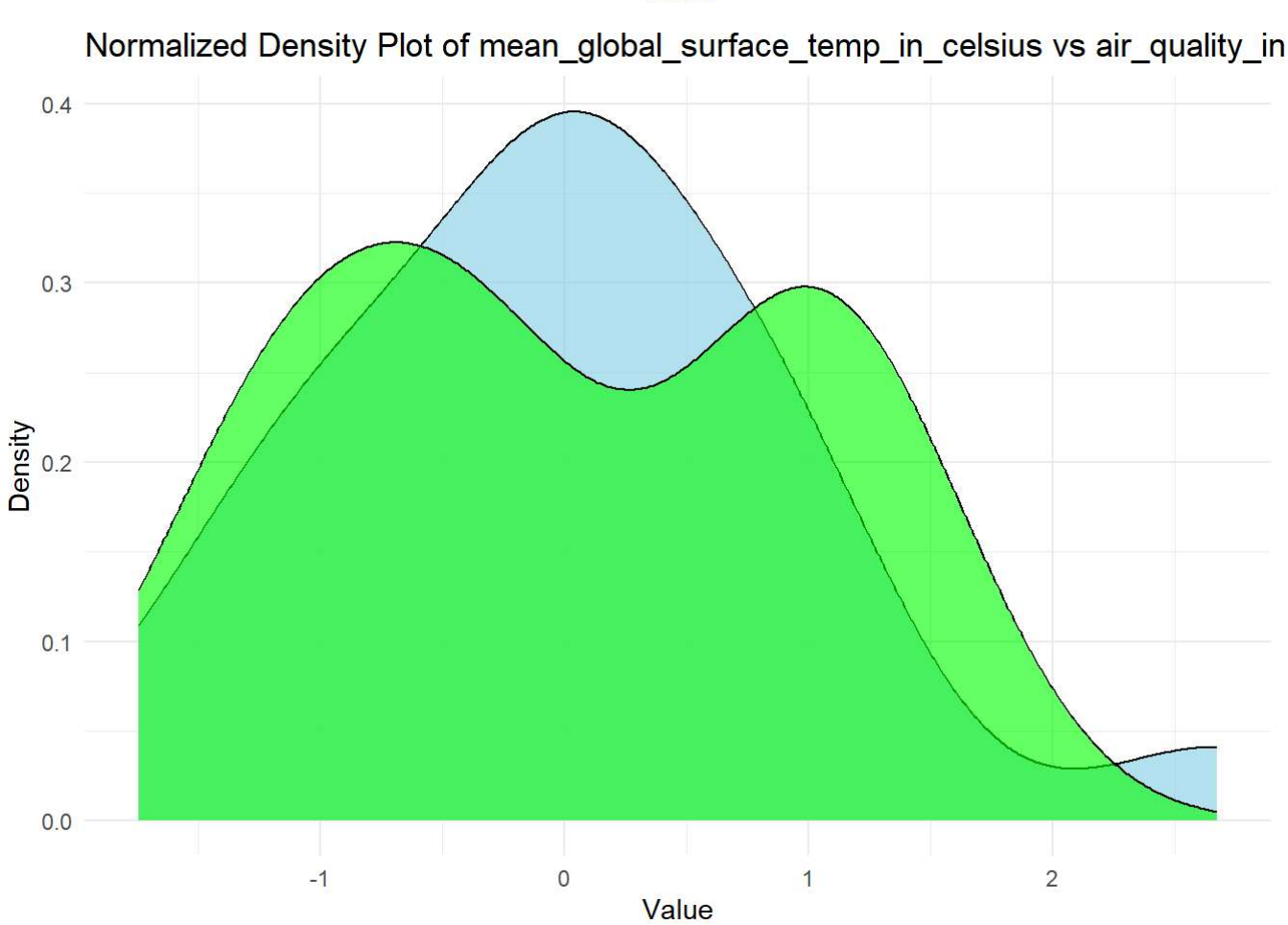

Original Density Plot of ghg_emissions_in_million_metric_tonnes_of_co2_equivalent
Density
0
1
2
3
4
8
12
16
20
Value


Normalized Density Plot of ghg_emissions_in_million_metric_tonnes_of_co2_equiv
Density
0.0
0.1
0.2
0.3
-2
-1
0
1
Value


Original Density Plot of carbon_stocks_in_millions_tonnes vs ghg_emissions_in_mil
Density
0
2
4
6
8.6
8.8
9.0
9.2
9.4
9.6
9.8
Value







Using Z score normalised data

```r
z_score_normalization <- function(x) {
    return((x - mean(x, na.rm = TRUE)) / sd(x, na.rm = TRUE))
}

relevant_cols <- c("land_area_in_1000_ha", "forest_area_in_1000_ha",
                   "carbon_stocks_in_millions_tonnes", "air_quality_in_mg_per_m3",
                   "ghg_emissions_in_million_metric_tonnes_of_co2_equivalents_per_yr",
                   "co2_emissions_in_tonnes")
# Apply z-score normalization to relevant columns
z_score_normalized_data <- log_normalized_data
z_score_normalized_data[relevant_cols] <- lapply(log_normalized_data[relevant_cols], z_score_normalization)

# Add additional columns to the normalized dataset
additional_cols <- c("iso2", "iso3", "country", "year", "disaster_count_in_number_of", "mean_global_surface_temp_in_celsiu
s")
z_score_normalized_data[additional_cols] <- log_normalized_data[additional_cols]

# Check the first few rows of the z-score normalized dataset
head(z_score_normalized_data)
write.csv(z_score_normalized_data,file="z_score_normalized_Data.csv")
```

Building Model

```r
library(dplyr)
library(forecast)

data <- z_score_normalized_data

iso3_country <- "KOR"
train_data <- data %>%
  filter(iso3 == iso3_country) %>%
  filter(year < 2015)
test_data <- data %>%
  filter(iso3 == iso3_country) %>%
  filter(year >= 2015)
train_data$year <- as.Date(as.character(train_data$year), format = "%Y")

ts_data <- ts(
  train_data$air_quality_in_mg_per_m3,
  frequency = 1, start = min(train_data$year)
)

linear_model <- lm(air_quality_in_mg_per_m3 ~ year, data = train_data)
arima_model <- auto.arima(ts_data)
ets_model <- ets(ts_data)

future_years <- seq(
  max(train_data$year) + 1 * 365, max(train_data$year) + 7 * 365,
  by = "years"
)

weights <- c(0.5,0.3,0.2)

forecast_linear <- predict(
  linear_model,
  newdata = data.frame(year = future_years)
)
forecast_arima <- forecast(arima_model, h = length(future_years))
forecast_ets <- forecast(ets_model, h = length(future_years))
ensemble_forecast <- (weights[1] * forecast_linear) +
  (weights[2] * forecast_arima$mean) +
  (weights[3] * forecast_ets$mean)

comparison <- data.frame(
  Year = future_years,
  Linear = forecast_linear,
  ARIMA = forecast_arima$mean,
  ETS = forecast_ets$mean,
  Ensemble = ensemble_forecast
)
print(comparison)
```

```
##         Year    Linear     ARIMA       ETS  Ensemble
## 1 2015-05-16 0.2074916 0.2984027 0.3742286 0.2681123
## 2 2016-05-16 0.2131667 0.2504948 0.3742286 0.2565775
## 3 2017-05-16 0.2188263 0.2223811 0.3742286 0.2509732
## 4 2018-05-16 0.2244859 0.2058832 0.3742286 0.2488536
## 5 2019-05-16 0.2301455 0.1962018 0.3742286 0.2487790
## 6 2020-05-16 0.2358206 0.1905204 0.3742286 0.2499122
```

