Analyzing the Impact of Climate Variables on Crop Yields - CO2, Temperature, and Precipitation

Introduction

In this section, we will focus on modeling the relationship between climate variables and crop yields. Specifically, we will investigate the effects of CO2 levels, temperature, and precipitation on crop productivity. By examining these variables, we aim to gain insights into how changes in climate conditions can impact agricultural outcomes.

The dependencies include -

- 1) R version: 4.1.2 (2021-11-01) and above
- 2) Installing the required libraries (as mentioned below)
- 3) Following Source files placed in the working directory of R environment. The files are publicly available in Kaggle:
  - **GlobalTemperatures.csv**: **Kaggle**
  - **archive.csv**: **Kaggle**
  - **crop_production.csv**: **Kaggle**
  - **climate_change_data.csv**: **Kaggle**

Step 1: Pre-Steps

Install the following required libraries (if not installed already in your local R env)

```r
# install.packages('dplyr') install.packages('reshape2')
# install.packages('tidyverse') install.packages('ggplot2')
# install.packages('lubridate') install.packages('caret')
# install.packages('dplyr') install.packages('class') install.packages('maps')
# install.packages('corrplot') install.packages('glmnet')
# install.packages('plotly') install.packages('gridExtra')

# Load the required libraries
suppressWarnings(suppressMessages(library(dplyr)))  # for %>%
suppressWarnings(suppressMessages(library(reshape2)))
suppressWarnings(suppressMessages(library(tidyverse)))
suppressWarnings(suppressMessages(library(ggplot2)))
suppressWarnings(suppressMessages(library(lubridate)))
suppressWarnings(suppressMessages(library(caret)))
suppressWarnings(suppressMessages(library(dplyr)))
suppressWarnings(suppressMessages(library(class)))
suppressWarnings(suppressMessages(library(maps)))
suppressWarnings(suppressMessages(library(corrplot)))
suppressWarnings(suppressMessages(library(glmnet)))
```

```r
suppressWarnings(suppressMessages(library(plotly)))
suppressWarnings(suppressMessages(library(gridExtra)))
```

Step 2: Loading Temperature & CO2 data, Data cleansing & Exploratory data analysis

Loading Global Temperature Data file - GlobalTemperatures.csv:

The dataset is sourced from Kaggle and comprises various columns, such as "dt" (date), "LandAverageTemperature," "LandAverageTemperatureUncertainty," "LandMaxTemperature," "LandMaxTemperatureUncertainty," "LandMinTemperature," "LandMinTemperatureUncertainty," "LandAndOceanAverageTemperature," and "LandAndOceanAverageTemperatureUncertainty."

```r
# Read data
data_global <- read.csv("GlobalTemperatures.csv")

head(data_global, 10)
```

```
##            dt LandAverageTemperature LandAverageTemperatureUncertainty
## 1  1750-01-01                  3.034                             3.574
## 2  1750-02-01                  3.083                             3.702
## 3  1750-03-01                  5.626                             3.076
## 4  1750-04-01                  8.490                             2.451
## 5  1750-05-01                 11.573                             2.072
## 6  1750-06-01                 12.937                             1.724
## 7  1750-07-01                 15.868                             1.911
## 8  1750-08-01                 14.750                             2.231
## 9  1750-09-01                 11.413                             2.637
## 10 1750-10-01                  6.367                             2.668
##    LandMaxTemperature LandMaxTemperatureUncertainty LandMinTemperature
## 1                  NA                            NA                 NA
## 2                  NA                            NA                 NA
## 3                  NA                            NA                 NA
## 4                  NA                            NA                 NA
## 5                  NA                            NA                 NA
## 6                  NA                            NA                 NA
## 7                  NA                            NA                 NA
## 8                  NA                            NA                 NA
## 9                  NA                            NA                 NA
## 10                 NA                            NA                 NA
##    LandMinTemperatureUncertainty LandAndOceanAverageTemperature
## 1                             NA                             NA
## 2                             NA                             NA
## 3                             NA                             NA
## 4                             NA                             NA
## 5                             NA                             NA
## 6                             NA                             NA
## 7                             NA                             NA
## 8                             NA                             NA
## 9                             NA                             NA
## 10                            NA                             NA
##    LandAndOceanAverageTemperatureUncertainty
```

```
## 1                                         NA
## 2                                         NA
## 3                                         NA
## 4                                         NA
## 5                                         NA
## 6                                         NA
## 7                                         NA
## 8                                         NA
## 9                                         NA
## 10                                        NA
```

```r
nrow(data_global)
```

```
## [1] 3192
```

```r
summary(data_global)
```

```
##       dt             LandAverageTemperature LandAverageTemperatureUncertainty
##  Length:3192        Min.   :-2.080         Min.   :0.0340
##  Class :character   1st Qu.: 4.312         1st Qu.:0.1867
##  Mode  :character   Median : 8.611         Median :0.3920
##                     Mean   : 8.375         Mean   :0.9385
##                     3rd Qu.:12.548         3rd Qu.:1.4192
##                     Max.   :19.021         Max.   :7.8800
##                     NA's   :12             NA's   :12
##  LandMaxTemperature LandMaxTemperatureUncertainty LandMinTemperature
##  Min.   : 5.90      Min.   :0.0440                Min.   :-5.407
##  1st Qu.:10.21      1st Qu.:0.1420                1st Qu.:-1.335
##  Median :14.76      Median :0.2520                Median : 2.950
##  Mean   :14.35      Mean   :0.4798                Mean   : 2.744
##  3rd Qu.:18.45      3rd Qu.:0.5390                3rd Qu.: 6.779
##  Max.   :21.32      Max.   :4.3730                Max.   : 9.715
##  NA's   :1200       NA's   :1200                  NA's   :1200
##  LandMinTemperatureUncertainty LandAndOceanAverageTemperature
##  Min.   :0.0450                Min.   :12.47
##  1st Qu.:0.1550                1st Qu.:14.05
##  Median :0.2790                Median :15.25
##  Mean   :0.4318                Mean   :15.21
##  3rd Qu.:0.4582                3rd Qu.:16.40
##  Max.   :3.4980                Max.   :17.61
##  NA's   :1200                  NA's   :1200
##  LandAndOceanAverageTemperatureUncertainty
##  Min.   :0.0420
##  1st Qu.:0.0630
##  Median :0.1220
##  Mean   :0.1285
##  3rd Qu.:0.1510
##  Max.   :0.4570
##  NA's   :1200
```

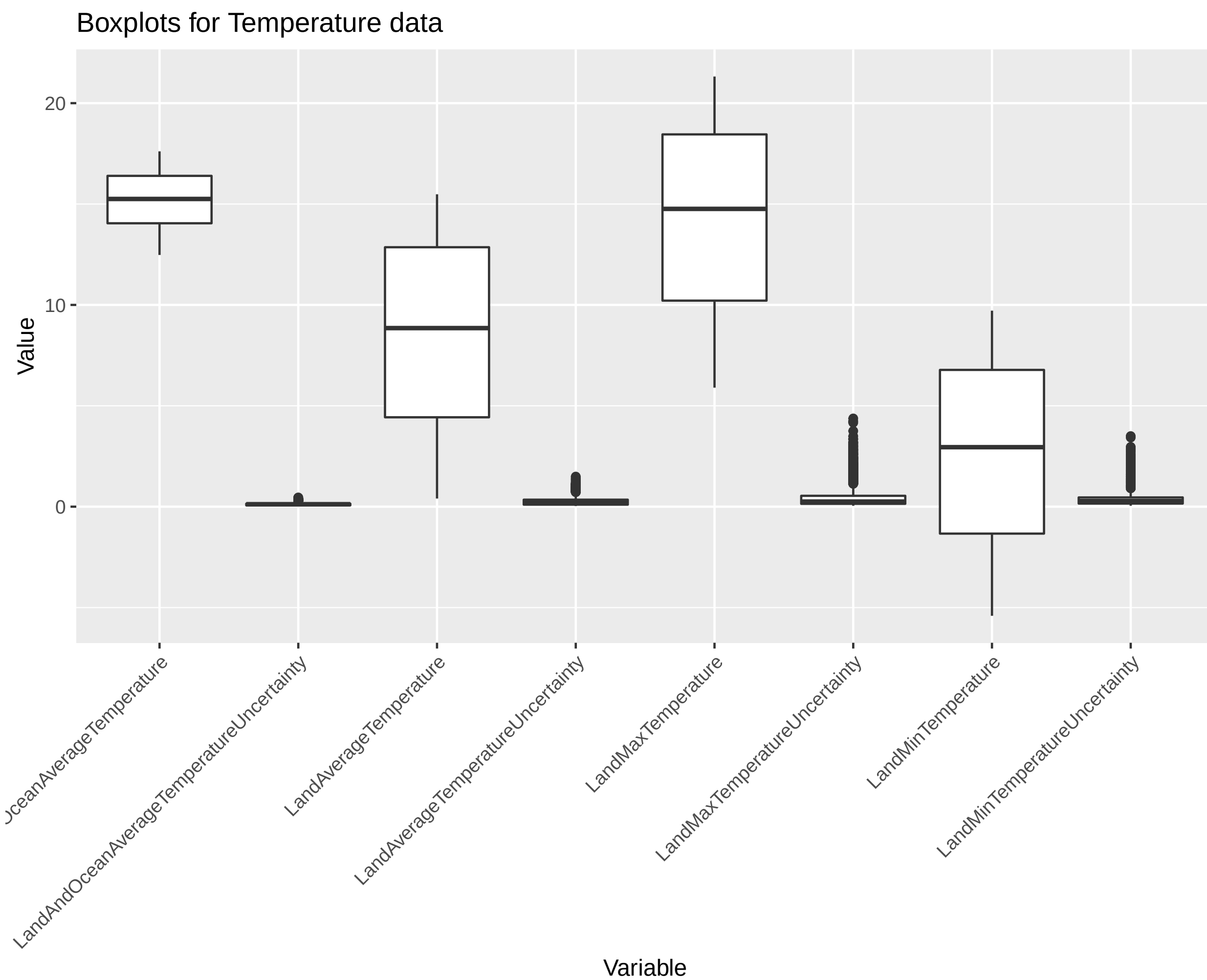
Data cleansing and Feature Engineering:

```r
# converting datatype of 'dt' column from character to Date
data_global$dt <- as.Date(data_global$dt)

# Dropping 'NA' rows
data_global <- data_global %>%
    drop_na()

summary(data_global)
```

```
##        dt              LandAverageTemperature LandAverageTemperatureUncertainty
##  Min.   :1850-01-01   Min.   : 0.404          Min.   :0.03400
##  1st Qu.:1891-06-23   1st Qu.: 4.430          1st Qu.:0.09975
##  Median :1932-12-16   Median : 8.851          Median :0.23000
##  Mean   :1932-12-16   Mean   : 8.572          Mean   :0.27666
##  3rd Qu.:1974-06-08   3rd Qu.:12.858          3rd Qu.:0.34725
##  Max.   :2015-12-01   Max.   :15.482          Max.   :1.49200
##  LandMaxTemperature LandMaxTemperatureUncertainty LandMinTemperature
##  Min.   : 5.90      Min.   :0.0440                Min.   :-5.407
##  1st Qu.:10.21      1st Qu.:0.1420                1st Qu.:-1.335
##  Median :14.76      Median :0.2520                Median : 2.950
##  Mean   :14.35      Mean   :0.4798                Mean   : 2.744
##  3rd Qu.:18.45      3rd Qu.:0.5390                3rd Qu.: 6.779
##  Max.   :21.32      Max.   :4.3730                Max.   : 9.715
##  LandMinTemperatureUncertainty LandAndOceanAverageTemperature
##  Min.   :0.0450                Min.   :12.47
##  1st Qu.:0.1550                1st Qu.:14.05
##  Median :0.2790                Median :15.25
##  Mean   :0.4318                Mean   :15.21
##  3rd Qu.:0.4582                3rd Qu.:16.40
##  Max.   :3.4980                Max.   :17.61
##  LandAndOceanAverageTemperatureUncertainty
##  Min.   :0.0420
##  1st Qu.:0.0630
##  Median :0.1220
##  Mean   :0.1285
##  3rd Qu.:0.1510
##  Max.   :0.4570
```

Checking for outliers in the data using boxplots:

```r
# Reshape the data to long format
data_global_long <- data_global %>%
    pivot_longer(cols = -dt)

# Create the boxplot
plt <- ggplot(data_global_long, aes(x = name, y = value)) + geom_boxplot() + labs(title =
↪  "Boxplots for Temperature data",
    x = "Variable", y = "Value")

# Rotate x-axis labels for better visibility
plt + theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



- **Inference:**
  - From the boxplots, we can see that the column in focus for our analysis - LandAndOceanAverageTemperature, doesn't have any outliers

Data Standardization: Calculate Anomaly for global temperature:

- **Why calculate Anomaly in temperature?**
  - Calculating the anomaly for global temperature is important in model building because it helps to capture the deviations or variations in temperature from a reference period. By calculating the anomaly, we can focus on the changes in temperature rather than the absolute temperature values. This is particularly useful in climate studies and modeling because it allows us to analyze and understand trends, patterns, and the impact of factors such as greenhouse gas emissions and human activities on temperature changes.

  - The anomaly provides a more meaningful measure as it takes into account the long-term average temperature for a specific reference period. It helps to remove the effects of seasonal variations and short-term fluctuations, allowing us to focus on the underlying changes in temperature over time. This anomaly data can then be used as a predictor variable in statistical models to assess its relationship with other variables and make predictions or inferences about future temperature trends.

  – Overall, calculating the anomaly for global temperature is a valuable step in model building as it provides a standardized and consistent metric for analyzing temperature changes and their drivers in the context of climate research and prediction.

We have chosen the period from 1945 to 1990 to calculate the temperature anomaly. This selection is based on the rationale that it covers the post-World War II era (starting from 1945) and encompasses the period of peak industrialization (until 1990). By considering this time frame, we aim to establish a baseline that avoids the potential influence of war-related pollution and focuses on the impact of industrial activities on temperature rise.

```
annual_mean_global <- data_global %>%
    mutate(year = lubridate::year(dt)) %>%
    group_by(year) %>%
    summarise(mean_temperature = mean(LandAndOceanAverageTemperature)) %>%
    mutate(reference_temperature_global = mean(mean_temperature[year %in% 1945:1990]),
        Anomaly = mean_temperature - reference_temperature_global)


head(annual_mean_global, 10)
```

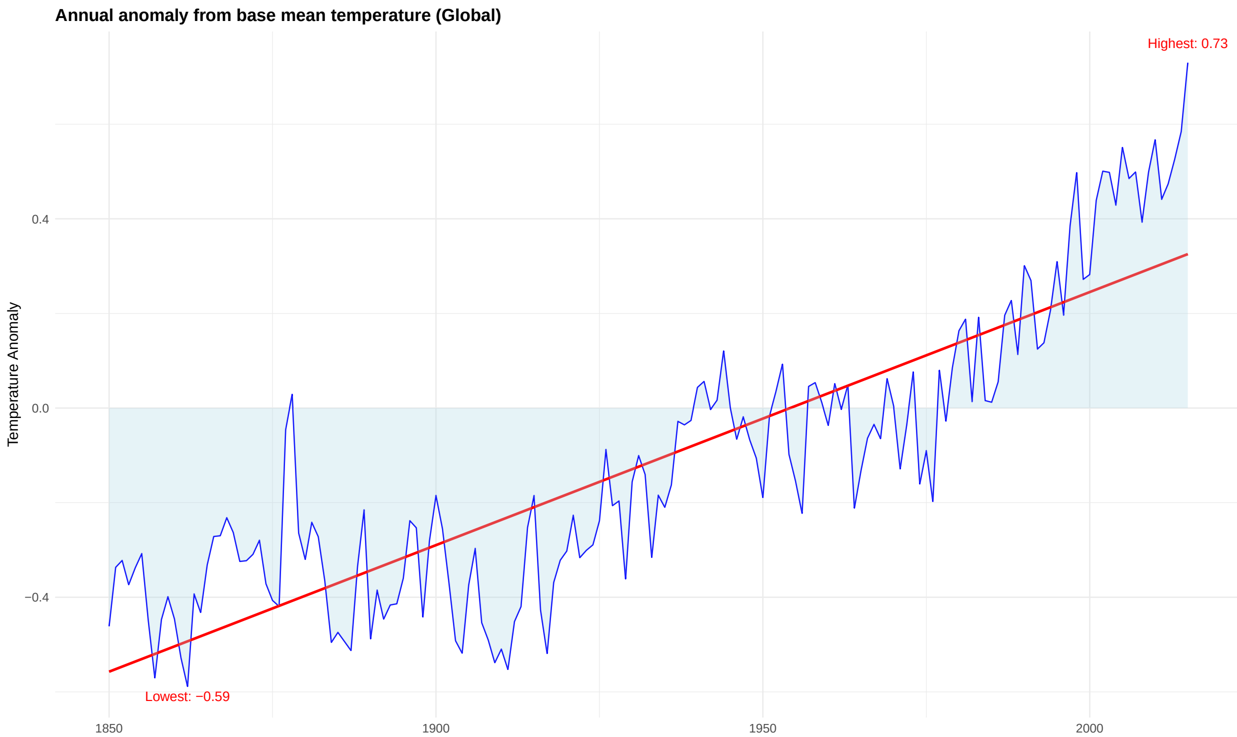
```
## # A tibble: 10 x 4
##     year mean_temperature reference_temperature_global Anomaly
##    <dbl>            <dbl>                        <dbl>   <dbl>
##  1  1850             14.9                         15.3  -0.461
##  2  1851             15.0                         15.3  -0.337
##  3  1852             15.0                         15.3  -0.322
##  4  1853             15.0                         15.3  -0.373
##  5  1854             15.0                         15.3  -0.338
##  6  1855             15.0                         15.3  -0.307
##  7  1856             14.9                         15.3  -0.449
##  8  1857             14.8                         15.3  -0.570
##  9  1858             14.9                         15.3  -0.447
## 10  1859             14.9                         15.3  -0.399
```

Plotting the temperature anomaly over the years -

```
# Find the highest and lowest data points
highest_point <- annual_mean_global[which.max(annual_mean_global$Anomaly), ]
lowest_point <- annual_mean_global[which.min(annual_mean_global$Anomaly), ]


plt <- ggplot() + geom_line(data = annual_mean_global, aes(x = year, y = Anomaly),
    color = "blue") + geom_smooth(data = annual_mean_global, aes(x = year, y = Anomaly),
    method = "lm", se = FALSE, color = "red") + geom_text(data = highest_point, aes(x =
    ↪  year,
    y = Anomaly, label = paste0("Highest: ", round(Anomaly, 2))), vjust = -1.5, color =
    ↪  "red") +
    geom_text(data = lowest_point, aes(x = year, y = Anomaly, label = paste0("Lowest: ",
        round(Anomaly, 2))), vjust = 1.5, color = "red") + geom_ribbon(data =
        ↪  annual_mean_global,
    aes(x = year, ymin = 0, ymax = Anomaly), fill = "lightblue", alpha = 0.3) +
    ↪  theme_minimal() +
```

```r
    labs(title = "Annual anomaly from base mean temperature (Global)", x = "", y =
    ↪  "Temperature Anomaly") +
    theme(plot.title = element_text(size = 14, face = "bold"), axis.title =
    ↪  element_text(size = 12),
        axis.text = element_text(size = 10))

suppressWarnings(suppressMessages(plt))
```

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



- **Inference:**
  - Based on the plot, we can observe an increasing trend in the temperature anomaly over the years.
  - The temperature anomaly has risen by more than 0.4 degrees, reaching its highest point at around 0.73 degrees.
  - This indicates a significant deviation from the reference temperature, suggesting a potential warming trend or environmental change.

Loading CO2 data file - archive.csv:

The data file "archive.csv" is sourced from Kaggle and contains information about carbon dioxide (CO2) levels. The dataset includes columns such as "Year", "Month", "Decimal.Date", "Carbon.Dioxide..ppm.","Seasonally.Adjusted.CO2..ppm.","Carbon.Dioxide.Fit..ppm." and "Seasonally.Adjusted.CO2.Fit..ppm." The data provides insights into historical CO2 measurements and seasonal adjustments, allowing for further analysis and exploration of carbon dioxide trends over time.

```r
co2_ppm <- read.csv("archive.csv")

head(co2_ppm, 10)
```

```
##    Year Month Decimal.Date Carbon.Dioxide..ppm. Seasonally.Adjusted.CO2..ppm.
## 1  1958     1     1958.041                   NA                            NA
## 2  1958     2     1958.126                   NA                            NA
## 3  1958     3     1958.203               315.69                        314.42
## 4  1958     4     1958.288               317.45                        315.15
## 5  1958     5     1958.370               317.50                        314.73
## 6  1958     6     1958.455                   NA                            NA
## 7  1958     7     1958.537               315.86                        315.17
## 8  1958     8     1958.622               314.93                        316.17
## 9  1958     9     1958.707               313.21                        316.06
## 10 1958    10     1958.789                   NA                            NA
##    Carbon.Dioxide.Fit..ppm. Seasonally.Adjusted.CO2.Fit..ppm.
## 1                        NA                                NA
## 2                        NA                                NA
## 3                    316.18                            314.89
## 4                    317.30                            314.98
## 5                    317.83                            315.06
## 6                    317.22                            315.14
## 7                    315.87                            315.21
## 8                    314.01                            315.29
## 9                    312.48                            315.35
## 10                   312.45                            315.40
```

```r
nrow(co2_ppm)
```

```
## [1] 720
```

```r
summary(co2_ppm)
```

```
##       Year          Month         Decimal.Date  Carbon.Dioxide..ppm.
##  Min.   :1958   Min.   : 1.00   Min.   :1958   Min.   :313.2
##  1st Qu.:1973   1st Qu.: 3.75   1st Qu.:1973   1st Qu.:328.6
##  Median :1988   Median : 6.50   Median :1988   Median :349.8
##  Mean   :1988   Mean   : 6.50   Mean   :1988   Mean   :352.4
##  3rd Qu.:2002   3rd Qu.: 9.25   3rd Qu.:2003   3rd Qu.:373.2
##  Max.   :2017   Max.   :12.00   Max.   :2018   Max.   :407.6
##                                                NA's   :17
##  Seasonally.Adjusted.CO2..ppm. Carbon.Dioxide.Fit..ppm.
##  Min.   :314.4                 Min.   :312.4
##  1st Qu.:329.0                 1st Qu.:328.3
##  Median :349.8                 Median :349.4
##  Mean   :352.4                 Mean   :352.1
##  3rd Qu.:372.9                 3rd Qu.:372.8
##  Max.   :406.0                 Max.   :407.3
##  NA's   :17                    NA's   :13
##  Seasonally.Adjusted.CO2.Fit..ppm.
##  Min.   :314.9
```

```
##  1st Qu.:328.4
##  Median :349.3
##  Mean   :352.0
##  3rd Qu.:372.6
##  Max.   :405.8
##  NA's   :13
```

```r
# Handling missing values:
co2_ppm <- na.omit(co2_ppm)

nrow(co2_ppm)
```

```
## [1] 702
```

```r
summary(co2_ppm)
```

```
##       Year          Month         Decimal.Date  Carbon.Dioxide..ppm.
##  Min.   :1958   Min.   : 1.000   Min.   :1958   Min.   :313.2
##  1st Qu.:1973   1st Qu.: 4.000   1st Qu.:1973   1st Qu.:328.6
##  Median :1987   Median : 7.000   Median :1988   Median :349.7
##  Mean   :1987   Mean   : 6.517   Mean   :1988   Mean   :352.3
##  3rd Qu.:2002   3rd Qu.: 9.750   3rd Qu.:2002   3rd Qu.:373.1
##  Max.   :2017   Max.   :12.000   Max.   :2017   Max.   :407.6
##  Seasonally.Adjusted.CO2..ppm. Carbon.Dioxide.Fit..ppm.
##  Min.   :314.4                 Min.   :312.5
##  1st Qu.:329.0                 1st Qu.:328.5
##  Median :349.7                 Median :349.9
##  Mean   :352.3                 Mean   :352.3
##  3rd Qu.:372.8                 3rd Qu.:373.2
##  Max.   :406.0                 Max.   :407.3
##  Seasonally.Adjusted.CO2.Fit..ppm.
##  Min.   :314.9
##  1st Qu.:329.2
##  Median :349.8
##  Mean   :352.3
##  3rd Qu.:372.9
##  Max.   :405.8
```

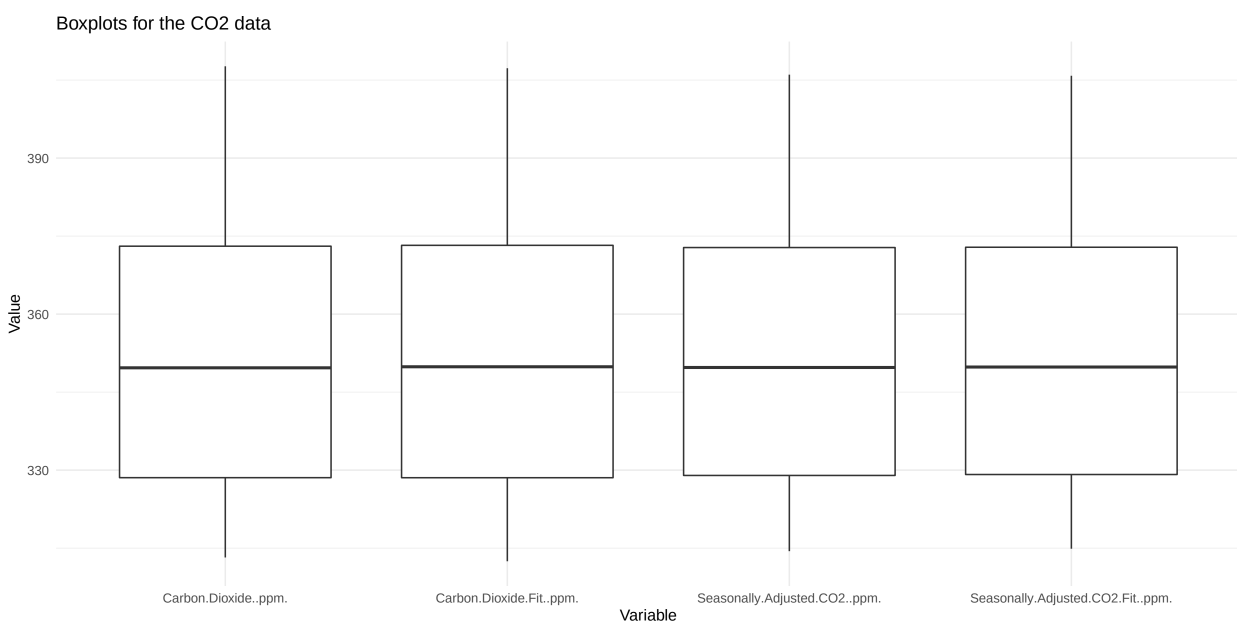
```r
head(co2_ppm, 10)
```

```
##    Year Month Decimal.Date Carbon.Dioxide..ppm. Seasonally.Adjusted.CO2..ppm.
## 3  1958     3     1958.203               315.69                        314.42
## 4  1958     4     1958.288               317.45                        315.15
## 5  1958     5     1958.370               317.50                        314.73
## 7  1958     7     1958.537               315.86                        315.17
## 8  1958     8     1958.622               314.93                        316.17
## 9  1958     9     1958.707               313.21                        316.06
## 11 1958    11     1958.874               313.33                        315.20
## 12 1958    12     1958.956               314.67                        315.44
## 13 1959     1     1959.041               315.58                        315.56
## 14 1959     2     1959.126               316.48                        315.88
```

```
##    Carbon.Dioxide.Fit..ppm. Seasonally.Adjusted.CO2.Fit..ppm.
## 3                    316.18                            314.89
## 4                    317.30                            314.98
## 5                    317.83                            315.06
## 7                    315.87                            315.21
## 8                    314.01                            315.29
## 9                    312.48                            315.35
## 11                   313.61                            315.46
## 12                   314.75                            315.51
## 13                   315.60                            315.57
## 14                   316.24                            315.63
```

Checking for outliers in the CO2 data:

```
# Create a melted version of the data
melted_data <- co2_ppm %>%
    pivot_longer(cols = -c(Year, Month, Decimal.Date), names_to = "Variable", values_to =
    ↪  "Value")

# Create the box plots for each column
plt <- ggplot(melted_data, aes(x = Variable, y = Value)) + geom_boxplot() + labs(title =
↪  "Boxplots for the CO2 data",
    x = "Variable", y = "Value") + theme_minimal()

# Display the box plots
print(plt)
```



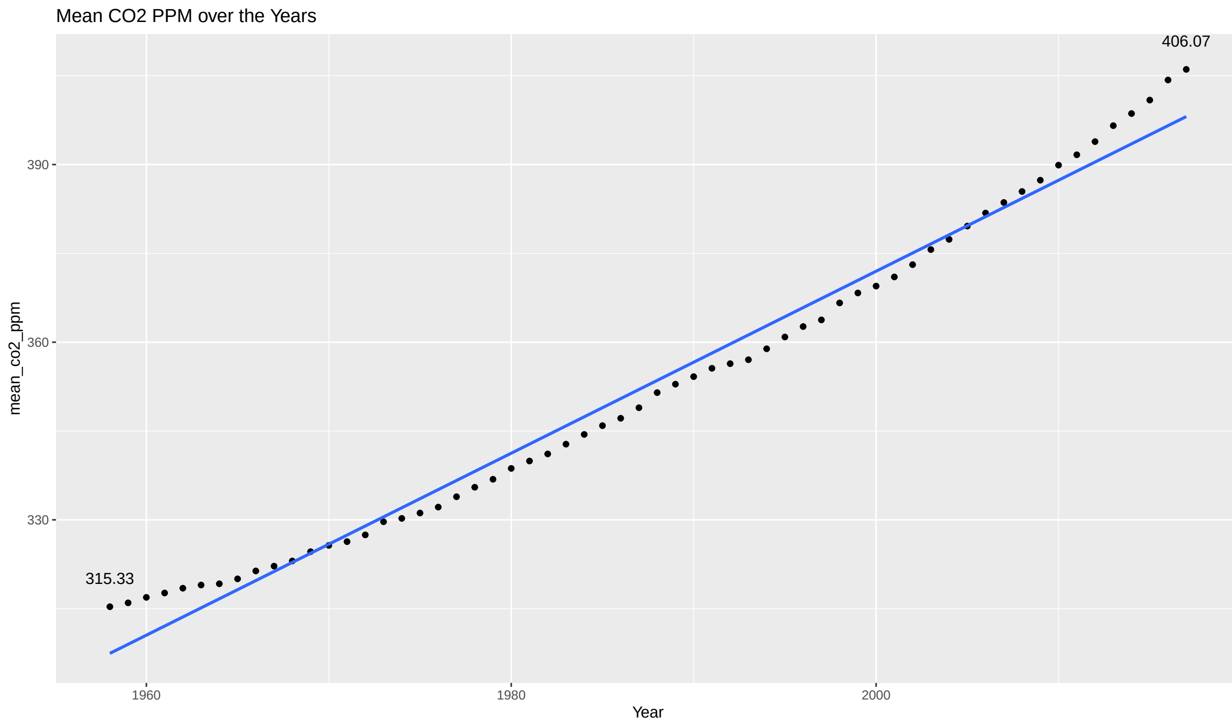
- **Inference:**
  - The box plots reveal that there are no data points outside the whiskers in the CO2 dataset, indicating the absence of outliers.

Data Standardization: Calculate annual mean CO2 levels

```r
annual_co2_ppm <- co2_ppm %>%
    group_by(Year) %>%
    summarise(mean_co2_ppm = mean(Carbon.Dioxide..ppm.))

head(annual_co2_ppm, 10)
```

```
## # A tibble: 10 x 2
##     Year mean_co2_ppm
##    <int>        <dbl>
##  1  1958         315.
##  2  1959         316.
##  3  1960         317.
##  4  1961         318.
##  5  1962         318.
##  6  1963         319.
##  7  1964         319.
##  8  1965         320.
##  9  1966         321.
## 10  1967         322.
```

```r
# Calculate the highest and lowest mean_co2_ppm values
highest_value <- max(annual_co2_ppm$mean_co2_ppm)
lowest_value <- min(annual_co2_ppm$mean_co2_ppm)

# Create the plot with a regression line and labels
plt <- ggplot(annual_co2_ppm, aes(x = Year, y = mean_co2_ppm)) + geom_point() +
↪   geom_smooth(method = "lm",
    se = FALSE) + geom_text(aes(x = Year, y = mean_co2_ppm, label = ifelse(mean_co2_ppm
    ↪  ==
    highest_value, mean_co2_ppm, ifelse(mean_co2_ppm == lowest_value, mean_co2_ppm,
    ""))), vjust = -1.5, nudge_y = 1) + labs(title = "Mean CO2 PPM over the Years",
    x = "Year", y = "mean_co2_ppm")

# Display the plot without warnings or messages
suppressWarnings(suppressMessages(print(plt)))
```



- **Inference:**
  - We can observe a positive trend in the mean CO2 ppm over the years, i.e., the data reveals a steady increase in the mean CO2 ppm, indicating a potential correlation with factors such as industrialization and human activities contributing to greenhouse gas emissions.
  - The dataset includes a wide range of mean CO2 ppm values, with the highest recorded value reaching 406.07 and the lowest observed at 315.33.
  - These extreme values highlight the significant variability in CO2 levels and emphasize the importance of monitoring and managing carbon dioxide emissions to mitigate their potential impact on the environment and climate.

Step 3: Merging Temperature & CO2 data

Merging Temperature and CO2 data:

```
head(annual_mean_global)
```

```
## # A tibble: 6 x 4
##    year mean_temperature reference_temperature_global Anomaly
##   <dbl>            <dbl>                        <dbl>   <dbl>
## 1  1850             14.9                         15.3  -0.461
## 2  1851             15.0                         15.3  -0.337
## 3  1852             15.0                         15.3  -0.322
## 4  1853             15.0                         15.3  -0.373
## 5  1854             15.0                         15.3  -0.338
## 6  1855             15.0                         15.3  -0.307
```

```r
head(annual_co2_ppm)
```

```
## # A tibble: 6 x 2
##    Year mean_co2_ppm
##   <int>        <dbl>
## 1  1958         315.
## 2  1959         316.
## 3  1960         317.
## 4  1961         318.
## 5  1962         318.
## 6  1963         319.
```

```r
merged_data <- merge(annual_mean_global, annual_co2_ppm, by.x = "year", by.y = "Year",
    all = TRUE)

head(merged_data)
```

```
##   year mean_temperature reference_temperature_global    Anomaly mean_co2_ppm
## 1 1850         14.86717                    15.32852 -0.4613514           NA
## 2 1851         14.99183                    15.32852 -0.3366848           NA
## 3 1852         15.00650                    15.32852 -0.3220181           NA
## 4 1853         14.95517                    15.32852 -0.3733514           NA
## 5 1854         14.99100                    15.32852 -0.3375181           NA
## 6 1855         15.02108                    15.32852 -0.3074348           NA
```