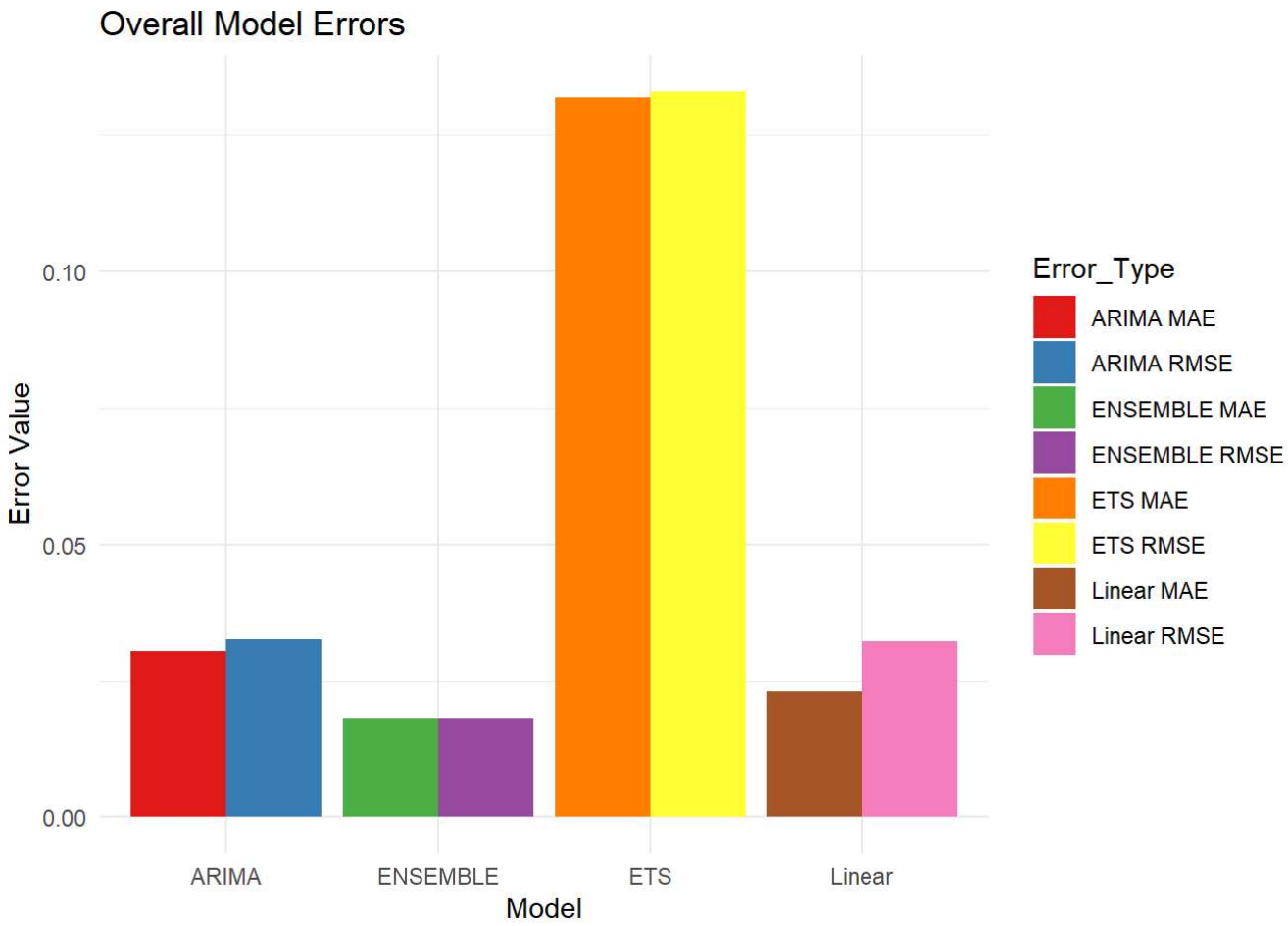
```
comparison$Linear_MAE <- abs(
  comparison$Linear - test_data$air_quality_in_mg_per_m3
)
comparison$ARIMA_MAE <- abs(
  comparison$ARIMA - test_data$air_quality_in_mg_per_m3
)
comparison$ETS_MAE <- abs(
  comparison$ETS - test_data$air_quality_in_mg_per_m3
)
comparison$ENSEMBLE_MAE <- abs(comparison$Ensemble - test_data$air_quality_in_mg_per_m3)

comparison$Linear_RMSE <- sqrt(mean(
  (comparison$Linear - test_data$air_quality_in_mg_per_m3)^2
))

comparison$ARIMA_RMSE <- sqrt(mean(
  (comparison$ARIMA - test_data$air_quality_in_mg_per_m3)^2
))
comparison$ETS_RMSE <- sqrt(mean(
  (comparison$ETS - test_data$air_quality_in_mg_per_m3)^2
))

comparison$ENSEMBLE_RMSE <- sqrt(mean(
  (comparison$Ensemble - test_data$air_quality_in_mg_per_m3)^2
))


errors <- comparison %>%
  group_by(Year) %>%
  summarize(
    Linear_MAE = mean(Linear_MAE),
    ARIMA_MAE = mean(ARIMA_MAE),
    ETS_MAE = mean(ETS_MAE),
    ENSEMBLE_MAE = mean(ENSEMBLE_MAE),

    Linear_RMSE = mean(Linear_RMSE),
    ARIMA_RMSE = mean(ARIMA_RMSE),
    ETS_RMSE = mean(ETS_RMSE),
    ENSEMBLE_RMSE = mean(ENSEMBLE_RMSE)
  )
print(errors)
```

```
## # A tibble: 6 × 9
##   Year       Linear_MAE ARIMA_MAE ETS_MAE ENSEMBLE_MAE Linear_RMSE ARIMA_RMSE ETS_RMSE ENSEMBLE_RMSE
##   <date>          <dbl>     <dbl>   <dbl>        <dbl>       <dbl>      <dbl>    <dbl>         <dbl>
## 1 2015-05-16    0.0444    0.0466  0.122        0.0163      0.0322     0.0326    0.133        0.0181
## 2 2016-05-16    0.0627    0.0254  0.0984       0.0193      0.0322     0.0326    0.133        0.0181
## 3 2017-05-16    0.0151    0.0116  0.140        0.0170      0.0322     0.0326    0.133        0.0181
## 4 2018-05-16    0.00559   0.0242  0.144        0.0188      0.0322     0.0326    0.133        0.0181
## 5 2019-05-16    0.00379   0.0377  0.140        0.0148      0.0322     0.0326    0.133        0.0181
## 6 2020-05-16    0.00768   0.0376  0.146        0.0218      0.0322     0.0326    0.133        0.0181
```

```
# Calculate overall model errors
overall_errors <- errors %>%
  summarise(
    Linear_MAE = mean(Linear_MAE),
    ARIMA_MAE = mean(ARIMA_MAE),
    ETS_MAE = mean(ETS_MAE),
    ENSEMBLE_MAE = mean(ENSEMBLE_MAE),
    Linear_RMSE = mean(Linear_RMSE),
    ARIMA_RMSE = mean(ARIMA_RMSE),
    ETS_RMSE = mean(ETS_RMSE),
    ENSEMBLE_RMSE = mean(ENSEMBLE_RMSE)
  )

# Print overall model errors
print(overall_errors)
```

```
## # A tibble: 1 × 8
##   Linear_MAE ARIMA_MAE ETS_MAE ENSEMBLE_MAE Linear_RMSE ARIMA_RMSE ETS_RMSE ENSEMBLE_RMSE
##        <dbl>     <dbl>   <dbl>        <dbl>       <dbl>      <dbl>    <dbl>         <dbl>
## 1     0.0232    0.0305   0.132       0.0180      0.0322     0.0326    0.133        0.0181
```

```r
library(ggplot2)

# Convert overall_errors to a format suitable for plotting
overall_errors_plot <- overall_errors %>%
  pivot_longer(cols = contains("_MAE") | contains("_RMSE"),
               names_to = "Error_Type",
               values_to = "Error_Value") %>%
  mutate(Model = gsub("_MAE|_RMSE", "", Error_Type),
         Error_Type = gsub("_MAE", " MAE", Error_Type),
         Error_Type = gsub("_RMSE", " RMSE", Error_Type))

# Draw bar plot
ggplot(overall_errors_plot, aes(x = Model, y = Error_Value, fill = Error_Type)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Overall Model Errors",
       x = "Model",
       y = "Error Value") +
  scale_fill_brewer(palette = "Set1") +
  theme_minimal()
```



Create a scatter plot for predicted vs actual values

```r
library(dplyr)
library(forecast)

data <- z_score_normalized_data

iso3_country <- "KOR"
train_data <- data %>%
  filter(iso3 == iso3_country) %>%
  filter(year < 2015)
test_data <- data %>%
  filter(iso3 == iso3_country) %>%
  filter(year >= 2015)
train_data$year <- as.Date(as.character(train_data$year), format = "%Y")

ts_data <- ts(
  train_data$air_quality_in_mg_per_m3,
  frequency = 1, start = min(train_data$year)
)

linear_model <- lm(air_quality_in_mg_per_m3 ~ year, data = train_data)
arima_model <- auto.arima(ts_data)
ets_model <- ets(ts_data)

future_years <- seq(
  max(train_data$year) + 1 * 365, max(train_data$year) + 7 * 365,
  by = "years"
)

weights <- c(0.5,0.3,0.2)

forecast_linear <- predict(
  linear_model,
  newdata = data.frame(year = future_years)
)
forecast_arima <- forecast(arima_model, h = length(future_years))
forecast_ets <- forecast(ets_model, h = length(future_years))
ensemble_forecast <- (weights[1] * forecast_linear) +
  (weights[2] * forecast_arima$mean) +
  (weights[3] * forecast_ets$mean)

comparison <- data.frame(
  Year = future_years,
  Linear = forecast_linear,
  ARIMA = forecast_arima$mean,
  ETS = forecast_ets$mean,
  Ensemble = ensemble_forecast
)
print(comparison)
```

```
##         Year    Linear     ARIMA       ETS  Ensemble
## 1 2015-05-16 0.2074916 0.2984027 0.3742286 0.2681123
## 2 2016-05-16 0.2131667 0.2504948 0.3742286 0.2565775
## 3 2017-05-16 0.2188263 0.2223811 0.3742286 0.2509732
## 4 2018-05-16 0.2244859 0.2058832 0.3742286 0.2488536
## 5 2019-05-16 0.2301455 0.1962018 0.3742286 0.2487790
## 6 2020-05-16 0.2358206 0.1905204 0.3742286 0.2499122
```

```
comparison$Linear_MAE <- abs(
  comparison$Linear - test_data$air_quality_in_mg_per_m3
)
comparison$ARIMA_MAE <- abs(
  comparison$ARIMA - test_data$air_quality_in_mg_per_m3
)
comparison$ETS_MAE <- abs(
  comparison$ETS - test_data$air_quality_in_mg_per_m3
)
comparison$ENSEMBLE_MAE <- abs(comparison$Ensemble - test_data$air_quality_in_mg_per_m3)

comparison$Linear_RMSE <- sqrt(mean(
  (comparison$Linear - test_data$air_quality_in_mg_per_m3)^2
))

comparison$ARIMA_RMSE <- sqrt(mean(
  (comparison$ARIMA - test_data$air_quality_in_mg_per_m3)^2
))
comparison$ETS_RMSE <- sqrt(mean(
  (comparison$ETS - test_data$air_quality_in_mg_per_m3)^2
))

comparison$ENSEMBLE_RMSE <- sqrt(mean(
  (comparison$Ensemble - test_data$air_quality_in_mg_per_m3)^2
))

errors <- comparison %>%
  group_by(Year) %>%
  summarize(
    Linear_MAE = mean(Linear_MAE),
    ARIMA_MAE = mean(ARIMA_MAE),
    ETS_MAE = mean(ETS_MAE),
    ENSEMBLE_MAE = mean(ENSEMBLE_MAE),

    Linear_RMSE = mean(Linear_RMSE),
    ARIMA_RMSE = mean(ARIMA_RMSE),
    ETS_RMSE = mean(ETS_RMSE),
    ENSEMBLE_RMSE = mean(ENSEMBLE_RMSE)
  )
print(errors)
```