```r
### Dropping null values
merged_data <- merged_data %>%
    drop_na()

head(merged_data)
```

```
##   year mean_temperature reference_temperature_global      Anomaly mean_co2_ppm
## 1 1958         15.38208                    15.32852  0.053565217     315.3300
## 2 1959         15.34050                    15.32852  0.011981884     315.9817
## 3 1960         15.29192                    15.32852 -0.036601449     316.9083
## 4 1961         15.37992                    15.32852  0.051398551     317.6450
## 5 1962         15.32558                    15.32852 -0.002934783     318.4533
## 6 1963         15.37667                    15.32852  0.048148551     318.9925
```

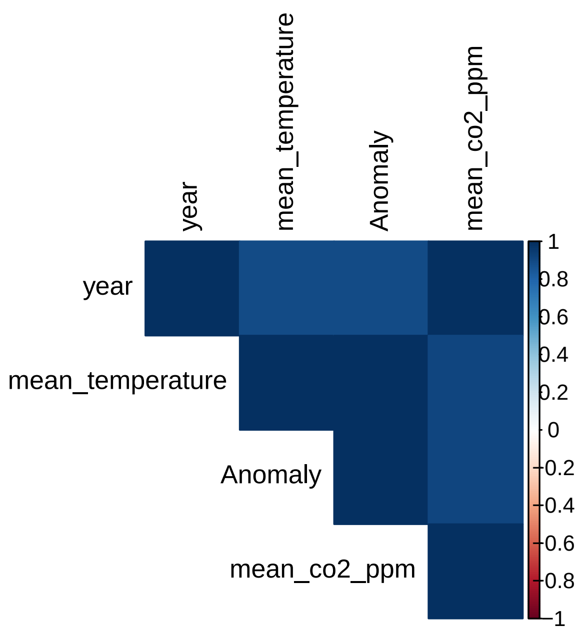
```r
summary(merged_data)
```

```
##       year      mean_temperature reference_temperature_global
##  Min.   :1958   Min.   :15.12    Min.   :15.33
##  1st Qu.:1972   1st Qu.:15.34    1st Qu.:15.33
##  Median :1986   Median :15.48    Median :15.33
##  Mean   :1986   Mean   :15.52    Mean   :15.33
##  3rd Qu.:2001   3rd Qu.:15.75    3rd Qu.:15.33
##  Max.   :2015   Max.   :16.06    Max.   :15.33
##     Anomaly          mean_co2_ppm
##  Min.   :-0.21143   Min.   :315.3
```

```
##  1st Qu.: 0.01202   1st Qu.:328.0
##  Median : 0.15061   Median :348.0
##  Mean   : 0.19066   Mean   :351.0
##  3rd Qu.: 0.41973   3rd Qu.:370.6
##  Max.   : 0.73007   Max.   :400.9
```

Correlation Plot - Temperature vs CO2:

```r
# Select the variables of interest
variables <- c("year", "mean_temperature", "Anomaly", "mean_co2_ppm")
data_subset <- merged_data[variables]

# Compute the correlation matrix
cor_matrix <- cor(data_subset)

# Increase the margin size
par(mar = c(2, 2, 2, 5))

# Create the correlation plot
corrplot(cor_matrix, method = "color", type = "upper", tl.col = "black")
```




```r
print(cor_matrix)
```

```
##                       year mean_temperature   Anomaly mean_co2_ppm
## year             1.0000000        0.8932957 0.8932957    0.9923057
## mean_temperature 0.8932957        1.0000000 1.0000000    0.9172674
## Anomaly          0.8932957        1.0000000 1.0000000    0.9172674
## mean_co2_ppm     0.9923057        0.9172674 0.9172674    1.0000000
```

- **Inference:**
  - From the above plots, we can see a positive correlation between - Anomaly and mean_co2_ppm i.e., 0.9172674

Step 4: Model I: "Anomaly ~ mean_co2_ppm"

Why chose Linear regression for modeling the effects of climate variables on crop yields?

Linear regression is chosen for modeling the effects of climate variables on crop yields because it is a simple and interpretable statistical method that allows us to examine the relationship between one or more independent variables (climate variables) and a dependent variable (crop yields). The coefficients in a linear regression model have clear interpretations. For instance, the coefficient of a climate variable represents the change in crop yields associated with a one-unit change in that climate variable, assuming all other variables are held constant. This interpretability helps in understanding the impact of climate variables on crop yields. Linear regression assumes a linear relationship between the independent variables and the dependent variable. While this assumption may not hold in all cases, it often provides a reasonable approximation for many real-world scenarios. The performance of a linear regression model is usually assessed using metrics such as R-squared, adjusted R-squared, root mean squared error (RMSE), or mean absolute error (MAE).

The following linear regression model is created to predict Anomaly using mean_co2_ppm as the predictor:

```r
model <- lm(Anomaly ~ mean_co2_ppm, data = merged_data)

summary(model)
```

```
##
## Call:
## lm(formula = Anomaly ~ mean_co2_ppm, data = merged_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.22936 -0.08854  0.01664  0.07845  0.17471
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -2.7700258  0.1722344  -16.08   <2e-16 ***
## mean_co2_ppm  0.0084357  0.0004895   17.23   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.09481 on 56 degrees of freedom
## Multiple R-squared:  0.8414, Adjusted R-squared:  0.8385
## F-statistic:   297 on 1 and 56 DF,  p-value: < 2.2e-16
```

- **Inference:**

The linear regression model evaluated for the relationship between the Anomaly variable and mean CO2 ppm yields the following results:

- 1) The intercept of the model is -2.7700258 with a standard error of 0.1722344. The coefficient for the mean_co2_ppm predictor variable is 0.0084357 with a standard error of 0.0004895. Both the intercept and mean_co2_ppm coefficient are statistically significant ($p < 0.0000000000000002$), indicating a strong association between mean CO2 ppm and the Anomaly.
- 2) The model's performance is evaluated based on the residuals, which measure the difference between the observed Anomaly values and the predicted values from the model. The minimum residual is -0.22936, the first quartile is -0.08854, the median is 0.01664, the third quartile is 0.07845, and the maximum residual is 0.17471.

- 3) The model's overall performance is assessed using the residual standard error, which is calculated to be 0.09481. The multiple R-squared value is 0.8414, indicating that approximately 84.14% of the variance in the Anomaly variable is explained by the mean CO2 ppm predictor. The adjusted R-squared value, which accounts for the number of predictors in the model, is 0.8385.
- 4) The F-statistic, with a value of 297 and degrees of freedom of 1 and 56, assesses the overall significance of the model. The extremely low p-value (< 0.00000000000000022) indicates that the model is highly significant.

Overall, the model demonstrates a strong relationship between mean CO2 ppm and the Anomaly variable, explaining a significant proportion of the variance in the Anomaly.

Creating Train, Test and Prediction models for Model I:

The following steps are performed to create a train-test split and fit a linear regression model using the variable 'mean_co2_ppm' as the predictor and 'Anomaly' as the target variable:

```
# Step 1: Create a train-test split A train-test split is created using a seed
# value of 123 for reproducibility. The training set contains 70% of the data,
# and the test set contains the remaining data.
set.seed(123)  # Set seed for reproducibility
train_indices <- sample(nrow(merged_data), nrow(merged_data) * 0.7)  # 70% for training
train_data <- merged_data[train_indices, ]
test_data <- merged_data[-train_indices, ]

# Step 2: Fit a linear regression model A linear regression model is fitted
# using the training data.
model <- lm(Anomaly ~ mean_co2_ppm, data = train_data)

# Step 3: Predict on the test data
predictions <- predict(model, newdata = test_data)

# Step 4: Evaluate the model
actual_values <- test_data$Anomaly

# Calculate Mean Squared Error (MSE)
mse <- mean((actual_values - predictions)^2)

# Calculate Root Mean Squared Error (RMSE)
rmse <- sqrt(mse)

# Calculate Mean Absolute Error (MAE)
mae <- mean(abs(actual_values - predictions))

# Print the evaluation metrics
cat("MSE:", mse, "\n")
```

```
## MSE: 0.01272262
```

```
cat("RMSE:", rmse, "\n")
```

```
## RMSE: 0.1127946
```

```r
cat("MAE:", mae, "\n")
```

```
## MAE: 0.1024163
```

- **Inference:**

The model training and evaluation process is as follows:

- 1) Train-Test Split: The dataset is randomly split into a training set and a test set. In this case, 70% of the data is used for training, and the remaining 30% is used for testing.
- 2) Model Fitting: A linear regression model is fitted using the training data. The model predicts the Anomaly variable based on the mean_co2_ppm predictor.
- 3) Prediction: The trained model is used to predict the Anomaly values for the test data.
- 4) Evaluation: The predicted values are compared to the actual Anomaly values from the test data to evaluate the model's performance.

- 5) Mean Squared Error (MSE): It measures the average squared difference between the predicted and actual values. The MSE value is 0.01272262.
- 6) Root Mean Squared Error (RMSE): It is the square root of MSE and provides an interpretable estimate of the average prediction error. The RMSE value is 0.1127946.
- 7) Mean Absolute Error (MAE): It calculates the average absolute difference between the predicted and actual values. The MAE value is 0.1024163.

These evaluation metrics provide insights into the performance of the linear regression model in predicting the Anomaly variable based on the mean_co2_ppm predictor. Lower values of MSE, RMSE, and MAE indicate better model performance, suggesting that the model's predictions are relatively close to the actual values.

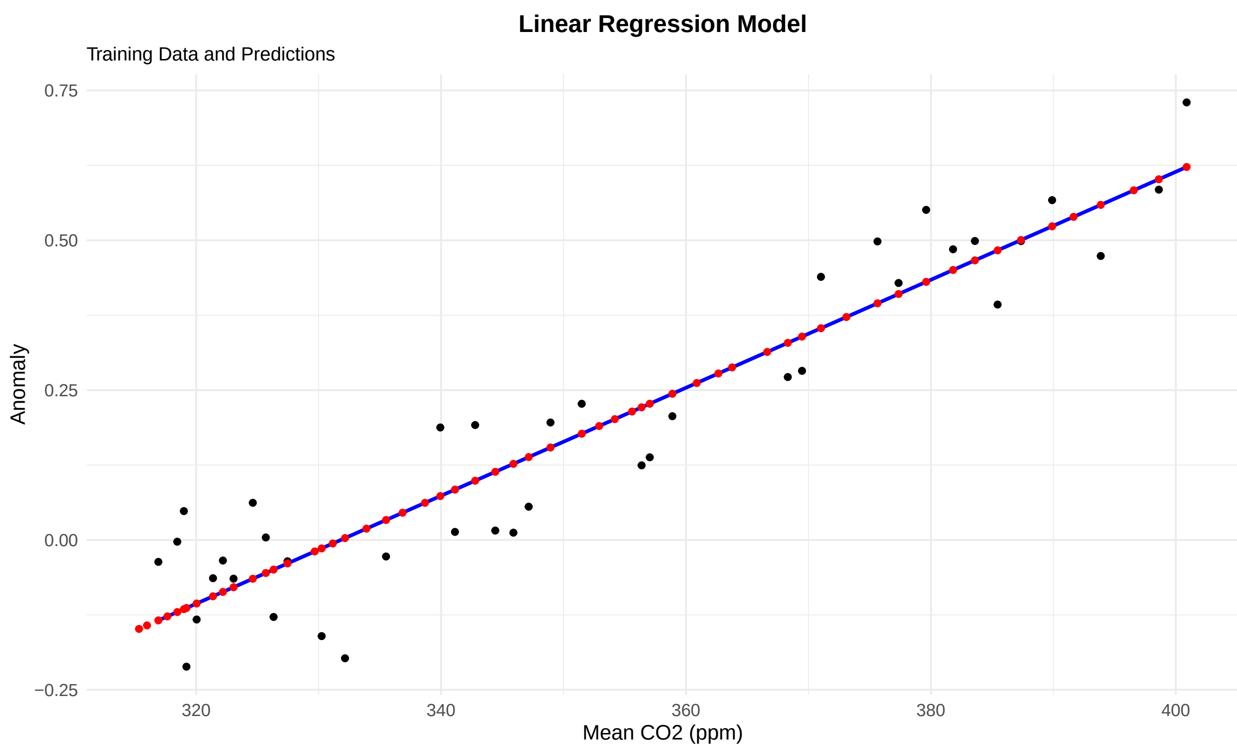
Note: Given the satisfactory performance of the model in terms of prediction, there is no need to perform hyperparameter optimization for model selection.

Visualization of the Model I:

```r
# Create a scatter plot of the training data
scatter_plot <- ggplot(data = train_data, aes(x = mean_co2_ppm, y = Anomaly)) +
↪  geom_point() +
    labs(x = "Mean CO2 (ppm)", y = "Anomaly") + theme_minimal()

# Add the regression line using the training data
reg_line_plot <- scatter_plot + geom_smooth(method = "lm", se = FALSE, color = "blue") +
    theme(plot.title = element_text(size = 14, face = "bold"), axis.title =
    ↪  element_text(size = 12),
        axis.text = element_text(size = 10))

# Predict the values using the linear regression model
merged_data$predicted <- predict(model, newdata = merged_data)

# Add the predicted values
```

```r
predictions_plot <- reg_line_plot + geom_point(data = merged_data, aes(x = mean_co2_ppm,
    y = predicted), color = "red") + labs(title = "Linear Regression Model", subtitle =
    ↪  "Training Data and Predictions",
    y = "Anomaly", color = "Predicted Values") + theme(plot.title = element_text(hjust =
    ↪  0.5),
    legend.position = "bottom")

# Display the plot
suppressWarnings(suppressMessages(predictions_plot))
```

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



- **Inference:**
  - The plot illustrates the relationship between the mean CO2 levels (x-axis) and the anomaly values (y-axis) using a scatter plot of the training data.
  - The plot includes a regression line fitted to the training data, represented by the blue line. Additionally, the predicted values from the linear regression model are shown as red points, providing an indication of how well the model aligns with the observed data.
  - The plot indicates that the predicted values (red points) tend to align closely with the regression line, suggesting that the linear regression model provides a reasonable fit to the training data. This alignment between the predicted values and the regression line supports the model's ability to capture the relationship between mean CO2 levels and the anomaly values.
  - Overall, the plot provides visual evidence of the model's performance in predicting the Temperature anomaly values based on mean CO2 levels, indicating a satisfactory fit between the observed data and the model's predictions.

Post model building/testing data clean-ups:

```r
head(merged_data, 10)
```

```
##    year mean_temperature reference_temperature_global      Anomaly mean_co2_ppm
## 1  1958         15.38208                     15.32852  0.053565217     315.3300
## 2  1959         15.34050                     15.32852  0.011981884     315.9817
## 3  1960         15.29192                     15.32852 -0.036601449     316.9083
## 4  1961         15.37992                     15.32852  0.051398551     317.6450
## 5  1962         15.32558                     15.32852 -0.002934783     318.4533
## 6  1963         15.37667                     15.32852  0.048148551     318.9925
## 7  1964         15.11708                     15.32852 -0.211434783     319.2022
## 8  1965         15.19575                     15.32852 -0.132768116     320.0358
## 9  1966         15.26467                     15.32852 -0.063851449     321.3700
## 10 1967         15.29417                     15.32852 -0.034351449     322.1800
##      predicted
## 1  -0.14845342
## 2  -0.14258293
## 3  -0.13423511
## 4  -0.12759890
## 5  -0.12031708
## 6  -0.11546004
## 7  -0.11357077
## 8  -0.10606124
## 9  -0.09404248
## 10 -0.08674565
```

```r
# Removing the 'predicted' column from the merged dataset
merged_data <- merged_data[, -6]

head(merged_data, 10)
```

```
##    year mean_temperature reference_temperature_global      Anomaly mean_co2_ppm
## 1  1958         15.38208                     15.32852  0.053565217     315.3300
## 2  1959         15.34050                     15.32852  0.011981884     315.9817
## 3  1960         15.29192                     15.32852 -0.036601449     316.9083
## 4  1961         15.37992                     15.32852  0.051398551     317.6450
## 5  1962         15.32558                     15.32852 -0.002934783     318.4533
## 6  1963         15.37667                     15.32852  0.048148551     318.9925
## 7  1964         15.11708                     15.32852 -0.211434783     319.2022
## 8  1965         15.19575                     15.32852 -0.132768116     320.0358
## 9  1966         15.26467                     15.32852 -0.063851449     321.3700
## 10 1967         15.29417                     15.32852 -0.034351449     322.1800
```

Step 5: Loading Crop Yield data, Data cleansing & Exploratory data analysis

Loading data file - crop_production.csv:

The dataset "crop_production.csv" is sourced from Kaggle, and it contains information related to crop yield for different countries over several years. The dataset consists of 20,566 rows and includes columns such as index, LOCATION (country code), INDICATOR (crop yield indicator), SUBJECT (type of crop), MEASURE (measurement unit), FREQUENCY (data frequency), TIME (year), Value (crop yield value), and Flag.Codes (optional flag codes). The dataset allows for further analysis of crop yields and their relationship with climate variables.