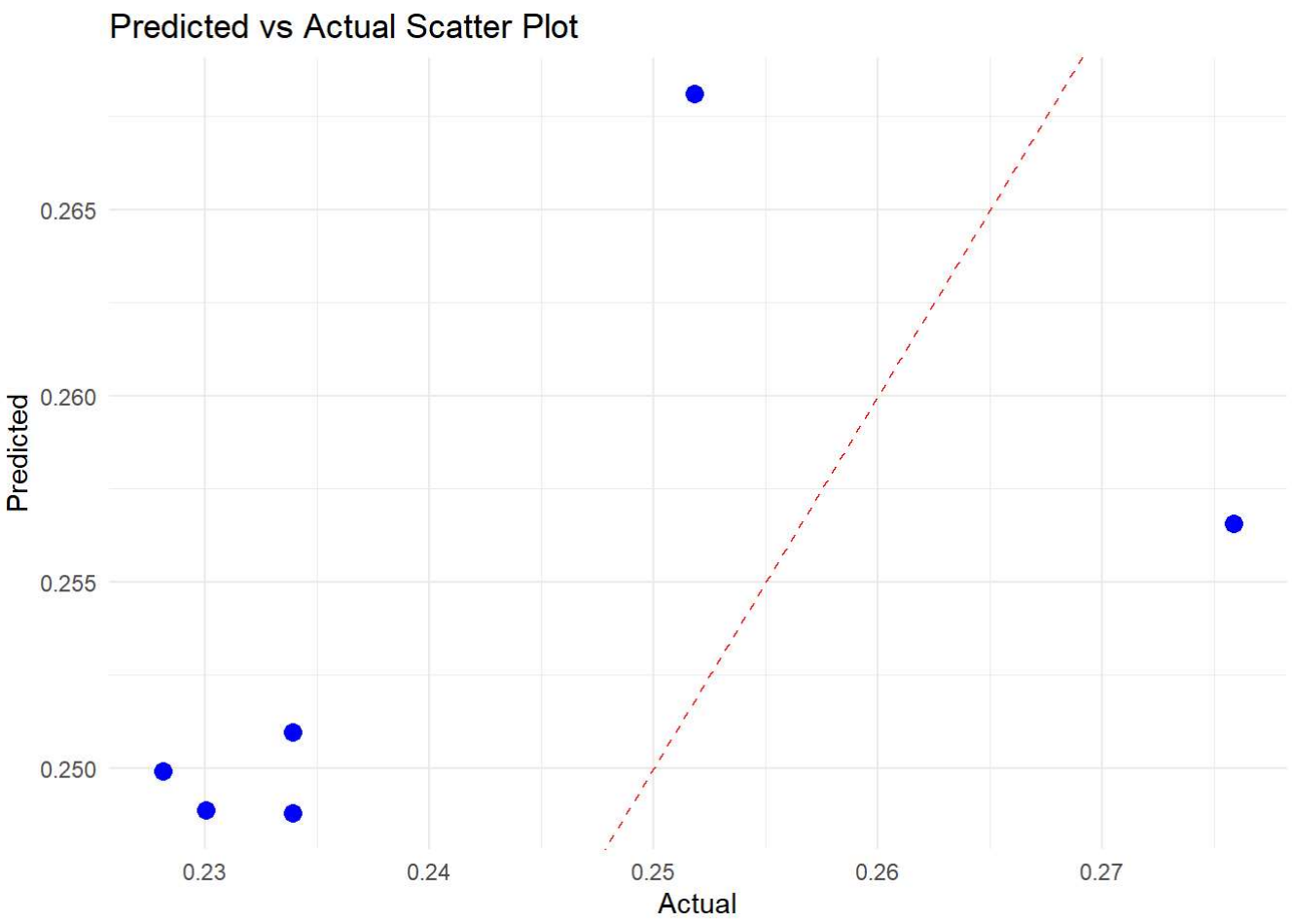
```
## # A tibble: 6 × 9
##   Year       Linear_MAE ARIMA_MAE ETS_MAE ENSEMBLE_MAE Linear_RMSE ARIMA_RMSE ETS_RMSE ENSEMBLE_RMSE
##   <date>          <dbl>     <dbl>   <dbl>        <dbl>       <dbl>      <dbl>    <dbl>         <dbl>
## 1 2015-05-16    0.0444     0.0466  0.122        0.0163      0.0322     0.0326    0.133        0.0181
## 2 2016-05-16    0.0627     0.0254  0.0984       0.0193      0.0322     0.0326    0.133        0.0181
## 3 2017-05-16    0.0151     0.0116  0.140        0.0170      0.0322     0.0326    0.133        0.0181
## 4 2018-05-16    0.00559    0.0242  0.144        0.0188      0.0322     0.0326    0.133        0.0181
## 5 2019-05-16    0.00379    0.0377  0.140        0.0148      0.0322     0.0326    0.133        0.0181
## 6 2020-05-16    0.00768    0.0376  0.146        0.0218      0.0322     0.0326    0.133        0.0181
```

```
# Calculate overall model errors
overall_errors <- errors %>%
  summarise(
    Linear_MAE = mean(Linear_MAE),
    ARIMA_MAE = mean(ARIMA_MAE),
    ETS_MAE = mean(ETS_MAE),
    ENSEMBLE_MAE = mean(ENSEMBLE_MAE),
    Linear_RMSE = mean(Linear_RMSE),
    ARIMA_RMSE = mean(ARIMA_RMSE),
    ETS_RMSE = mean(ETS_RMSE),
    ENSEMBLE_RMSE = mean(ENSEMBLE_RMSE)
  )

# Print overall model errors
print(overall_errors)
```

```
## # A tibble: 1 × 8
##   Linear_MAE ARIMA_MAE ETS_MAE ENSEMBLE_MAE Linear_RMSE ARIMA_RMSE ETS_RMSE ENSEMBLE_RMSE
##        <dbl>     <dbl>   <dbl>        <dbl>       <dbl>      <dbl>    <dbl>         <dbl>
## 1     0.0232    0.0305   0.132       0.0180      0.0322     0.0326    0.133        0.0181
```

```r
library(ggplot2)

# Convert overall_errors to a format suitable for plotting
overall_errors_plot <- overall_errors %>%
  pivot_longer(cols = contains("_MAE") | contains("_RMSE"),
               names_to = "Error_Type",
               values_to = "Error_Value") %>%
  mutate(Model = gsub("_MAE|_RMSE", "", Error_Type),
         Error_Type = gsub("_MAE", " MAE", Error_Type),
         Error_Type = gsub("_RMSE", " RMSE", Error_Type))



library(ggplot2)

# Assuming you have actual and predicted values stored in vectors
plt_data <- data %>%
  filter(iso3 == iso3_country) %>%
  filter(year >= 2015)
actual <- plt_data$air_quality_in_mg_per_m3 # Replace these with your actual values
predicted <- ensemble_forecast  # Replace these with your predicted values

# Create a data frame with actual and predicted values
df <- data.frame(actual = actual, predicted = predicted)

# Create a scatter plot for predicted vs actual values
ggplot(df, aes(x = actual, y = predicted)) +
  geom_point(color = "blue", size = 3) +
  geom_abline(intercept = 0, slope = 1, linetype = "dashed", color = "red") +  # Add a line of equality
  labs(x = "Actual", y = "Predicted", title = "Predicted vs Actual Scatter Plot") +
  theme_minimal()
```

```
## Don't know how to automatically pick scale for object of type <ts>. Defaulting to continuous.
```

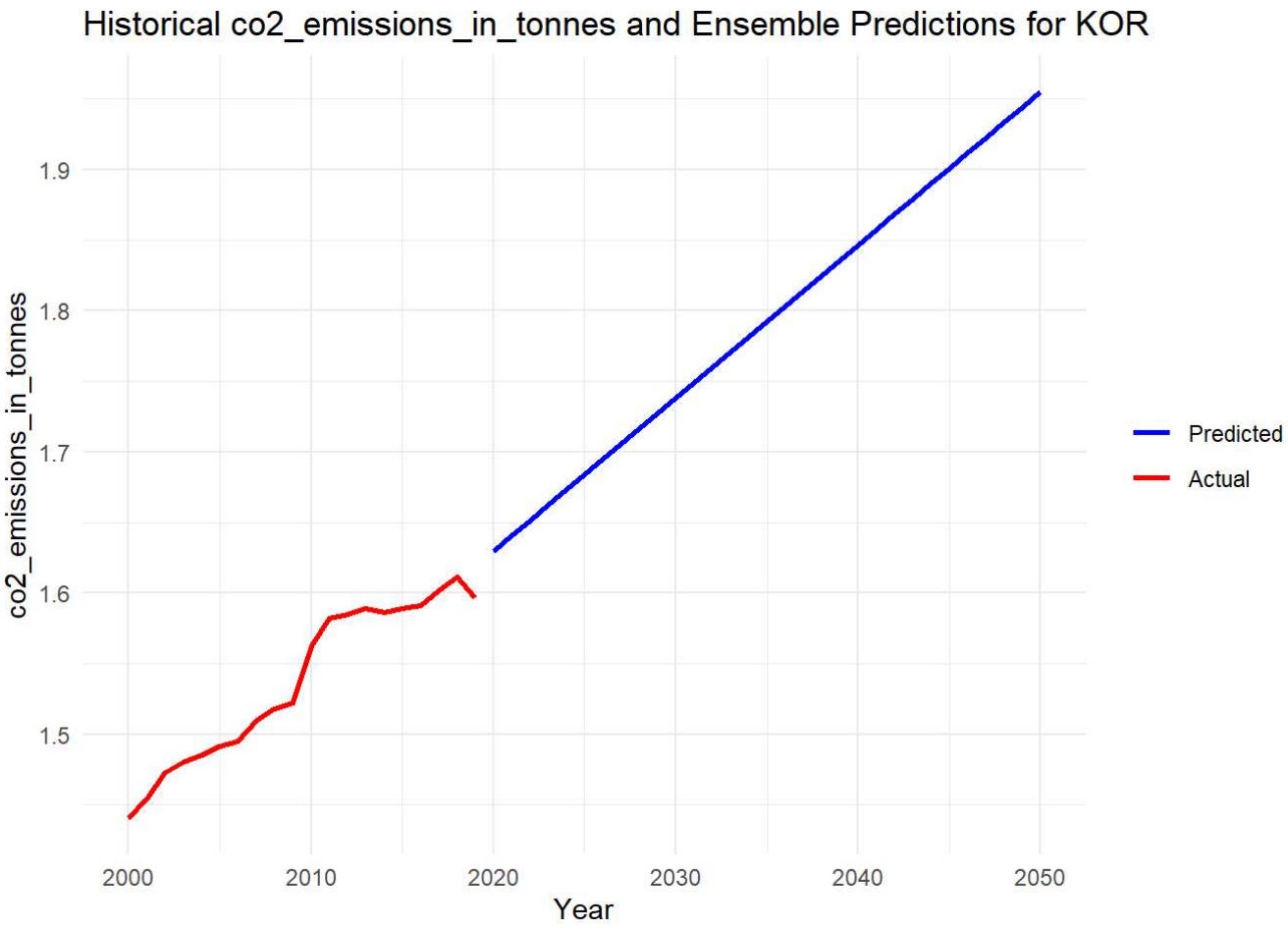


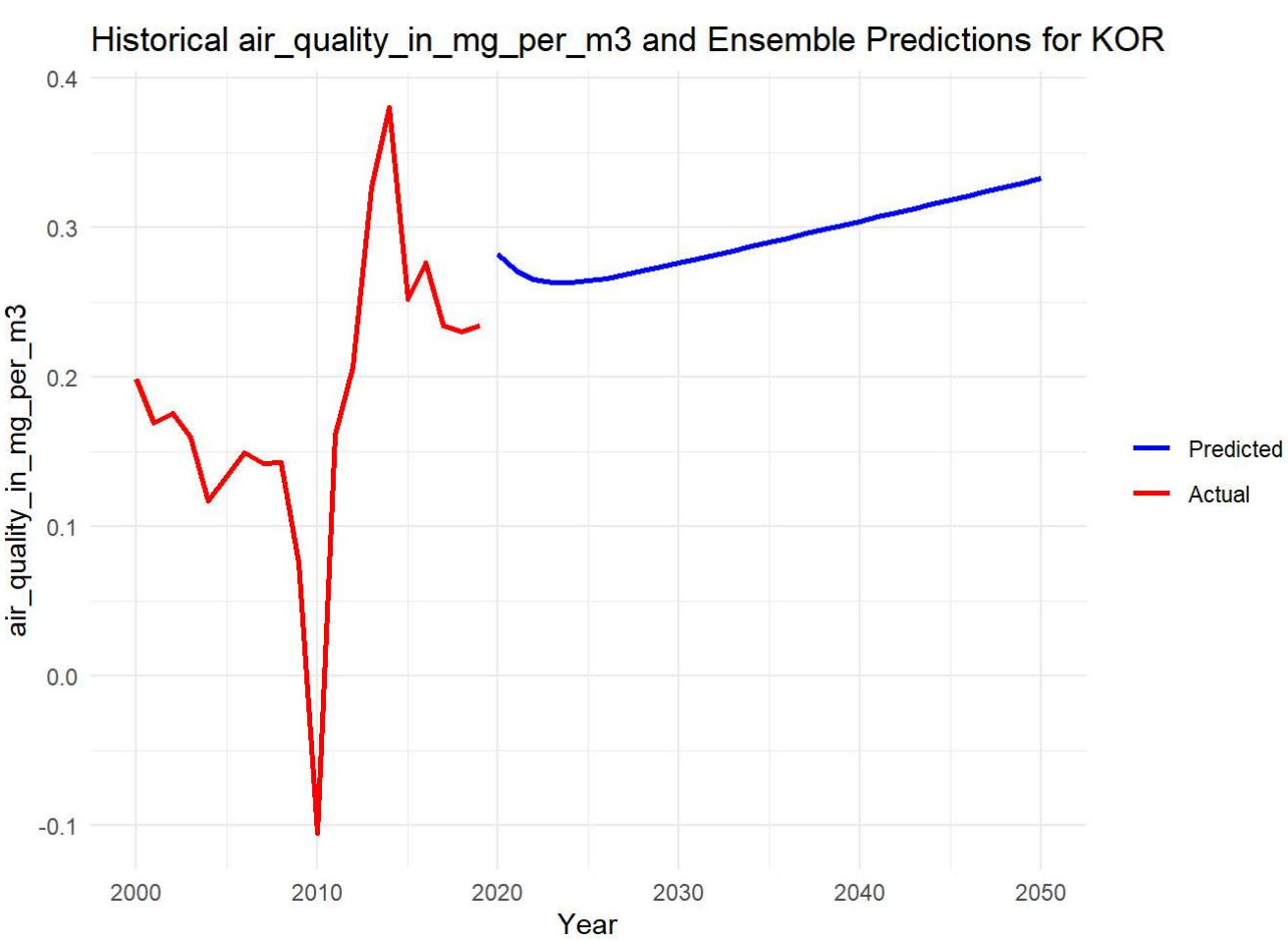
Results of Time Series Analysis