```r
# Load the data from the CSV file
crop_production <- read.csv("crop_production.csv")

nrow(crop_production)
```

```
## [1] 20566
```

```r
# View the first few rows of the data
head(crop_production, 10)
```

```
##    index LOCATION INDICATOR SUBJECT  MEASURE FREQUENCY TIME    Value Flag.Codes
## 1      0      AUS CROPYIELD    RICE TONNE_HA         A 1990 8.314607         NA
## 2      1      AUS CROPYIELD    RICE TONNE_HA         A 1991 8.394737         NA
## 3      2      AUS CROPYIELD    RICE TONNE_HA         A 1992 8.094340         NA
## 4      3      AUS CROPYIELD    RICE TONNE_HA         A 1993 8.336000         NA
## 5      4      AUS CROPYIELD    RICE TONNE_HA         A 1994 8.537815         NA
## 6      5      AUS CROPYIELD    RICE TONNE_HA         A 1995 7.051095         NA
## 7      6      AUS CROPYIELD    RICE TONNE_HA         A 1996 8.256579         NA
## 8      7      AUS CROPYIELD    RICE TONNE_HA         A 1997 9.006803         NA
## 9      8      AUS CROPYIELD    RICE TONNE_HA         A 1998 9.202703         NA
## 10     9      AUS CROPYIELD    RICE TONNE_HA         A 1999 8.274809         NA
```

Data Cleansing and Feature Engineering:

Remove the first column (index column) and last column (Flag.Codes), as they do not support our analysis -

```r
crop_production <- crop_production[, -c(1, ncol(crop_production))]

summary(crop_production)
```

```
##    LOCATION          INDICATOR           SUBJECT            MEASURE         
##  Length:20566       Length:20566       Length:20566       Length:20566      
##  Class :character   Class :character   Class :character   Class :character  
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character  
##                                                                             
##                                                                             
##                                                                             
##   FREQUENCY              TIME          Value          
##  Length:20566       Min.   :1970   Min.   :      0.0  
##  Class :character   1st Qu.:1999   1st Qu.:      2.0  
##  Mode  :character   Median :2008   Median :     25.6  
##                     Mean   :2008   Mean   :  12492.8  
##                     3rd Qu.:2017   3rd Qu.:   1563.0  
##                     Max.   :2025   Max.   :1146044.3
```

```r
nrow(crop_production)
```

```
## [1] 20566
```

Removing future years data - 2024 and 2025 as well, as they are future dated records and would not help our analysis -

```
crop_production <- crop_production[crop_production$TIME != 2024 & crop_production$TIME !=
    2025, ]

summary(crop_production)
```

```
##    LOCATION          INDICATOR           SUBJECT            MEASURE
##  Length:19416       Length:19416       Length:19416       Length:19416
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##   FREQUENCY              TIME          Value
##  Length:19416       Min.   :1970   Min.   :      0.0
##  Class :character   1st Qu.:1998   1st Qu.:      1.9
##  Mode  :character   Median :2007   Median :     25.0
##                     Mean   :2007   Mean   :  12242.2
##                     3rd Qu.:2015   3rd Qu.:   1531.3
##                     Max.   :2023   Max.   :1117103.5
```

Check for null values in each column -

```
null_columns <- colSums(is.na(crop_production))

# Display columns with null values
cols_with_null <- names(null_columns[null_columns > 0])

print(cols_with_null)
```

```
## character(0)
```

- **Inference:**
    - There are no null values in the attributes as evident from the data summary

Running few other data exploratory measures, we have -

```
nrow(crop_production)
```

```
## [1] 19416
```

```
unique(crop_production$INDICATOR)
```

```
## [1] "CROPYIELD"
```

```r
unique(crop_production$MEASURE)
```

```
## [1] "TONNE_HA"   "THND_TONNE" "THND_HA"
```

```r
unique(crop_production$FREQUENCY)
```

```
## [1] "A"
```

```r
unique(crop_production$SUBJECT)
```

```
## [1] "RICE"    "WHEAT"   "MAIZE"   "SOYBEAN"
```

- **Inference:**
  - The column "INDICATOR" has a unique value "CROPYIELD," indicating that the dataset focuses on crop yield-related information.
  - The column "MEASURE" has three unique values: "TONNE_HA," "THND_TONNE," and "THND_HA," representing different measurement units for crop yield.
  - The "FREQUENCY" column has a single unique value "A," indicating that the data is measured annually. The "SUBJECT" column includes four unique values, namely "RICE," "WHEAT," "MAIZE," and "SOYBEAN," representing different types of crops for which the crop yield data is recorded. This dataset provides comprehensive information on crop yields for various crops and can be used for further analysis and insights into the relationship between crop production and climate variables.

Also removing the INDICATOR and FREQUENCY column, as they are static/single-valued for the entire dataset, we have -

```r
crop_production <- crop_production %>%
    select(-c("INDICATOR", "FREQUENCY"))
```

Data Standardization: crop_production$MEASURE

To convert the units "TONNE_HA", "THND_TONNE", and "THND_HA" to their respective descriptions, we can use the following conversions:

- "TONNE_HA" represents metric tonnes per hectare.
- "THND_TONNE" represents thousand metric tonnes.
- "THND_HA" represents thousand hectares.

These conversions provide a way to express agricultural or land-related quantities in different units.

Converting "TONNE_HA" to "THND_TONNE" and "THND_HA" to "THND_TONNE" in the crop_production data frame -

```r
head(subset(crop_production, MEASURE == "TONNE_HA"), 10)
```

```
##    LOCATION SUBJECT  MEASURE TIME    Value
## 1       AUS    RICE TONNE_HA 1990 8.314607
## 2       AUS    RICE TONNE_HA 1991 8.394737
## 3       AUS    RICE TONNE_HA 1992 8.094340
## 4       AUS    RICE TONNE_HA 1993 8.336000
## 5       AUS    RICE TONNE_HA 1994 8.537815
## 6       AUS    RICE TONNE_HA 1995 7.051095
## 7       AUS    RICE TONNE_HA 1996 8.256579
## 8       AUS    RICE TONNE_HA 1997 9.006803
## 9       AUS    RICE TONNE_HA 1998 9.202703
## 10      AUS    RICE TONNE_HA 1999 8.274809
```

```r
# This code divides the values in the 'Value' column by 1000, effectively
# converting them from tonne per hectare to thousand tonne per hectare.
crop_production$Value <- with(crop_production, ifelse(MEASURE == "TONNE_HA", Value/1000,
    Value))

head(subset(crop_production, MEASURE == "TONNE_HA"), 10)
```

```
##    LOCATION SUBJECT  MEASURE TIME       Value
## 1       AUS    RICE TONNE_HA 1990 0.008314607
## 2       AUS    RICE TONNE_HA 1991 0.008394737
## 3       AUS    RICE TONNE_HA 1992 0.008094340
## 4       AUS    RICE TONNE_HA 1993 0.008336000
## 5       AUS    RICE TONNE_HA 1994 0.008537815
## 6       AUS    RICE TONNE_HA 1995 0.007051095
## 7       AUS    RICE TONNE_HA 1996 0.008256579
## 8       AUS    RICE TONNE_HA 1997 0.009006803
## 9       AUS    RICE TONNE_HA 1998 0.009202703
## 10      AUS    RICE TONNE_HA 1999 0.008274809
```

To convert the values from "THND_HA" (thousand per hectare) to "THND_TONNE" (thousand tonnes per hectare), we need to multiply the values by a conversion factor. The conversion factor is based on the assumption that 1 thousand tonnes is equal to 1 million kilograms.

- Step 1: Multiply the values in the "THND_HA" column by 1000 to convert them to kilograms per hectare.
- Step 2: Divide the values in kilograms per hectare by 1,000,000 to convert them to tonnes per hectare.

```r
head(subset(crop_production, MEASURE == "THND_HA"), 10)
```

```
##      LOCATION SUBJECT MEASURE TIME Value
## 3317      IRN   WHEAT THND_HA 1990  6278
## 3318      IRN   WHEAT THND_HA 1991  6558
## 3319      IRN   WHEAT THND_HA 1992  6930
## 3320      IRN   WHEAT THND_HA 1993  7190
## 3321      IRN   WHEAT THND_HA 1994  6782
## 3322      IRN   WHEAT THND_HA 1995  6567
## 3323      IRN   WHEAT THND_HA 1996  6328
## 3324      IRN   WHEAT THND_HA 1997  6299
## 3325      IRN   WHEAT THND_HA 1998  6180
## 3326      IRN   WHEAT THND_HA 1999  4739
```

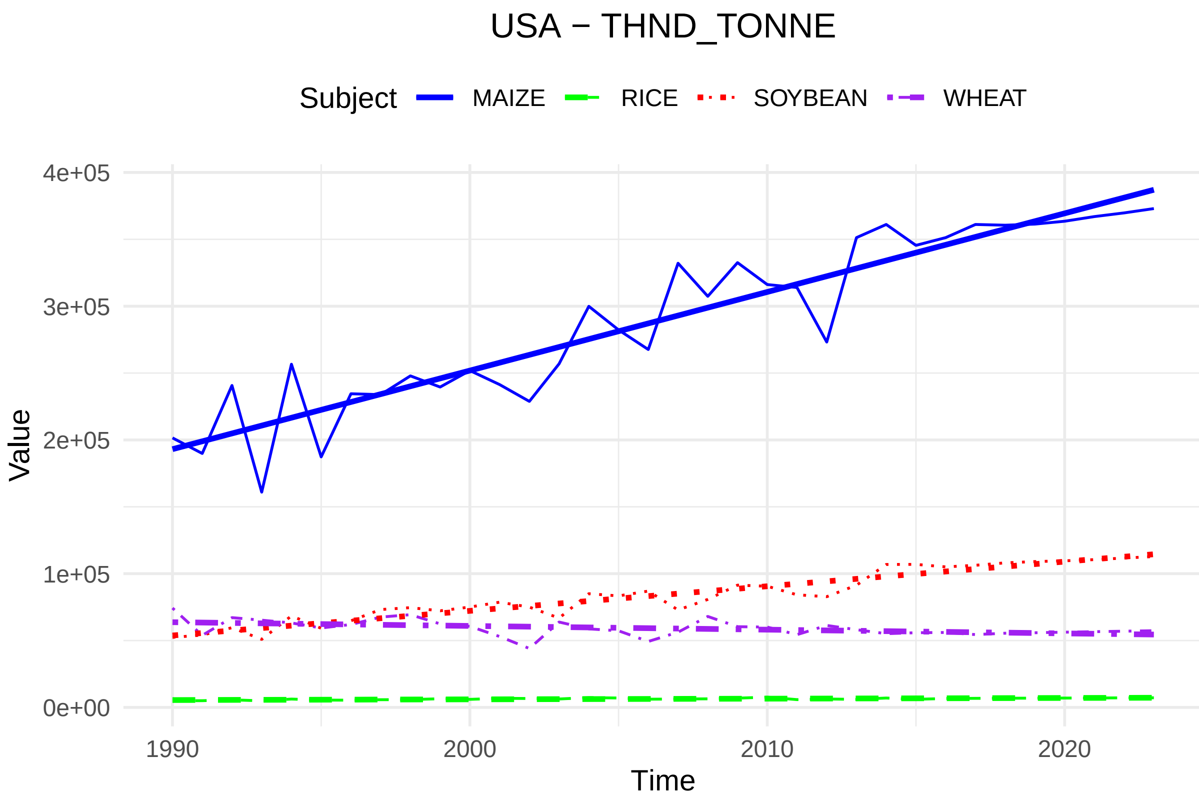
```r
crop_production$Value[crop_production$MEASURE == "THND_HA"] <-
↪   crop_production$Value[crop_production$MEASURE ==
    "THND_HA"] * 1000/1e+06

head(subset(crop_production, MEASURE == "THND_HA"), 10)
```

```
##      LOCATION SUBJECT MEASURE TIME Value
## 3317      IRN   WHEAT THND_HA 1990 6.278
## 3318      IRN   WHEAT THND_HA 1991 6.558
## 3319      IRN   WHEAT THND_HA 1992 6.930
## 3320      IRN   WHEAT THND_HA 1993 7.190
## 3321      IRN   WHEAT THND_HA 1994 6.782
## 3322      IRN   WHEAT THND_HA 1995 6.567
## 3323      IRN   WHEAT THND_HA 1996 6.328
## 3324      IRN   WHEAT THND_HA 1997 6.299
## 3325      IRN   WHEAT THND_HA 1998 6.180
## 3326      IRN   WHEAT THND_HA 1999 4.739
```

Now all the measure values are converted to THND_TONNE, rendering the "Measure" column useless, which can be removed.

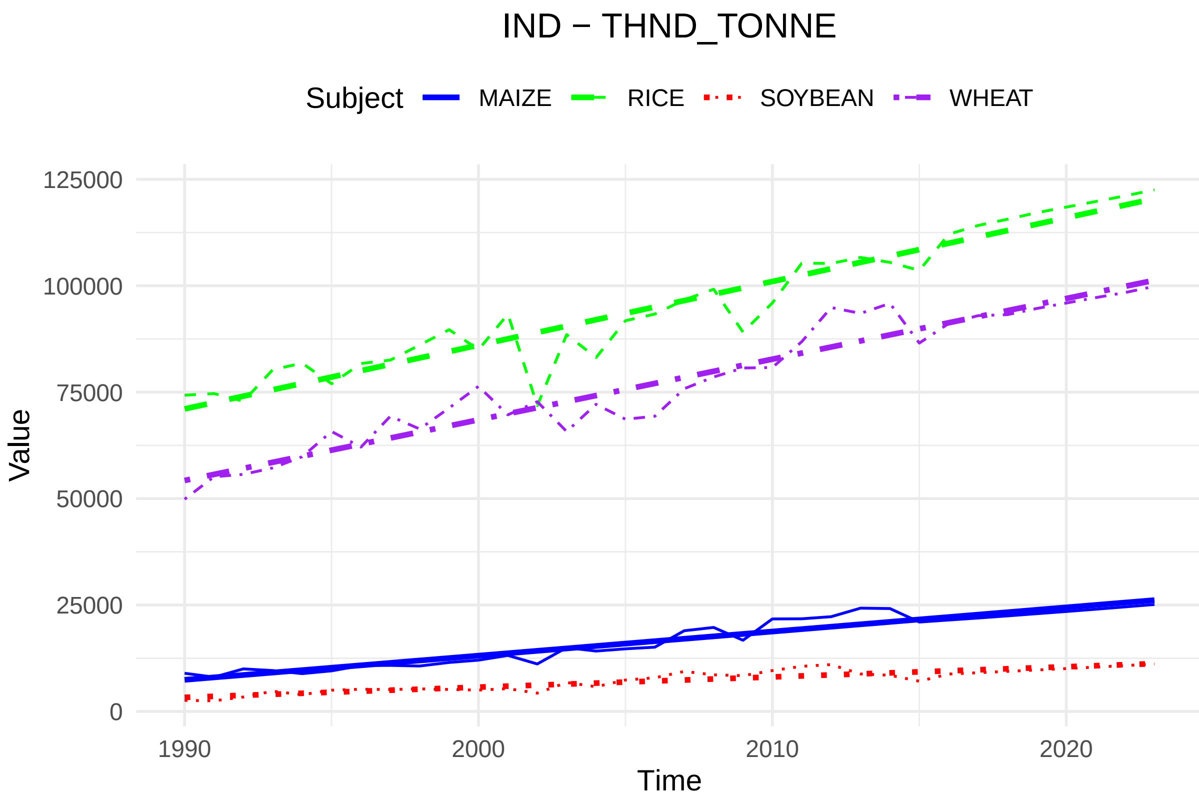
```r
# Filter the dataframe based on the selected location and measure
df_select <- subset(crop_production, LOCATION == "USA" & MEASURE == "THND_TONNE")

# Plot all 4 subjects
ggplot(df_select, aes(x = TIME, y = Value, color = SUBJECT, linetype = SUBJECT)) +
    geom_line() + geom_smooth(method = "lm", se = FALSE) + labs(title = paste("USA",
    "-", "THND_TONNE")) + theme_minimal() + theme(legend.position = "top") + guides(color
    ↪   = guide_legend(title = "Subject"),
    linetype = guide_legend(title = "Subject")) + scale_color_manual(values = c("blue",
    "green", "red", "purple")) + scale_linetype_manual(values = c("solid", "dashed",
    "dotted", "dotdash")) + xlab("Time") + ylab("Value") + theme(plot.title =
    ↪   element_text(hjust = 0.5)) +
    theme(plot.margin = margin(10, 10, 20, 10))
```

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



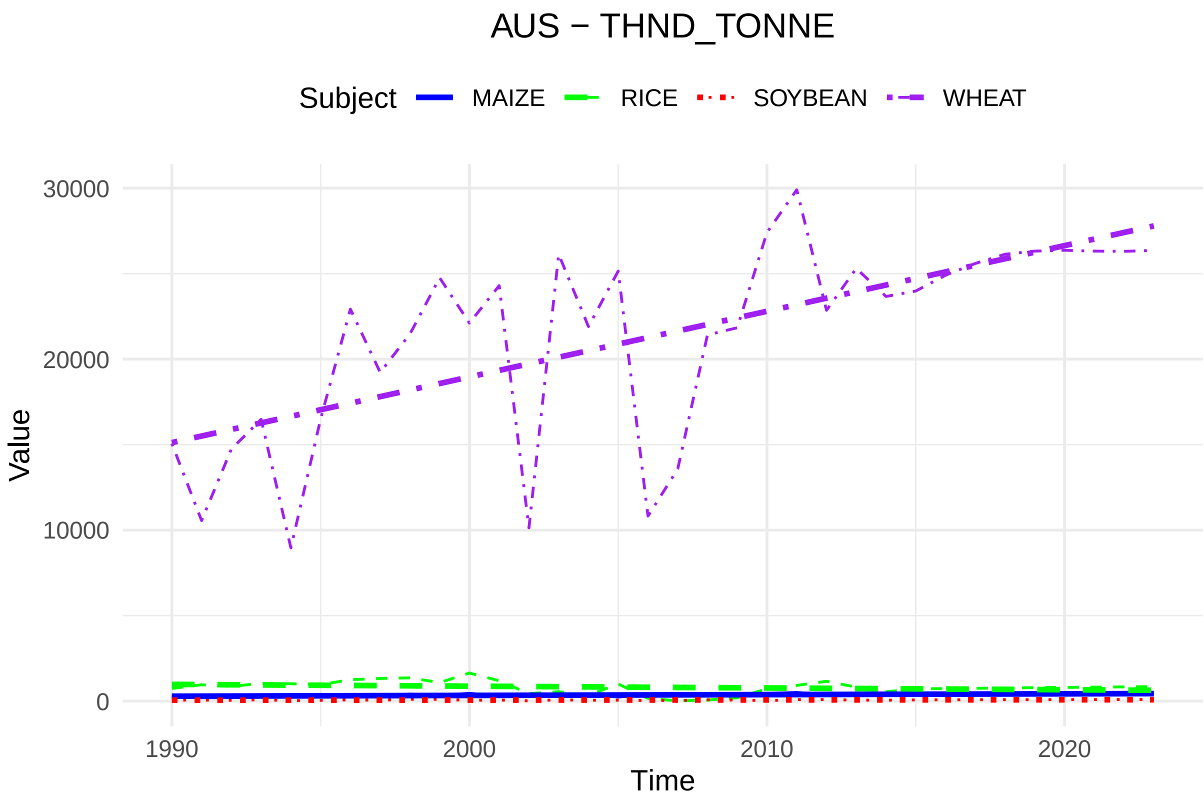

```r
# Filter the dataframe based on the selected location and measure
df_select <- subset(crop_production, LOCATION == "IND" & MEASURE == "THND_TONNE")

# Plot all 4 subjects
ggplot(df_select, aes(x = TIME, y = Value, color = SUBJECT, linetype = SUBJECT)) +
    geom_line() + geom_smooth(method = "lm", se = FALSE) + labs(title = paste("IND",
    "-", "THND_TONNE")) + theme_minimal() + theme(legend.position = "top") + guides(color
    ↪  = guide_legend(title = "Subject"),
    linetype = guide_legend(title = "Subject")) + scale_color_manual(values = c("blue",
    "green", "red", "purple")) + scale_linetype_manual(values = c("solid", "dashed",
    "dotted", "dotdash")) + xlab("Time") + ylab("Value") + theme(plot.title =
    ↪  element_text(hjust = 0.5)) +
    theme(plot.margin = margin(10, 10, 20, 10))
```

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




```r
# Filter the dataframe based on the selected location and measure
df_select <- subset(crop_production, LOCATION == "AUS" & MEASURE == "THND_TONNE")

# Plot all 4 subjects
ggplot(df_select, aes(x = TIME, y = Value, color = SUBJECT, linetype = SUBJECT)) +
    geom_line() + geom_smooth(method = "lm", se = FALSE) + labs(title = paste("AUS",
    "-", "THND_TONNE")) + theme_minimal() + theme(legend.position = "top") + guides(color
    ↪  = guide_legend(title = "Subject"),
    linetype = guide_legend(title = "Subject")) + scale_color_manual(values = c("blue",
    "green", "red", "purple")) + scale_linetype_manual(values = c("solid", "dashed",
    "dotted", "dotdash")) + xlab("Time") + ylab("Value") + theme(plot.title =
    ↪  element_text(hjust = 0.5)) +
    theme(plot.margin = margin(10, 10, 20, 10))
```

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



- **Inference:**
  - From the above plots, we can observe that the crop yields exhibit a positive trend over the years, indicating a likely response to the increasing demand for food to support the growing population.

Crop yield data standardization from country level to global level

```r
head(crop_production, 10)
```

```
##    LOCATION SUBJECT  MEASURE TIME       Value
## 1       AUS    RICE TONNE_HA 1990 0.008314607
## 2       AUS    RICE TONNE_HA 1991 0.008394737
## 3       AUS    RICE TONNE_HA 1992 0.008094340
## 4       AUS    RICE TONNE_HA 1993 0.008336000
## 5       AUS    RICE TONNE_HA 1994 0.008537815
## 6       AUS    RICE TONNE_HA 1995 0.007051095
## 7       AUS    RICE TONNE_HA 1996 0.008256579
## 8       AUS    RICE TONNE_HA 1997 0.009006803
## 9       AUS    RICE TONNE_HA 1998 0.009202703
## 10      AUS    RICE TONNE_HA 1999 0.008274809
```

```r
# Aggregating the data to just have Time (year) and Value columns
crop_production_aggregated_data <- aggregate(Value ~ TIME, data = crop_production,
    FUN = mean)
```

```r
head(crop_production_aggregated_data)
```

```
##   TIME Value
## 1 1970 1e-04
## 2 1971 1e-04
## 3 1972 1e-04
## 4 1973 1e-04
## 5 1974 1e-04
## 6 1975 1e-04
```

Step 6: Merging Crop Yield data with Temperature anomaly & Mean CO2 PPM

```r
## Temperature and CO2 - merged dataset
head(merged_data)
```

```
##   year mean_temperature reference_temperature_global      Anomaly mean_co2_ppm
## 1 1958         15.38208                     15.32852  0.053565217     315.3300
## 2 1959         15.34050                     15.32852  0.011981884     315.9817
## 3 1960         15.29192                     15.32852 -0.036601449     316.9083
## 4 1961         15.37992                     15.32852  0.051398551     317.6450
## 5 1962         15.32558                     15.32852 -0.002934783     318.4533
## 6 1963         15.37667                     15.32852  0.048148551     318.9925
```

```r
# Rename the column in crop_production_aggregated_data
colnames(crop_production_aggregated_data)[1] <- "year"

head(crop_production_aggregated_data)
```

```
##   year Value
## 1 1970 1e-04
## 2 1971 1e-04
## 3 1972 1e-04
## 4 1973 1e-04
## 5 1974 1e-04
## 6 1975 1e-04
```

```r
merged_data <- merge(crop_production_aggregated_data, merged_data, by = "year")


## Renaming columns - Value to mean_crop_yield; and Anomaly to
## Temperature_anomaly
merged_data <- merged_data %>%
    rename(mean_crop_yield = Value, Temperature_anomaly = Anomaly)

head(merged_data)
```

```
##   year mean_crop_yield mean_temperature reference_temperature_global
```

```
## 1 1970          1e-04         15.33267                      15.32852
## 2 1971          1e-04         15.20000                      15.32852
## 3 1972          1e-04         15.29292                      15.32852
## 4 1973          1e-04         15.40475                      15.32852
## 5 1974          1e-04         15.16808                      15.32852
## 6 1975          1e-04         15.23867                      15.32852
##   Temperature_anomaly mean_co2_ppm
## 1         0.004148551     325.6833
## 2        -0.128518116     326.3192
## 3        -0.035601449     327.4575
## 4         0.076231884     329.6775
## 5        -0.160434783     330.2442
## 6        -0.089851449     331.1525
```

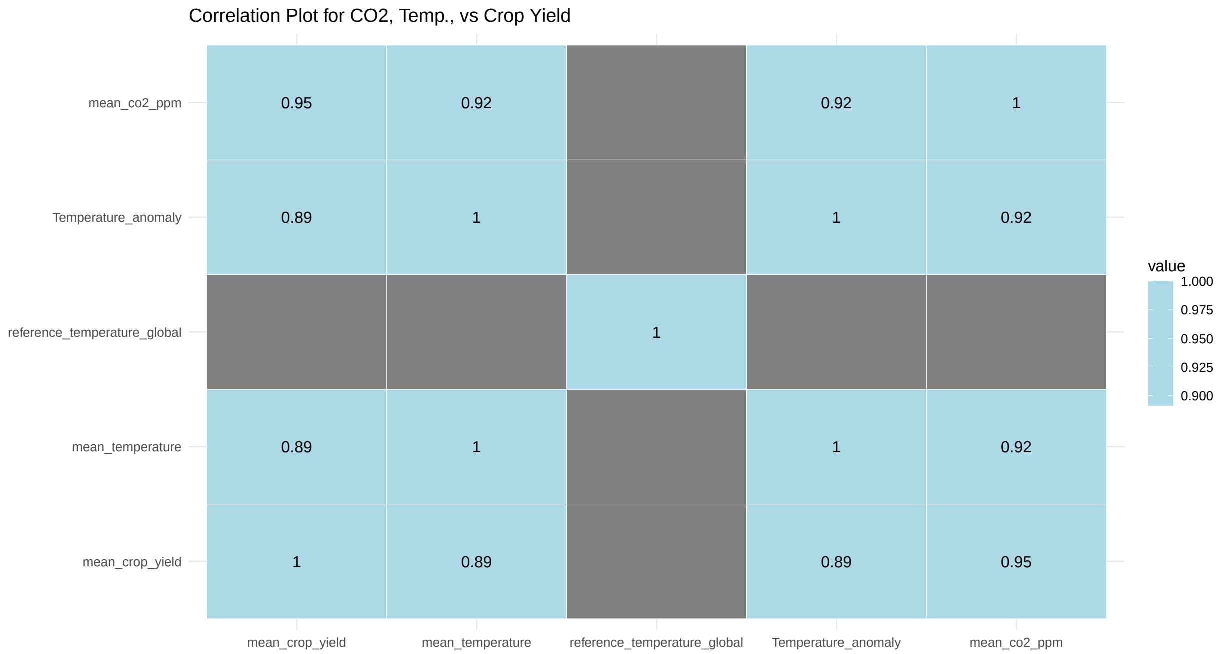
Correlation Plot for merged data:.

```r
# Calculate the correlation matrix
suppressWarnings(cor_matrix <- cor(merged_data[, c("mean_crop_yield", "mean_temperature",
    "reference_temperature_global", "Temperature_anomaly", "mean_co2_ppm")]))

# Convert correlation matrix to long format
cor_df_long <- reshape2::melt(cor_matrix)

print(cor_matrix)
```

```
##                              mean_crop_yield mean_temperature
## mean_crop_yield                    1.0000000        0.8913189
## mean_temperature                   0.8913189        1.0000000
## reference_temperature_global              NA               NA
## Temperature_anomaly                0.8913189        1.0000000
## mean_co2_ppm                       0.9472038        0.9232389
##                              reference_temperature_global Temperature_anomaly
## mean_crop_yield                                        NA           0.8913189
## mean_temperature                                       NA           1.0000000
## reference_temperature_global                            1                  NA
## Temperature_anomaly                                    NA           1.0000000
## mean_co2_ppm                                           NA           0.9232389
##                              mean_co2_ppm
## mean_crop_yield                 0.9472038
## mean_temperature                0.9232389
## reference_temperature_global           NA
## Temperature_anomaly             0.9232389
## mean_co2_ppm                    1.0000000
```

```r
# Create the correlation plot
suppressWarnings(ggplot(data = cor_df_long, aes(x = Var1, y = Var2)) + geom_tile(aes(fill
↪  = value),
    color = "white") + geom_text(aes(label = round(value, 2)), color = "black") +
    scale_fill_gradient(low = "lightblue", high = "lightblue") + theme_minimal() +
    labs(x = "", y = "", title = "Correlation Plot for CO2, Temp., vs Crop Yield"))
```

```
## Warning: Removed 8 rows containing missing values (geom_text).
```



- **Inference:**

Based on the above correlation matrix, the correlations between the variables are as follows:

- The correlation between "mean_crop_yield" and "mean_temperature" is 0.8913189.
- The correlation between "mean_crop_yield" and "Temperature_anomaly" is also 0.8913189.
- The correlation between "mean_crop_yield" and "mean_co2_ppm" is 0.9472038.

These correlations indicate a strong positive relationship between "mean_crop_yield" and both "mean_temperature" and "Temperature_anomaly". Additionally, there is a very strong positive correlation between "mean_crop_yield" and "mean_co2_ppm". The plot indicates a strong relationship between these variables, suggesting that they are closely related and exhibit a notable degree of interdependence.

Step 7: Model II: "mean_crop_yield ~ Temperature_anomaly + mean_co2_ppm"

The following code fits a linear regression model to predict the 'mean_crop_yield' variable based on the predictors 'Temperature_anomaly' and 'mean_co2_ppm'. The model is fitted using the 'merged_data' dataset.

```r
model <- lm(mean_crop_yield ~ Temperature_anomaly + mean_co2_ppm, data = merged_data)

summary(model)
```

```
##
## Call:
## lm(formula = mean_crop_yield ~ Temperature_anomaly + mean_co2_ppm,
##     data = merged_data)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max 
## -3392.4  -782.5     7.3  1110.3  2829.8 
## 
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)    
## (Intercept)          -59102.5     8902.7  -6.639 4.30e-08 ***
## Temperature_anomaly    2315.0     2561.0   0.904    0.371    
## mean_co2_ppm            176.3       26.4   6.679 3.76e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 1519 on 43 degrees of freedom
## Multiple R-squared:  0.8991, Adjusted R-squared:  0.8944 
## F-statistic: 191.6 on 2 and 43 DF,  p-value: < 2.2e-16
```

- **Inference:**

The linear regression model results for the equation mean_crop_yield ~ Temperature_anomaly + mean_co2_ppm are as follows:

- Intercept: The estimated intercept of the regression model is -59102.5. It represents the expected mean crop yield when both predictor variables (Temperature_anomaly and mean_co2_ppm) are zero.
- Temperature_anomaly: The estimated coefficient for Temperature_anomaly is 2315.0. It indicates that for each unit increase in the Temperature_anomaly, the mean crop yield is expected to increase by 2315.0 units, holding other variables constant. However, the p-value (0.371) suggests that this relationship is not statistically significant at a conventional significance level of 0.05.
- mean_co2_ppm: The estimated coefficient for mean_co2_ppm is 176.3. It indicates that for each unit increase in mean_co2_ppm, the mean crop yield is expected to increase by 176.3 units, holding other variables constant. The p-value (< 0.0000000376) suggests that this relationship is statistically significant.
- The R-squared value of 0.8991 indicates that 89.91% of the variation in mean crop yield can be explained by the predictor variables Temperature_anomaly and mean_co2_ppm.
- The adjusted R-squared value of 0.8944 adjusts the R-squared value for the number of predictor variables and sample size.
- The F-statistic of 191.6 with a very low p-value (< 0.00000000000000022) indicates that the overall regression model is statistically significant.
- The residual standard error of 1519 represents the estimate of the standard deviation of the residuals, which measures the average distance between the observed mean crop yield values and the predicted values from the regression model.
- The model indicates that both "Temperature_anomaly" and "mean_co2_ppm" have a significant effect on predicting "mean_crop_yield".

To train, test, and predict using the linear regression model (Model II), we can follow these steps:

70% of the data is randomly sampled and assigned to the "train_data" dataframe, while the remaining 30% is assigned to the "test_data" dataframe.

```r
# Step 1: Split the data into training and testing sets:
set.seed(123)  # For reproducibility
train_indices <- sample(1:nrow(merged_data), 0.7 * nrow(merged_data))
train_data <- merged_data[train_indices, ]
test_data <- merged_data[-train_indices, ]
```

```r
# Step 2: Train the linear regression model using the training data:
model <- lm(mean_crop_yield ~ Temperature_anomaly + mean_co2_ppm, data = train_data)

# Step 3: Predict the mean crop yield using the test data:
predictions <- predict(model, newdata = test_data)

# Step 4: Compare the predicted values with the actual values in the test data:
comparison <- data.frame(Actual = test_data$mean_crop_yield, Predicted = predictions)

head(comparison, 10)
```

```
##       Actual Predicted
## 1     0.0001 -1793.069
## 2     0.0001 -2099.195
## 6     0.0001 -1137.600
## 16    0.0001  1746.962
## 18    0.0001  2850.709
## 21 3418.9500  4093.113
## 22 3416.3773  4239.078
## 23 3608.6541  3919.568
## 24 3536.0512  4077.588
## 30 8435.2198  6457.762
```

```r
# Step 5: Evaluate the model performance using appropriate metrics (e.g., mean
# squared error, root mean squared error, mean absolute error):
mse <- mean((test_data$mean_crop_yield - predictions)^2)
rmse <- sqrt(mse)
mae <- mean(abs(test_data$mean_crop_yield - predictions))

cat("MSE:", mse, "\n")
```

```
## MSE: 2106268
```

```r
cat("RMSE:", rmse, "\n")
```

```
## RMSE: 1451.299
```

```r
cat("MAE:", mae, "\n")
```

```
## MAE: 1229.882
```

- **Inference:**

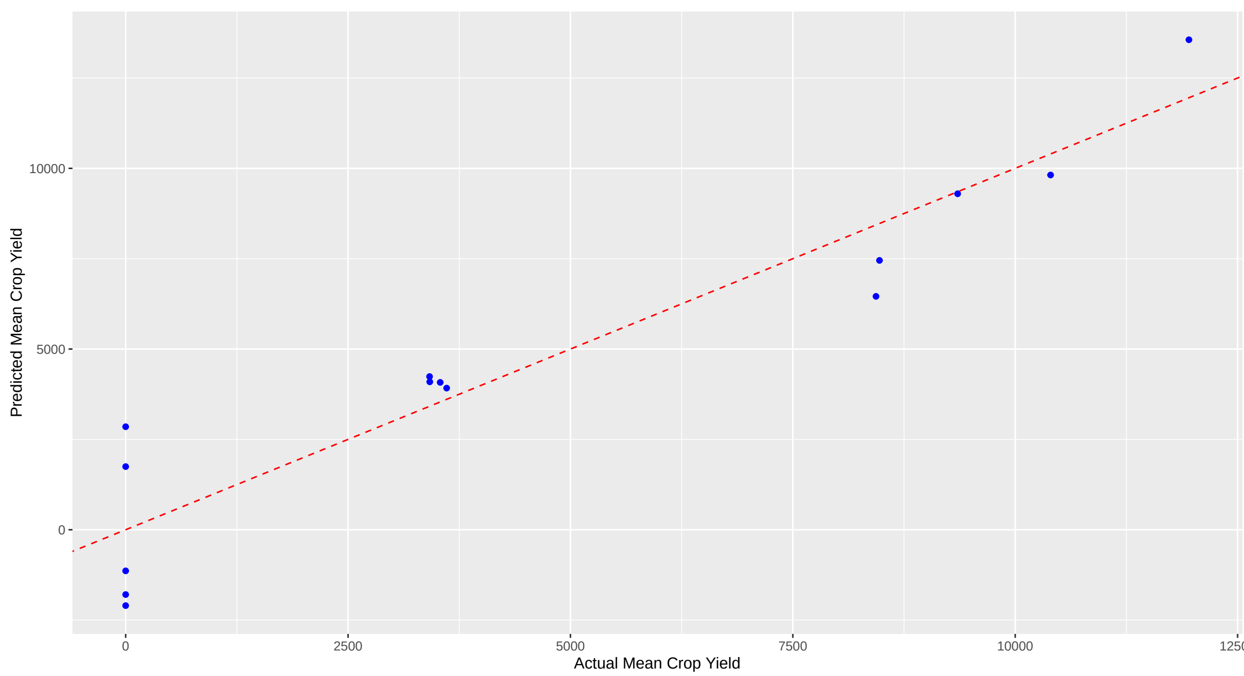
To evaluate the model performance, we split the data into training and testing sets. The model was trained using the training data and then used to predict the mean crop yield for the test data. We calculated the Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE) as evaluation metrics. The performance metrics for the model are as follows:

- MSE: 2106268
- RMSE: 1451.299
- MAE: 1229.882

In general, lower values of MSE, RMSE, and MAE indicate better model performance. However, it is important to consider the scale and nature of the data to determine whether the obtained values are acceptable. In this case, the MSE of 2106268, RMSE of 1451.299, and MAE of 1229.882 suggest that there is some level of error in the predictions of the mean crop yield. It is recommended to compare these values with the range and variability of the crop yield data to get a better understanding of the model's performance.

To plot the linear regression model (Model II):