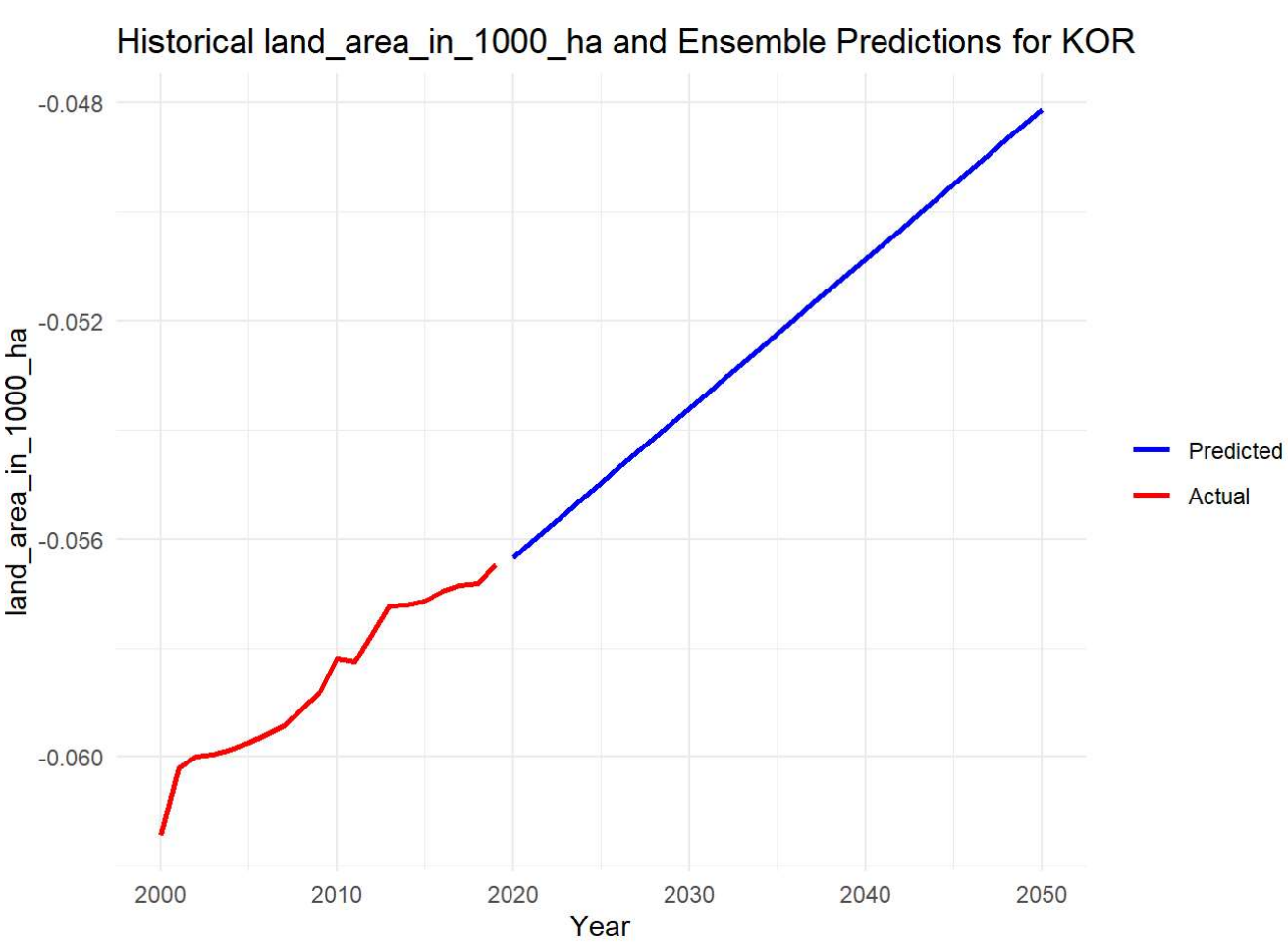
Plotting the Historical data along with the predicted data in a single plot.