```
# Make predictions on the test data using the model
predictions <- predict(model, newdata = test_data)

# Create a new data frame combining the observed and predicted values
plot_data <- data.frame(Actual = test_data$mean_crop_yield, Predicted = predictions)

# Plot the observed vs predicted values
plot <- ggplot(plot_data, aes(x = Actual, y = Predicted)) + geom_point(color = "blue") +
    geom_abline(slope = 1, intercept = 0, color = "red", linetype = "dashed") + labs(x =
    ↪  "Actual Mean Crop Yield",
    y = "Predicted Mean Crop Yield")

# Display the plot
plot
```



- **Inference:**
  - The plot demonstrates a reasonably accurate alignment of the actual and predicted points with the regression line. However, there is potential for improvement, as indicated by the MSE, RMSE, and MAE scores obtained during model training and testing.

– To further enhance the model's performance, we will pursue Hyperparameter optimization techniques.

Hyper-parameter optimization for model selection for Model II: (mean_crop_yield ~ Temperature_anomaly + mean_co2_ppm, data = merged_data)

To perform hyperparameter optimization for model selection in linear regression, we usually focus on tuning the hyperparameters of the regression algorithm itself, rather than the specific variables in the model. In the case of above linear regression, there aren't many hyperparameters to tune, as the model is relatively simple.

However to explore hyperparameter optimization for model selection in a broader sense, we could consider using techniques such as regularization methods like Ridge regression or Lasso regression. These techniques introduce additional hyperparameters, such as the regularization strength (lambda) in Ridge regression or the regularization parameter (alpha) in Lasso regression. By tuning these hyperparameters, we could control the level of regularization and potentially improve the model's performance.

why perform hyperparameter optimization using Ridge regression (CV)?

- Performing hyperparameter optimization using Ridge regression with cross-validation (CV) is a common practice to find the best hyperparameters for the model.
- Ridge regression is a linear regression technique that adds a penalty term to the ordinary least squares (OLS) cost function, which helps to regularize the model.
- The regularization term controls the complexity of the model and prevents overfitting, especially when dealing with multicollinearity in the feature variables.

```r
## Ridge regression as an alternative to ordinary linear regression: Run the
## below if you get any error when running glmnet() update.packages(c('Matrix',
## 'survival', 'recipes', 'textplot', 'mgcv'))

# Prepare the data
X <- model.matrix(mean_crop_yield ~ Temperature_anomaly + mean_co2_ppm, data =
↪  merged_data)
y <- merged_data$mean_crop_yield

# Perform hyperparameter optimization using cross-validation
ridge_model <- cv.glmnet(X, y, alpha = 0, lambda = seq(0.1, 1, by = 0.1))

# Select the best lambda value
best_lambda <- ridge_model$lambda.min

print(best_lambda)
```

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

```r
# Fit the final Ridge regression model with the selected lambda
final_model <- glmnet(X, y, alpha = 0, lambda = best_lambda)

# Summary of the final model
summary(final_model)
```

```
##           Length Class     Mode
## a0        1      -none-    numeric
```

```
## beta      3      dgCMatrix S4
## df        1      -none-    numeric
## dim       2      -none-    numeric
## lambda    1      -none-    numeric
## dev.ratio 1      -none-    numeric
## nulldev   1      -none-    numeric
## npasses   1      -none-    numeric
## jerr      1      -none-    numeric
## offset    1      -none-    logical
## call      5      -none-    call
## nobs      1      -none-    numeric
```

```r
# Each coefficient indicates how much the target variable is expected to change
# when the corresponding predictor variable increases by one unit, assuming all
# other variables remain constant.
coefficients <- coef(final_model)

# In the context of statistical models, degrees of freedom refer to the number
# of values in the final calculation of a statistic that are free to vary.
degrees_of_freedom <- final_model$df

# The deviance is a measure of the difference between the model's predicted
# values and the observed data.
deviance_ratio <- final_model$dev.ratio
```

```r
print(coefficients)
```

```
## 4 x 1 sparse Matrix of class "dgCMatrix"
##                              s0
## (Intercept)          -58929.547
## (Intercept)                   .
## Temperature_anomaly    2365.253
## mean_co2_ppm            175.790
```

```r
print(degrees_of_freedom)
```

```
## [1] 2
```

```r
print(deviance_ratio)
```

```
## [1] 0.8991113
```

- **Inference:**
  - After performing hyperparameter optimization using ridge regression, the selected best lambda value is 1. This lambda value represents the regularization parameter that controls the amount of shrinkage applied to the coefficients.
  - The final Ridge regression model is fitted using the selected lambda value. The model summary provides information about the different components of the model, such as the intercept, coefficients, degrees of freedom, and deviance ratio.

- Coefficients:
  - The intercept term (Intercept) has a coefficient of -58929.547.
  - The Temperature_anomaly variable has a coefficient of 2365.253.
  - The mean_co2_ppm variable has a coefficient of 175.790.
- The degrees of freedom for the final model are 2, indicating that there are 2 predictors (temperature anomaly and mean CO2 ppm) in the model.
- The deviance ratio of 0.8991113 provides a measure of the model's goodness of fit, indicating that approximately 89.91% of the variance in mean crop yield is explained by the model. This suggests that the model is performing reasonably well in capturing the relationship between the predictors and the response variable.

These results suggest that the final Ridge regression model with the selected lambda value can be used to predict mean crop yield and provides valuable insights into the relationship between the predictors and the target variable. Hence we could stop further Hyper parameter optimization for this model (Model II).

Post model-testing Clean-ups -

```r
head(merged_data)
```

```
##   year mean_crop_yield mean_temperature reference_temperature_global
## 1 1970           1e-04         15.33267                    15.32852
## 2 1971           1e-04         15.20000                    15.32852
## 3 1972           1e-04         15.29292                    15.32852
## 4 1973           1e-04         15.40475                    15.32852
## 5 1974           1e-04         15.16808                    15.32852
## 6 1975           1e-04         15.23867                    15.32852
##   Temperature_anomaly mean_co2_ppm
## 1         0.004148551     325.6833
## 2        -0.128518116     326.3192
## 3        -0.035601449     327.4575
## 4         0.076231884     329.6775
## 5        -0.160434783     330.2442
## 6        -0.089851449     331.1525
```

```r
# renaming the columns for better match
merged_data <- merged_data %>%
    rename(Year = year)

# Print the updated merged data
head(merged_data)
```

```
##   Year mean_crop_yield mean_temperature reference_temperature_global
## 1 1970           1e-04         15.33267                    15.32852
## 2 1971           1e-04         15.20000                    15.32852
## 3 1972           1e-04         15.29292                    15.32852
## 4 1973           1e-04         15.40475                    15.32852
## 5 1974           1e-04         15.16808                    15.32852
## 6 1975           1e-04         15.23867                    15.32852
##   Temperature_anomaly mean_co2_ppm
## 1         0.004148551     325.6833
## 2        -0.128518116     326.3192
## 3        -0.035601449     327.4575
```

```
## 4         0.076231884     329.6775
## 5        -0.160434783     330.2442
## 6        -0.089851449     331.1525
```

Step 8: Loading Climate change data (Precipitation), Data cleansing & EDA

Load the Precipitation data from the CSV file - climate_change_data.csv

The Climate change (Precipitation) data is loaded from "climate_change_data.csv" (Source: Kaggle). The dataset contains information related to climate change, including various weather-related measurements for different locations and countries. The columns in the dataset are as follows:

- Date: The date and time of the weather measurement.
- Location: The name of the location where the weather measurement was taken.
- Country: The country to which the location belongs.
- Temperature: The recorded temperature at the location.
- CO2.Emissions: The level of CO2 emissions recorded at the location.
- Sea.Level.Rise: The recorded sea level rise at the location.
- Precipitation: The amount of precipitation (rainfall) recorded at the location.
- Humidity: The recorded humidity level at the location.
- Wind.Speed: The recorded wind speed at the location.

```r
climate_change_data <- read.csv("climate_change_data.csv")

head(climate_change_data, 10)
```

```
##                              Date         Location       Country Temperature
## 1  2000-01-01 00:00:00.000000000  New Williamtown        Latvia   10.688986
## 2  2000-01-01 20:09:43.258325832     North Rachel  South Africa   13.814430
## 3  2000-01-02 16:19:26.516651665 West Williamland French Guiana   27.323718
## 4  2000-01-03 12:29:09.774977497      South David       Vietnam   12.309581
## 5  2000-01-04 08:38:53.033303330   New Scottburgh       Moldova   13.210885
## 6  2000-01-05 04:48:36.291629162     South Nathan  Saint Helena    6.229326
## 7  2000-01-06 00:58:19.549954995 Port Richardfurt        Tuvalu   21.646738
## 8  2000-01-06 21:08:02.808280828         Adambury     Australia   19.730800
## 9  2000-01-07 17:17:46.066606660   Williamsonberg         Qatar   19.858114
## 10 2000-01-08 13:27:29.324932493     North Thomas          Chad   14.121563
##    CO2.Emissions Sea.Level.Rise Precipitation Humidity Wind.Speed
## 1       403.1189    0.717506028     13.835237 23.63126  18.492026
## 2       396.6635    1.205714578     40.974084 43.98295  34.249300
## 3       451.5532   -0.160782970     42.697931 96.65260  34.124261
## 4       422.4050   -0.475931471      5.193341 47.46794   8.554563
## 5       410.4730    1.135756628     78.695280 61.78967   8.001164
## 6       392.4733    1.122209652     76.368331 48.97389  30.398908
## 7       387.6484    0.058471241      9.650389 11.40228  15.720944
## 8       448.1803    0.001415079     93.360755 21.52635  29.993495
## 9       379.6188    0.584880621      6.218846 30.86195  37.519472
## 10      410.5171   -1.712224247     15.351583 88.42279  47.922521
```

```r
dim(climate_change_data)
```

```
## [1] 10000     9
```

```r
summary(climate_change_data)
```

```
##      Date             Location           Country           Temperature
##  Length:10000       Length:10000       Length:10000       Min.   :-3.804
##  Class :character   Class :character   Class :character   1st Qu.:11.578
##  Mode  :character   Mode  :character   Mode  :character   Median :14.981
##                                                           Mean   :14.936
##                                                           3rd Qu.:18.306
##                                                           Max.   :33.977
##  CO2.Emissions   Sea.Level.Rise      Precipitation        Humidity
##  Min.   :182.1   Min.   :-4.092155   Min.   : 0.01014   Min.   : 0.019
##  1st Qu.:367.1   1st Qu.:-0.673809   1st Qu.:24.49752   1st Qu.:24.713
##  Median :400.8   Median : 0.002332   Median :49.81897   Median :49.678
##  Mean   :400.2   Mean   :-0.003152   Mean   :49.88121   Mean   :49.771
##  3rd Qu.:433.3   3rd Qu.: 0.675723   3rd Qu.:74.52499   3rd Qu.:75.206
##  Max.   :582.9   Max.   : 4.116559   Max.   :99.99190   Max.   :99.960
##    Wind.Speed
##  Min.   : 0.00173
##  1st Qu.:12.53973
##  Median :24.91079
##  Mean   :25.08207
##  3rd Qu.:37.67026
##  Max.   :49.99766
```

The climate_change_data dataset consists of 10,000 rows and 9 columns.

Data cleansing and Feature Engineering

Convert data types:

Ensure that each column has the correct data type. Dates should be converted to the Date data type, and numeric values should be converted to the appropriate numeric types.

```r
climate_change_data$Date <- as.Date(climate_change_data$Date)
climate_change_data$Temperature <- as.numeric(climate_change_data$Temperature)

summary(climate_change_data)
```

```
##       Date              Location           Country           Temperature
##  Min.   :2000-01-01   Length:10000       Length:10000       Min.   :-3.804
##  1st Qu.:2005-09-30   Class :character   Class :character   1st Qu.:11.578
##  Median :2011-07-01   Mode  :character   Mode  :character   Median :14.981
##  Mean   :2011-07-01                                         Mean   :14.936
##  3rd Qu.:2017-03-31                                         3rd Qu.:18.306
##  Max.   :2022-12-31                                         Max.   :33.977
##  CO2.Emissions   Sea.Level.Rise      Precipitation        Humidity
##  Min.   :182.1   Min.   :-4.092155   Min.   : 0.01014   Min.   : 0.019
##  1st Qu.:367.1   1st Qu.:-0.673809   1st Qu.:24.49752   1st Qu.:24.713
##  Median :400.8   Median : 0.002332   Median :49.81897   Median :49.678
##  Mean   :400.2   Mean   :-0.003152   Mean   :49.88121   Mean   :49.771
##  3rd Qu.:433.3   3rd Qu.: 0.675723   3rd Qu.:74.52499   3rd Qu.:75.206
##  Max.   :582.9   Max.   : 4.116559   Max.   :99.99190   Max.   :99.960
##    Wind.Speed
```

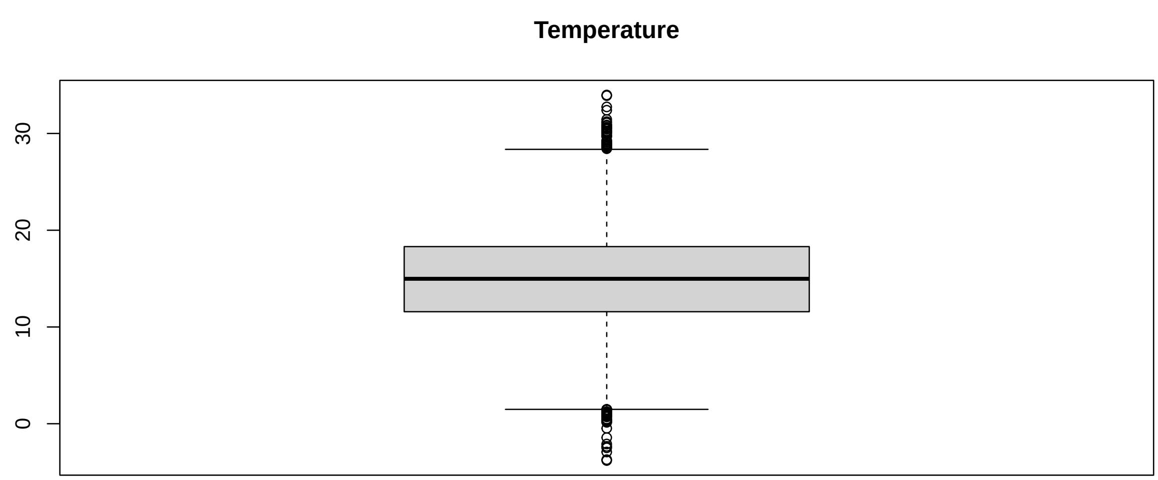
```
##   Min.   : 0.00173
##   1st Qu.:12.53973
##   Median :24.91079
##   Mean   :25.08207
##   3rd Qu.:37.67026
##   Max.   :49.99766
```