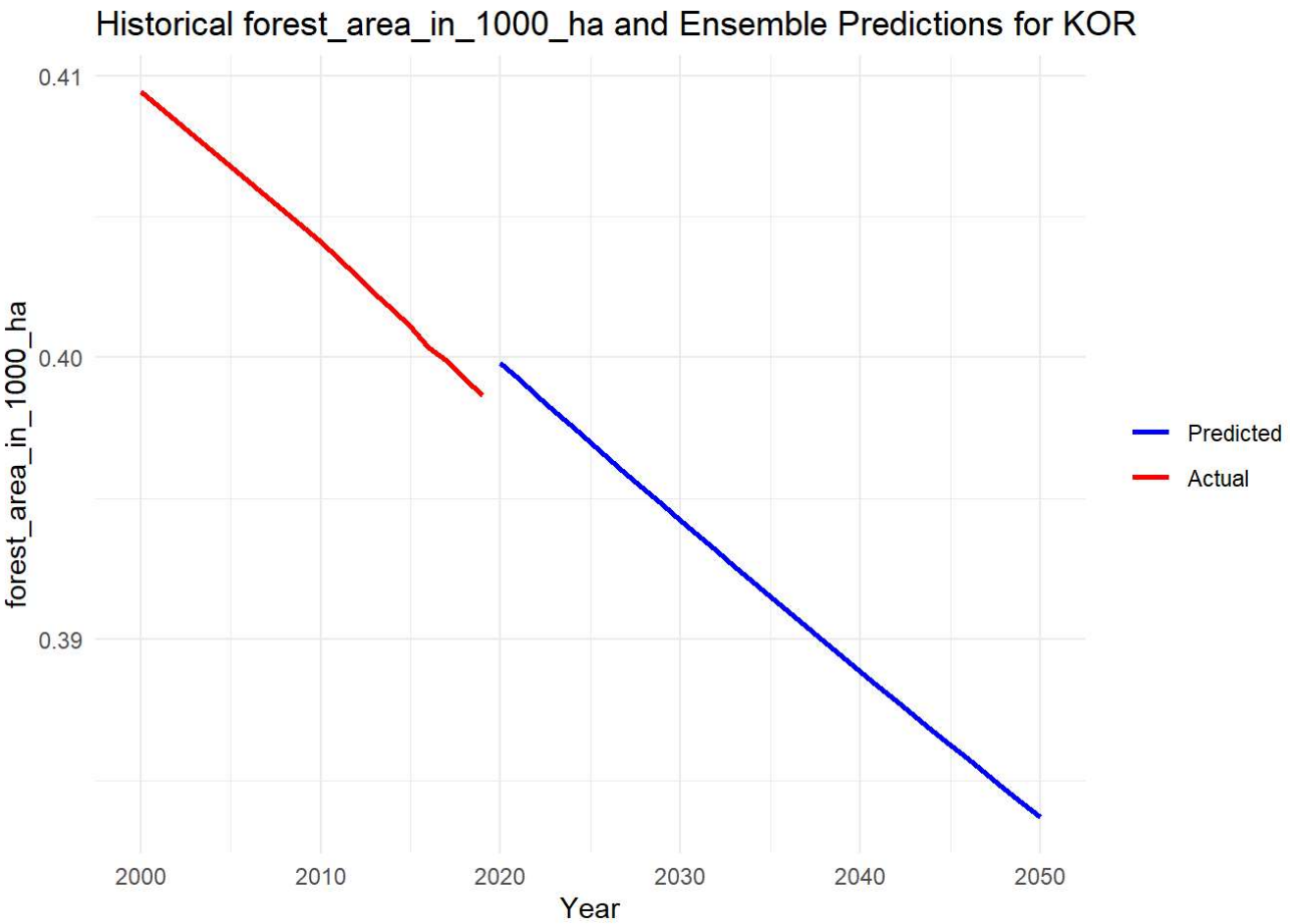
```r
library(dplyr)
library(forecast)
library(ggplot2)

# Load your data
data <- z_score_normalized_data

# Specify the country code
iso3_country <- "KOR"

# Filter the data for training and testing
train_data <- data %>%
  filter(iso3 == iso3_country) %>%
  filter(year < 2015)
test_data <- data %>%
  filter(iso3 == iso3_country) %>%
  filter(year >= 2015)

# Convert year to numeric
train_data$year <- as.numeric(train_data$year)

# Create a list to store ensemble predictions for each variable
ensemble_predictions_list <- list()

# Define future years from 2020 to 2050
future_years <- 2020:2050

# Function to make predictions for a given variable
make_predictions <- function(variable) {
  # Create time series data
  ts_data <- ts(train_data[[variable]], frequency = 1, start = min(train_data$year))

  # Fit models
  linear_model <- lm(train_data[[variable]] ~ year, data = train_data)
  arima_model <- auto.arima(ts_data)
  ets_model <- ets(ts_data)

  # Make predictions using the ensemble model
  forecast_linear <- predict(linear_model, newdata = data.frame(year = future_years))
  forecast_arima <- forecast(arima_model, h = length(future_years))
  forecast_ets <- forecast(ets_model, h = length(future_years))
  ensemble_forecast <- (weights[1] * forecast_linear) +
    (weights[2] * forecast_arima$mean) +
    (weights[3] * forecast_ets$mean)

  # Return ensemble predictions
  return(ensemble_forecast)
}

# Predict and store ensemble predictions for each variable
variables <- c("co2_emissions_in_tonnes", "air_quality_in_mg_per_m3", "land_area_in_1000_ha",
               "forest_area_in_1000_ha", "carbon_stocks_in_millions_tonnes",
               "ghg_emissions_in_million_metric_tonnes_of_co2_equivalents_per_yr")

for (variable in variables) {
  ensemble_predictions_list[[variable]] <- make_predictions(variable)
}

# Create a data frame with the predicted values and the corresponding years for each variable
ensemble_predictions <- data.frame(Year = future_years)
for (variable in variables) {
  ensemble_predictions[[paste0(variable, "_predictions")]] <- ensemble_predictions_list[[variable]]
}

# Plot historic data for all variables with respective predicted values

# Filter historical data for the years 1999 to 2020
historical_data <- data %>%
  filter(iso3 == iso3_country & year >= 1999 & year < 2020)

# Function to plot historical data and ensemble predictions for a given variable
plot_variable <- function(variable) {
  # Plot historical data and ensemble predictions
  ggplot() +
    geom_line(data = historical_data, aes(x = year, y = !!sym(variable), color = "Historical Data"), size = 1) +
    geom_line(data = ensemble_predictions, aes(x = Year, y = !!sym(paste0(variable, "_predictions")), color = "Ensemble Pred
ictions"), size = 1) +
    labs(title = paste("Historical", variable, "and Ensemble Predictions for", iso3_country),
         x = "Year",
         y = variable) +
```

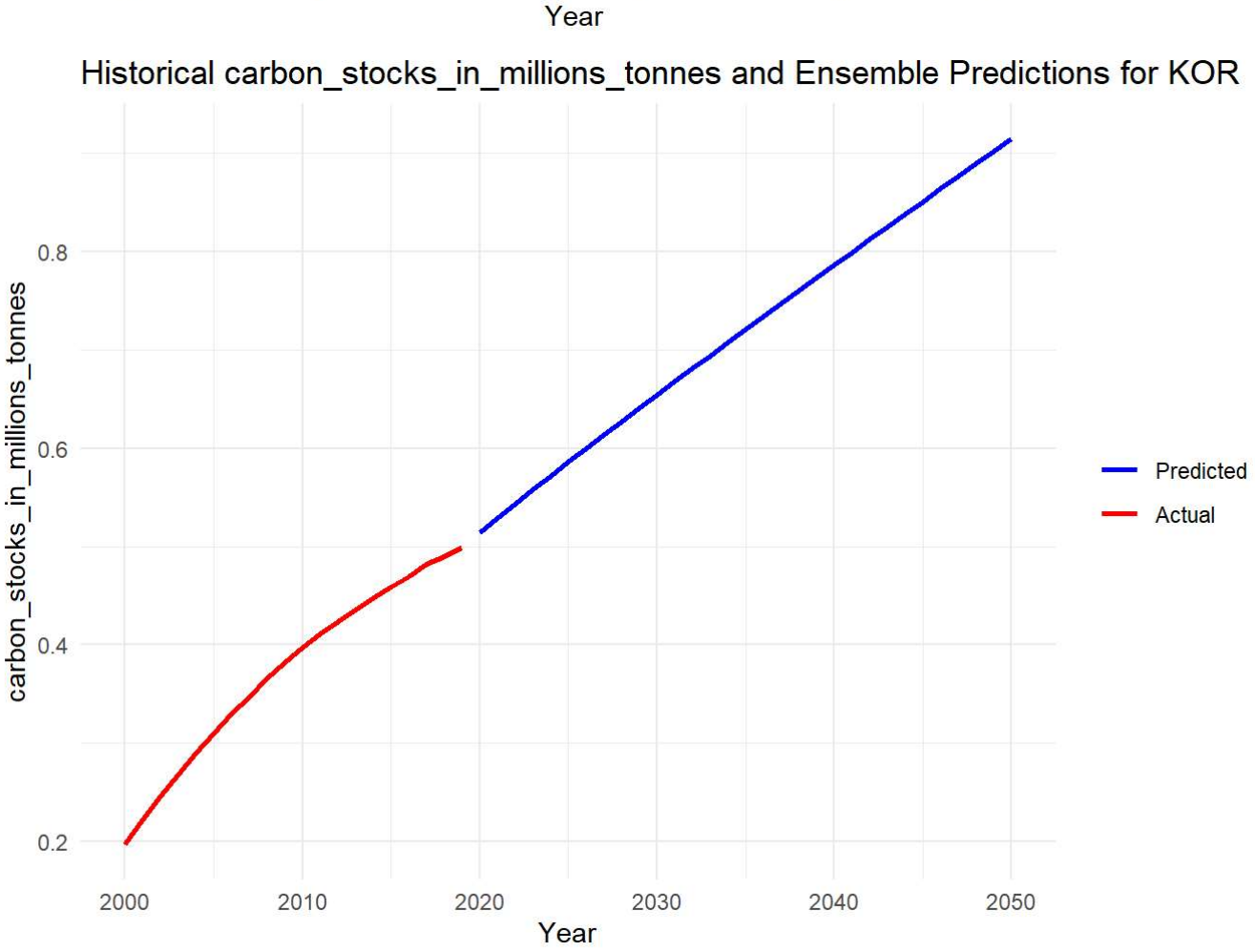
```
    scale_color_manual(values = c("blue", "red"),
                       labels = c("Predicted", "Actual")) +
    theme_minimal() +
    theme(legend.title = element_blank()) # Remove legend title
}

# Plot historic data for all variables with respective predicted values
plots <- lapply(variables, plot_variable)

# Loop through each plot and print it
for (i in seq_along(plots)) {
  print(plots[[i]])
}
```

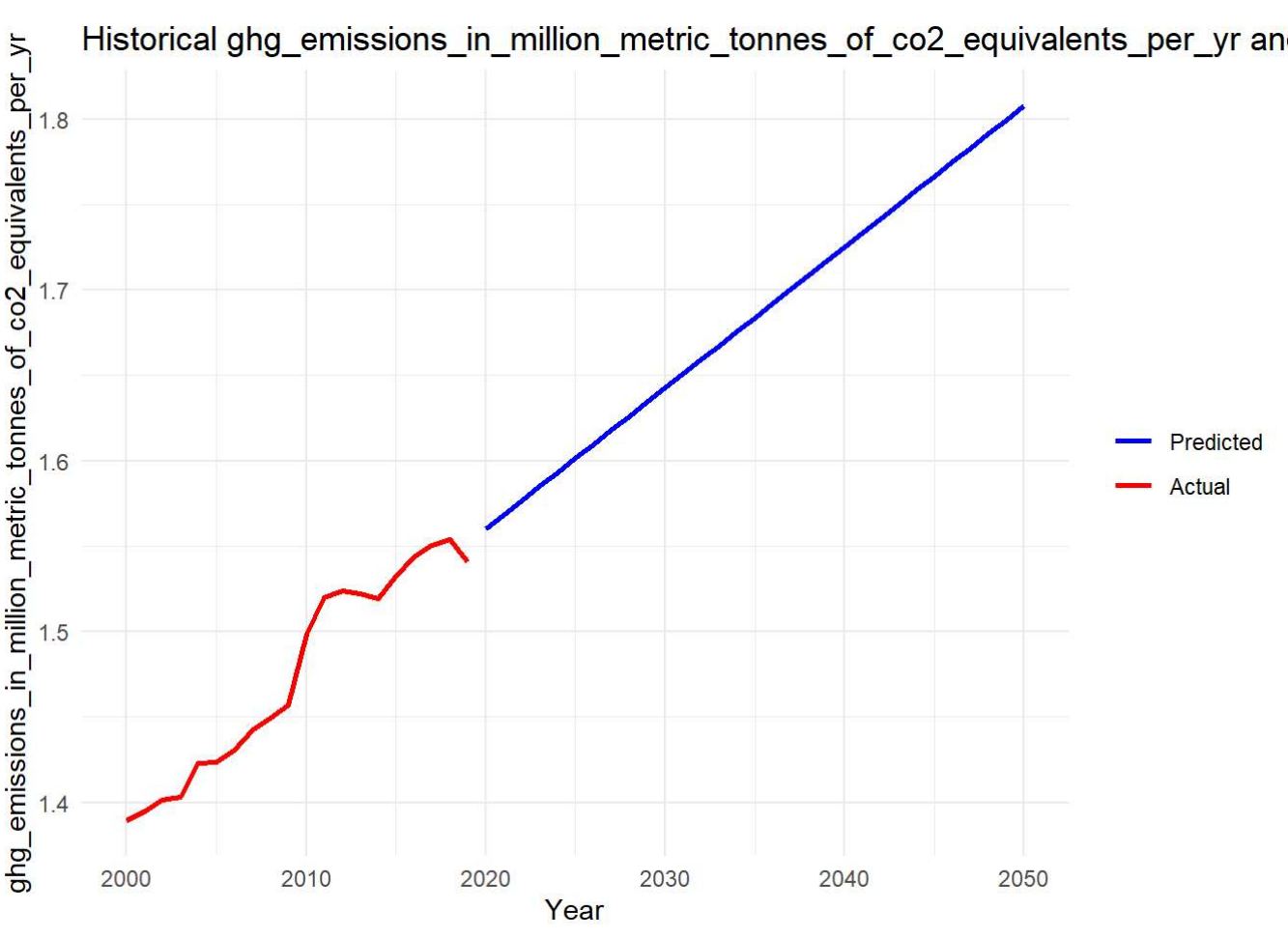

Historical co2_emissions_in_tonnes and Ensemble Predictions for KOR
co2_emissions_in_tonnes
1.9
1.8
1.7
1.6
1.5
2000
2010
2020
2030
2040
2050
Year
Predicted
Actual


Historical air_quality_in_mg_per_m3 and Ensemble Predictions for KOR
air_quality_in_mg_per_m3
0.4
0.3
0.2
0.1
0.0
-0.1
2000
2010
2020
2030
2040
2050
Year
Predicted
Actual


Historical land_area_in_1000_ha and Ensemble Predictions for KOR
land_area_in_1000_ha
-0.048
-0.052
-0.056
-0.060
2000
2010
2020
2030
2040
2050
Year
Predicted
Actual


Historical forest_area_in_1000_ha and Ensemble Predictions for KOR
forest_area_in_1000_ha
0.41
0.40
0.39
2000
2010
2020
2030
2040
2050
Year
Predicted
Actual


Historical carbon_stocks_in_millions_tonnes and Ensemble Predictions for KOR
carbon_stocks_in_millions_tonnes
0.8
0.6
0.4
0.2
2000
2010
2020
2030
2040
2050
Year
Predicted
Actual


Historical ghg_emissions_in_million_metric_tonnes_of_co2_equivalents_per_yr and
ghg_emissions_in_million_metric_tonnes_of_co2_equivalents_p
1.8
1.7
1.6
1.5
1.4
2000
2010
2020
2030
2040
2050
Year
Predicted
Actual