```r
# Remove duplicates: Check for and remove any duplicate rows in the dataset.
climate_change_data <- unique(climate_change_data)

dim(climate_change_data)
```

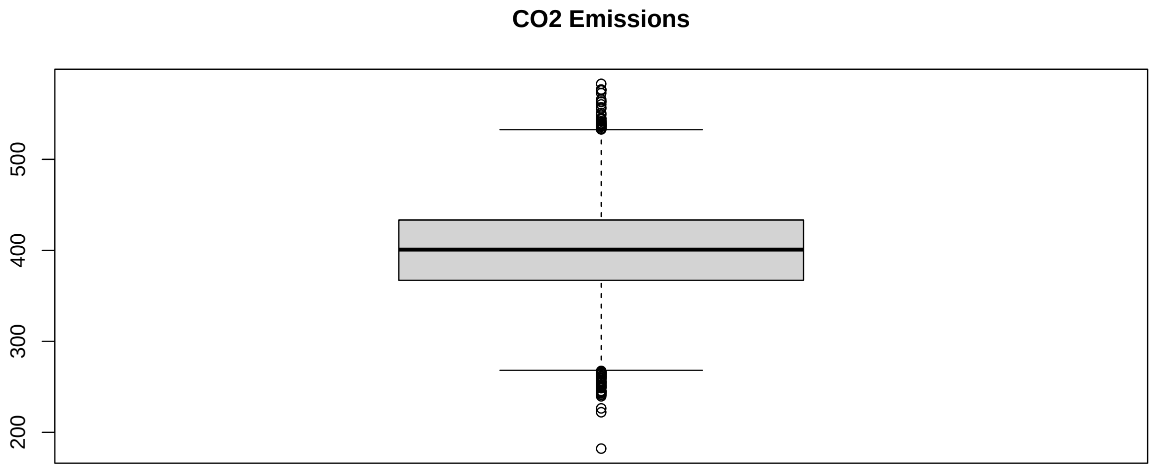
```
## [1] 10000     9
```

Check for outliers in climate_change_data -

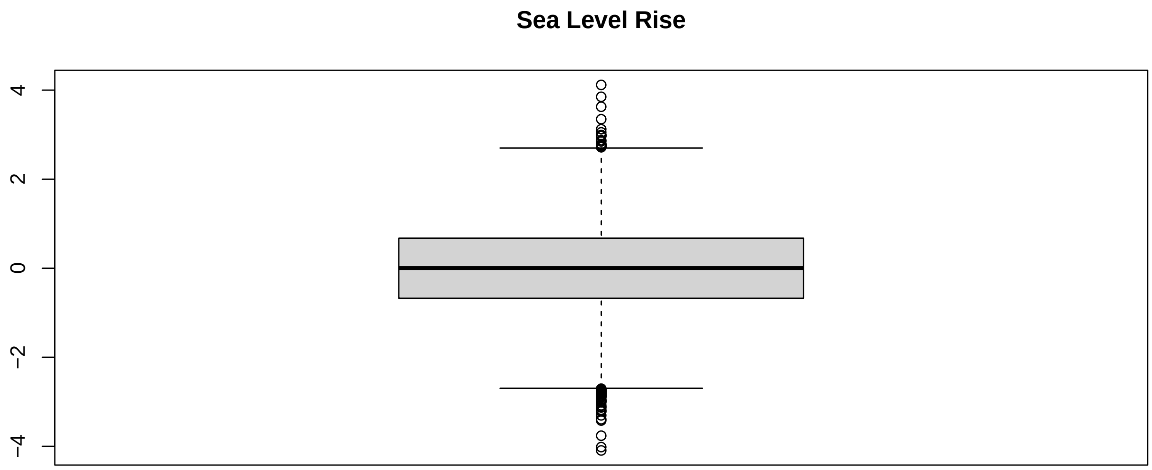
```r
# Create the boxplot for each variable
boxplot(climate_change_data$Temperature, main = "Temperature")
```



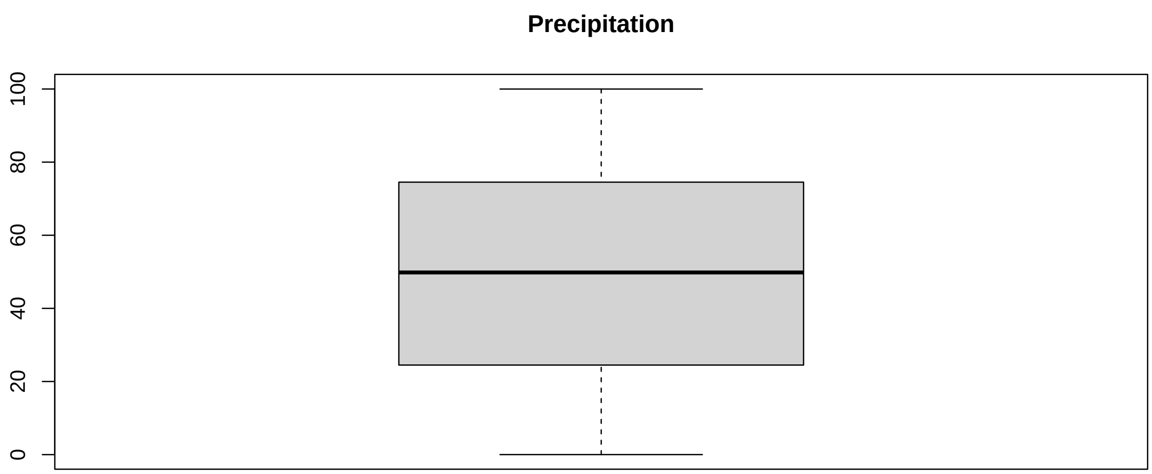

```r
boxplot(climate_change_data$CO2.Emissions, main = "CO2 Emissions")
```



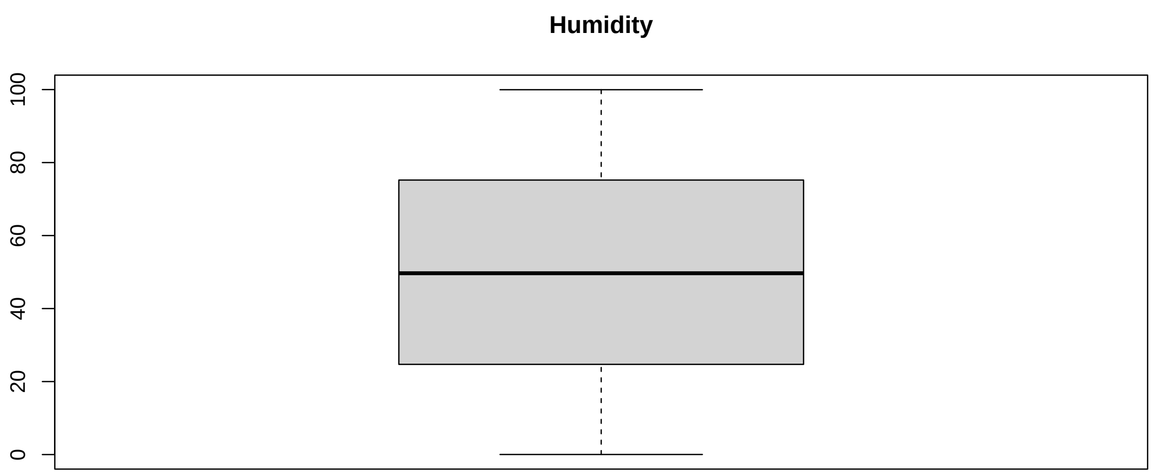

```r
boxplot(climate_change_data$Sea.Level.Rise, main = "Sea Level Rise")
```



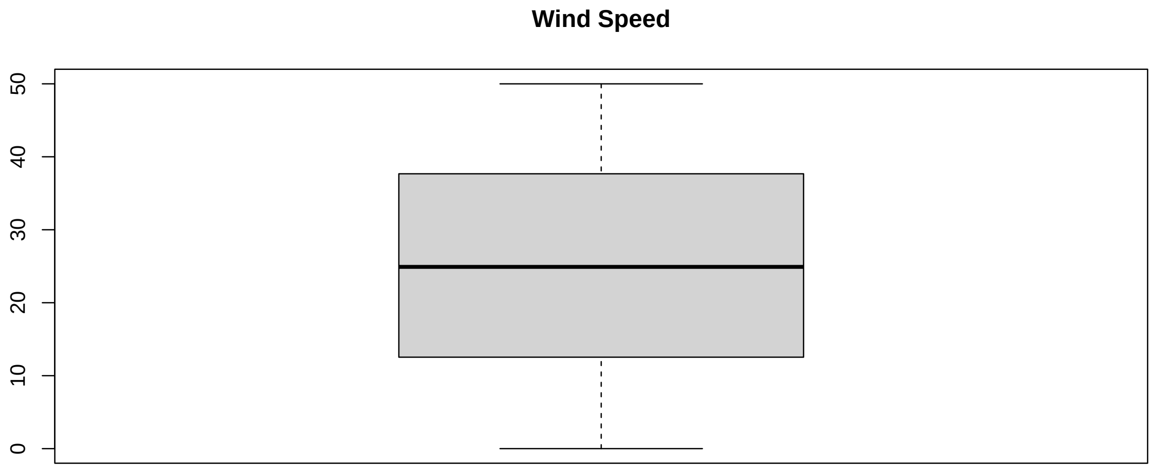

```r
boxplot(climate_change_data$Precipitation, main = "Precipitation")
```




```r
boxplot(climate_change_data$Humidity, main = "Humidity")
```




```r
boxplot(climate_change_data$Wind.Speed, main = "Wind Speed")
```



- **Inference:**
  - No outliers were detected in the climate_change_data$Precipitation variable, which is the specific variable of interest in our analysis.

```
# Extract the year from the Date column
climate_change_data$Year <- year(climate_change_data$Date)

head(climate_change_data, 10)
```

```
##          Date          Location       Country Temperature CO2.Emissions
## 1  2000-01-01  New Williamtown        Latvia   10.688986      403.1189
## 2  2000-01-01     North Rachel  South Africa   13.814430      396.6635
## 3  2000-01-02 West Williamland French Guiana   27.323718      451.5532
## 4  2000-01-03      South David       Vietnam   12.309581      422.4050
## 5  2000-01-04   New Scottburgh       Moldova   13.210885      410.4730
## 6  2000-01-05     South Nathan  Saint Helena    6.229326      392.4733
## 7  2000-01-06 Port Richardfurt        Tuvalu   21.646738      387.6484
## 8  2000-01-06         Adambury     Australia   19.730800      448.1803
## 9  2000-01-07   Williamsonberg         Qatar   19.858114      379.6188
## 10 2000-01-08     North Thomas          Chad   14.121563      410.5171
##    Sea.Level.Rise Precipitation Humidity Wind.Speed Year
## 1     0.717506028     13.835237 23.63126  18.492026 2000
## 2     1.205714578     40.974084 43.98295  34.249300 2000
## 3    -0.160782970     42.697931 96.65260  34.124261 2000
## 4    -0.475931471      5.193341 47.46794   8.554563 2000
## 5     1.135756628     78.695280 61.78967   8.001164 2000
## 6     1.122209652     76.368331 48.97389  30.398908 2000
## 7     0.058471241      9.650389 11.40228  15.720944 2000
## 8     0.001415079     93.360755 21.52635  29.993495 2000
## 9     0.584880621      6.218846 30.86195  37.519472 2000
## 10   -1.712224247     15.351583 88.42279  47.922521 2000
```

Standardize the precipitation data:

Calculate the mean and standard deviation of precipitation as follows:

```
mean_precip <- mean(climate_change_data$Precipitation)
sd_precip <- sd(climate_change_data$Precipitation)

climate_change_data$Standardized_Precipitation <- (climate_change_data$Precipitation -
    mean_precip)/sd_precip

# Group the data by year and country and calculate the aggregate of the last
# column
aggregated_data <- climate_change_data %>%
    group_by(Year) %>%
    summarize(Standardized_Precipitation = Standardized_Precipitation)
```

```
## `summarise()` has grouped output by 'Year'. You can override using the
## `.groups` argument.
```

```
head(aggregated_data, 10)
```

```
## # A tibble: 10 x 2
## # Groups:   Year [1]
##     Year Standardized_Precipitation
##    <dbl>                      <dbl>
##  1  2000                     -1.25
##  2  2000                     -0.309
##  3  2000                     -0.249
##  4  2000                     -1.55
##  5  2000                      0.998
##  6  2000                      0.918
##  7  2000                     -1.39
##  8  2000                      1.51
##  9  2000                     -1.51
## 10  2000                     -1.20
```

Group the data by year and aggregate the standardized precipitation -

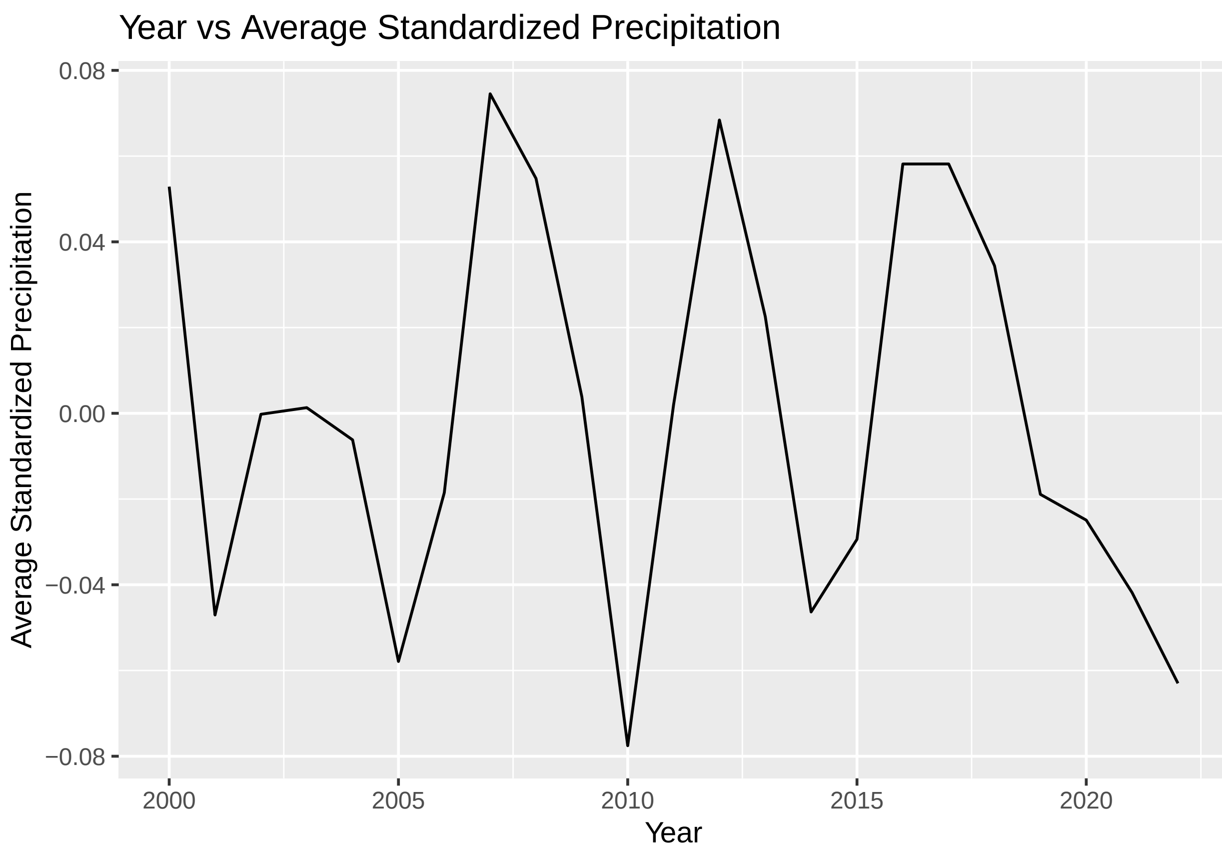
```
grouped_data <- aggregated_data %>%
    group_by(Year) %>%
    summarise(Avg_Standardized_Precipitation = mean(Standardized_Precipitation))

head(grouped_data)
```

```
## # A tibble: 6 x 2
##    Year Avg_Standardized_Precipitation
##   <dbl>                          <dbl>
## 1  2000                       0.0529
## 2  2001                      -0.0470
## 3  2002                      -0.000222
## 4  2003                       0.00132
## 5  2004                      -0.00620
## 6  2005                      -0.0579
```

```r
summary(grouped_data)
```

```
##       Year      Avg_Standardized_Precipitation
##  Min.   :2000   Min.   :-0.0775550
##  1st Qu.:2006   1st Qu.:-0.0356137
##  Median :2011   Median :-0.0002222
##  Mean   :2011   Mean   :-0.0000340
##  3rd Qu.:2016   3rd Qu.: 0.0436465
##  Max.   :2022   Max.   : 0.0745403
```

```r
#### Plot year vs average standardized precipitation
ggplot(grouped_data, aes(x = Year, y = Avg_Standardized_Precipitation)) + geom_line() +
    labs(title = "Year vs Average Standardized Precipitation", x = "Year", y = "Average
    ↪  Standardized Precipitation")
```



- **Inference:**
  - The observed variations in average standardized precipitation may be attributed to the seasonal patterns present in the data.
  - When analyzing average standardized precipitation data, which is a way to normalize or standardize the precipitation values across different locations or time periods, one may observe variations or differences in the values.

Step 9: Merging Precipitation data with Temperature, CO2, and crop yield data

Merging the merged_data df with the grouped_data df, we have -

```
merged_data <- merge(merged_data, grouped_data, by = "Year")

# Print the merged data
head(merged_data)
```

```
##   Year mean_crop_yield mean_temperature reference_temperature_global
## 1 2000        8336.612         15.61067                     15.32852
## 2 2001        8474.338         15.76750                     15.32852
## 3 2002        8241.606         15.82917                     15.32852
## 4 2003        8448.621         15.82658                     15.32852
## 5 2004        9339.853         15.75725                     15.32852
## 6 2005        9353.173         15.87925                     15.32852
##   Temperature_anomaly mean_co2_ppm Avg_Standardized_Precipitation
## 1           0.2821486     369.4750                    0.052878428
## 2           0.4389819     371.0208                   -0.047049657
## 3           0.5006486     373.0967                   -0.000222233
## 4           0.4980652     375.6367                    0.001317474
## 5           0.4287319     377.3625                   -0.006198974
## 6           0.5507319     379.6100                   -0.057899793
```

```
summary(merged_data)
```

```
##       Year      mean_crop_yield mean_temperature reference_temperature_global
##  Min.   :2000   Min.   : 8242   Min.   :15.61    Min.   :15.33
##  1st Qu.:2004   1st Qu.: 9033   1st Qu.:15.77    1st Qu.:15.33
##  Median :2008   Median :10036   Median :15.83    Median :15.33
##  Mean   :2008   Mean   : 9996   Mean   :15.82    Mean   :15.33
##  3rd Qu.:2011   3rd Qu.:10766   3rd Qu.:15.86    3rd Qu.:15.33
##  Max.   :2015   Max.   :12103   Max.   :16.06    Max.   :15.33
##  Temperature_anomaly  mean_co2_ppm   Avg_Standardized_Precipitation
##  Min.   :0.2821      Min.   :369.5   Min.   :-0.0775550
##  1st Qu.:0.4405      1st Qu.:376.9   1st Qu.:-0.0336305
##  Median :0.4984      Median :384.5   Median : 0.0005476
##  Mean   :0.4935      Mean   :384.7   Mean   :-0.0001789
##  3rd Qu.:0.5321      3rd Qu.:392.2   3rd Qu.: 0.0301466
##  Max.   :0.7301      Max.   :400.9   Max.   : 0.0745403
```

Step 10: Model III: "Avg_Standardized_Precipitation ~ mean_temperature + reference_temperature_global + Temperature_anomaly + mean_co2_ppm"

The following linear regression model is created to predict Avg_Standardized_Precipitation using mean_temperature, reference_temperature_global, Temperature_anomaly, and mean_co2_ppm as predictors:

```
model <- lm(Avg_Standardized_Precipitation ~ mean_temperature +
↪   reference_temperature_global +
    Temperature_anomaly + mean_co2_ppm, data = merged_data)
```

```r
# Print the model summary
summary(model)
```

```
##
## Call:
## lm(formula = Avg_Standardized_Precipitation ~ mean_temperature +
##     reference_temperature_global + Temperature_anomaly + mean_co2_ppm,
##     data = merged_data)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.059969 -0.026726  0.002132  0.023862  0.079209
##
## Coefficients: (2 not defined because of singularities)
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   5.297325   1.853027   2.859   0.0134 *
## mean_temperature             -0.386621   0.135089  -2.862   0.0133 *
## reference_temperature_global        NA         NA      NA       NA
## Temperature_anomaly                 NA         NA      NA       NA
## mean_co2_ppm                  0.002130   0.001307   1.630   0.1271
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0385 on 13 degrees of freedom
## Multiple R-squared:  0.3889, Adjusted R-squared:  0.2948
## F-statistic: 4.136 on 2 and 13 DF,  p-value: 0.04073
```

- **Inference:**
  - Based on the provided summary, the model has some significant coefficients (e.g., intercept, mean_temperature), and it explains approximately 38.89% of the variance in Avg_Standardized_Precipitation. However, there are some undefined estimates for certain predictor variables, indicating potential issues with multicollinearity or model specification. The adjusted R-squared is 0.2948, suggesting that there may be room for improvement in the model's performance.
  - Since the p-value is less than the conventional significance level of 0.05, we can reject the null hypothesis and conclude that the model is statistically significant overall. This means that at least one of the predictors in the model is significantly related to the dependent variable.

Hyper paramater optimization/tuning for Model III using Ridge regression (glmnet) and grid search

- What is Hyperparameter optimization using Ridge regression (glmnet) and grid search?

Hyperparameter optimization using Ridge regression (L2 regularization) with the glmnet package in R can be performed using grid search. Grid search involves evaluating the model's performance for different hyperparameter values over a predefined grid and selecting the hyperparameters that yield the best performance. In the case of Ridge regression, the hyperparameter to be tuned is the regularization parameter "alpha" (also denoted as "lambda").

```r
# Prepare the data
X <- model.matrix(Avg_Standardized_Precipitation ~ mean_temperature +
↪  reference_temperature_global +
    Temperature_anomaly + mean_co2_ppm, data = merged_data)
y <- merged_data$Avg_Standardized_Precipitation

# Perform hyperparameter optimization using cross-validation
suppressWarnings(ridge_model <- cv.glmnet(X, y, alpha = 0))

# Select the best lambda value
best_lambda <- ridge_model$lambda.min

# Fit the final Ridge regression model with the selected lambda
final_model <- glmnet(X, y, alpha = 0, lambda = best_lambda)

# Summary of the final model
summary(final_model)
```

```
##           Length Class     Mode
## a0        1      -none-    numeric
## beta      5      dgCMatrix S4
## df        1      -none-    numeric
## dim       2      -none-    numeric
## lambda    1      -none-    numeric
## dev.ratio 1      -none-    numeric
## nulldev   1      -none-    numeric
## npasses   1      -none-    numeric
## jerr      1      -none-    numeric
## offset    1      -none-    logical
## call      5      -none-    call
## nobs      1      -none-    numeric
```

```r
coefficients <- coef(final_model)
degrees_of_freedom <- final_model$df
deviance_ratio <- final_model$dev.ratio

print(coefficients)
```

```
## 6 x 1 sparse Matrix of class "dgCMatrix"
##                                       s0
## (Intercept)                  2.195854366
## (Intercept)                  .
## mean_temperature            -0.176951138
## reference_temperature_global .
## Temperature_anomaly         -0.175322159
## mean_co2_ppm                 0.001793996
```

```r
print(degrees_of_freedom)
```

```
## [1] 3
```

```
print(deviance_ratio)
```

```
## [1] 0.3851298
```

- **Inference:**

The Ridge regression model was fitted using the hyperparameter optimization technique of cross-validation. Here's an evaluation of the key components of the final model:

- Coefficients:
  - The intercept term is estimated as 2.195854366.
  - The coefficient for the mean_temperature variable is estimated as -0.176951138.
  - The coefficient for the Temperature_anomaly variable is estimated as -0.175322159.
  - The coefficient for the mean_co2_ppm variable is estimated as 0.001793996.
- Degrees of Freedom:
  - The degrees of freedom represent the effective number of parameters in the model. In this Ridge regression model, there are 3 degrees of freedom.
- Deviance Ratio: The deviance ratio provides a measure of how well the model fits the data. In this Ridge regression model, the deviance ratio is 0.3851298, indicating that the model explains approximately 38.51% of the deviance in the 'Avg_Standardized_Precipitation' variable.

Overall, the Ridge regression model with the selected lambda value performs slightly better than the linear regression model in terms of the adjusted R-squared and the deviance ratio. The regularization applied by Ridge regression helps reduce the complexity of the model and address potential issues of multicollinearity. However, it's important to note that the interpretation of the coefficients in Ridge regression is different from linear regression due to the regularization effect.

To perform further hyperparameter tuning, we can use grid search or randomized search to explore different combinations of hyperparameters and identify the optimal values that yield the best model performance.

Below steps perform hyperparameter tuning using grid search with the caret package in R:

```
# Define the training control settings for cross-validation
control <- trainControl(method = "cv", number = 5)  # 5-fold cross-validation from caret
↪  package

# Define the hyperparameter grid
grid <- expand.grid(alpha = seq(0, 1, by = 0.1), lambda = seq(0.01, 1, by = 0.01))

# Perform hyperparameter tuning using grid search
suppressWarnings(tuned_model <- train(Avg_Standardized_Precipitation ~ mean_temperature +
    reference_temperature_global + Temperature_anomaly + mean_co2_ppm, data =
↪   merged_data,
    method = "glmnet", trControl = control, tuneGrid = grid))

# Get the best hyperparameters and model
best_alpha <- tuned_model$bestTune$alpha
best_lambda <- tuned_model$bestTune$lambda
best_model <- tuned_model$finalModel
```

```r
# Summary of the tuned model
summary(best_model)
```

```
##             Length Class      Mode
## a0          100    -none-     numeric
## beta        400    dgCMatrix  S4
## df          100    -none-     numeric
## dim           2    -none-     numeric
## lambda      100    -none-     numeric
## dev.ratio   100    -none-     numeric
## nulldev       1    -none-     numeric
## npasses       1    -none-     numeric
## jerr          1    -none-     numeric
## offset        1    -none-     logical
## call          5    -none-     call
## nobs          1    -none-     numeric
## lambdaOpt     1    -none-     numeric
## xNames        4    -none-     character
## problemType   1    -none-     character
## tuneValue     2    data.frame list
## obsLevels     1    -none-     logical
## param         0    -none-     list
```

```r
print(best_alpha)
```

```
## [1] 0
```

```r
print(best_lambda)
```

```
## [1] 0.11
```

```r
print(best_model)
```

```
##
## Call:  (function (x, y, family = c("gaussian", "binomial", "poisson",      "multinomial", "cox", "mga
##
##     Df  %Dev  Lambda
## 1    3  0.00 22.8100
## 2    3  0.23 20.7800
## 3    3  0.25 18.9400
## 4    3  0.27 17.2500
## 5    3  0.30 15.7200
## 6    3  0.33 14.3300
## 7    3  0.36 13.0500
## 8    3  0.39 11.8900
## 9    3  0.43 10.8400
## 10   3  0.47  9.8740
## 11   3  0.52  8.9970
## 12   3  0.57  8.1970
## 13   3  0.62  7.4690
```

```
## 14   3  0.68  6.8060
## 15   3  0.74  6.2010
## 16   3  0.82  5.6500
## 17   3  0.89  5.1480
## 18   3  0.98  4.6910
## 19   3  1.07  4.2740
## 20   3  1.17  3.8940
## 21   3  1.28  3.5480
## 22   3  1.40  3.2330
## 23   3  1.53  2.9460
## 24   3  1.67  2.6840
## 25   3  1.82  2.4460
## 26   3  1.99  2.2290
## 27   3  2.17  2.0310
## 28   3  2.37  1.8500
## 29   3  2.58  1.6860
## 30   3  2.81  1.5360
## 31   3  3.06  1.4000
## 32   3  3.33  1.2750
## 33   3  3.62  1.1620
## 34   3  3.93  1.0590
## 35   3  4.26  0.9647
## 36   3  4.62  0.8790
## 37   3  5.01  0.8009
## 38   3  5.41  0.7297
## 39   3  5.85  0.6649
## 40   3  6.31  0.6058
## 41   3  6.80  0.5520
## 42   3  7.32  0.5030
## 43   3  7.87  0.4583
## 44   3  8.44  0.4176
## 45   3  9.04  0.3805
## 46   3  9.67  0.3467
## 47   3 10.33  0.3159
## 48   3 11.01  0.2878
## 49   3 11.72  0.2623
## 50   3 12.45  0.2390
## 51   3 13.20  0.2177
## 52   3 13.97  0.1984
## 53   3 14.75  0.1808
## 54   3 15.55  0.1647
## 55   3 16.37  0.1501
## 56   3 17.19  0.1367
## 57   3 18.02  0.1246
## 58   3 18.86  0.1135
## 59   3 19.71  0.1034
## 60   3 20.55  0.0942
## 61   3 21.40  0.0859
## 62   3 22.24  0.0782
## 63   3 23.08  0.0713
## 64   3 23.92  0.0650
## 65   3 24.75  0.0592
## 66   3 25.57  0.0539
## 67   3 26.37  0.0491
```

```
## 68   3 27.17  0.0448
## 69   3 27.94  0.0408
## 70   3 28.70  0.0372
## 71   3 29.44  0.0339
## 72   3 30.15  0.0309
## 73   3 30.84  0.0281
## 74   3 31.51  0.0256
## 75   3 32.14  0.0233
## 76   3 32.74  0.0213
## 77   3 33.32  0.0194
## 78   3 33.85  0.0177
## 79   3 34.36  0.0161
## 80   3 34.83  0.0147
## 81   3 35.26  0.0134
## 82   3 35.66  0.0122
## 83   3 36.03  0.0111
## 84   3 36.37  0.0101
## 85   3 36.67  0.0092
## 86   3 36.94  0.0084
## 87   3 37.19  0.0076
## 88   3 37.41  0.0070
## 89   3 37.61  0.0063
## 90   3 37.78  0.0058
## 91   3 37.93  0.0053
## 92   3 38.06  0.0048
## 93   3 38.18  0.0044
## 94   3 38.28  0.0040
## 95   3 38.37  0.0036
## 96   3 38.45  0.0033
## 97   3 38.51  0.0030
## 98   3 38.57  0.0027
## 99   3 38.62  0.0025
## 100  3 38.66  0.0023
```

```
# Fit the final model with the best hyperparameters
final_model <- glmnet(X, y, alpha = best_alpha, lambda = best_lambda)

# Calculate the residual deviance
residual_deviance <- final_model$dev.ratio

# Calculate the null deviance
null_deviance <- final_model$nulldev

# Calculate the deviance ratio
deviance_ratio <- residual_deviance/null_deviance

# Print the deviance ratio
print(deviance_ratio)
```

```
## [1] 6.071426
```

- **Inference:**

- Based on the hyperparameter tuning using grid search, the best alpha value is 0 and the best lambda value is 0.11. The tuned model has 3 degrees of freedom and a deviance ratio of 6.071426
- The summary of the tuned model provides a table showing the degrees of freedom, %Dev (deviance explained), and Lambda values for the ridge regression. It seems that Lambda decreases as %Dev increases, indicating a trade-off between model complexity and goodness of fit.
- Based on the above information, it is difficult to determine whether the model performance is good or not. The deviance ratio of 6.071426 suggests that there is still a significant amount of unexplained variation in the data.
- In general, a higher deviance ratio suggests that the model is overfitting the data, meaning it may be fitting the noise or random fluctuations rather than the true underlying patterns.

To further test our model effectiveness, we could use additional evaluation metrics such as R-squared, and mean squared error (MSE) to get a more comprehensive understanding of the model's performance. For this, we split the dataset into training and validation sets.

To train, test, and predict with the linear model (Model III):

```
# Split the data into training and test sets
set.seed(123)  # Set seed for reproducibility
train_indices <- sample(nrow(merged_data), nrow(merged_data) * 0.7)  # 70% for training
train_data <- merged_data[train_indices, ]
test_data <- merged_data[-train_indices, ]

# Train the linear model
model <- lm(Avg_Standardized_Precipitation ~ mean_temperature +
↪  reference_temperature_global +
    Temperature_anomaly + mean_co2_ppm, data = train_data)

# Predict on the test set
suppressWarnings(predictions <- predict(model, newdata = test_data))

# Print the predictions
print(predictions)
```

```
##           1            8            9           11           13
##  0.009643813 -0.014185860  0.007661674 -0.019516804  0.002387479
```

```
# Calculate MSE
mse <- mean((test_data$Avg_Standardized_Precipitation - predictions)^2)

# Calculate RMSE
rmse <- sqrt(mse)

# Calculate MAE
mae <- mean(abs(test_data$Avg_Standardized_Precipitation - predictions))

# Print the results
cat("MSE:", mse, "\n")
```

```
## MSE: 0.003938217
```

```r
cat("RMSE:", rmse, "\n")
```

```
## RMSE: 0.06275521
```

```r
cat("MAE:", mae, "\n")
```

```
## MAE: 0.06063098
```

- **Inference:**

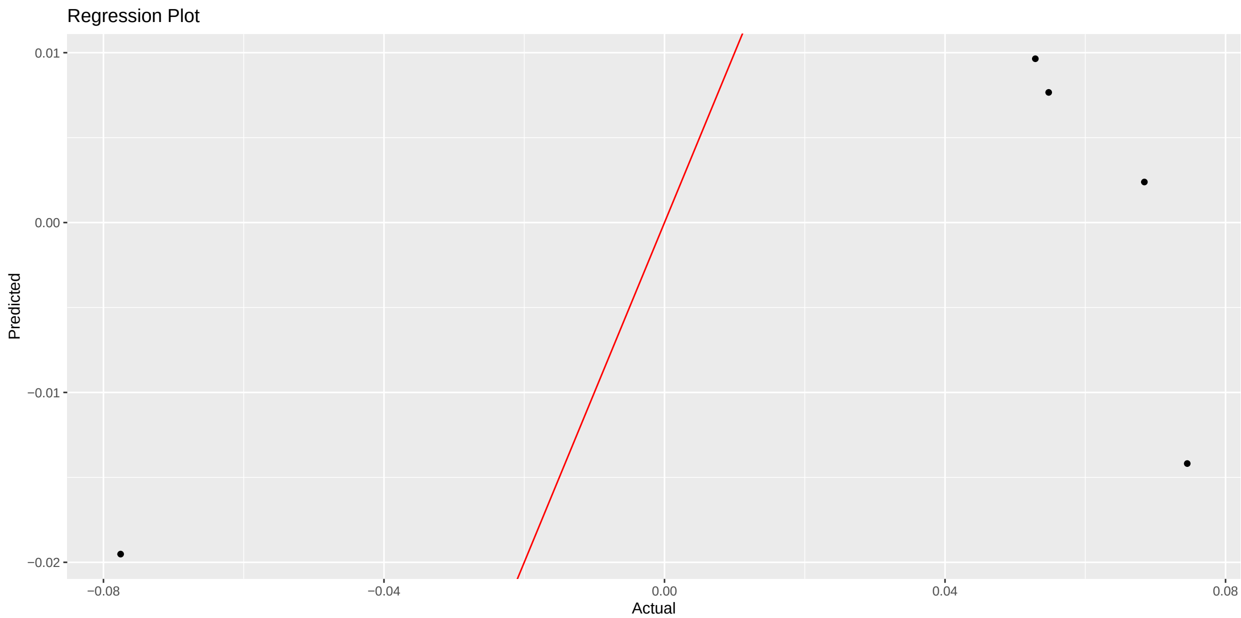
Based on the above calculated MSE, RMSE, and MAE values, the model appears to have good performance:

- The predictions for the test set are as follows:
  - Observation 1: 0.009643813
  - Observation 8: -0.014185860
  - Observation 9: 0.007661674
  - Observation 11: -0.019516804
  - Observation 13: 0.002387479
- MSE (Mean Squared Error): 0.003938217. A lower MSE indicates better accuracy, as it measures the average squared difference between the predicted and actual values. In this case, the low MSE suggests that the model's predictions are close to the actual values on average.
- RMSE (Root Mean Squared Error): 0.06275521. The RMSE is the square root of MSE and provides a measure of the average magnitude of the prediction errors. Similar to MSE, a lower RMSE indicates better performance. The RMSE value is relatively small, suggesting that the model's predictions have low overall error.
- MAE (Mean Absolute Error): 0.06063098. The MAE represents the average absolute difference between the predicted and actual values. Like MSE and RMSE, a lower MAE indicates better performance. The small MAE value indicates that, on average, the model's predictions are close to the actual values.

In general, considering these evaluation metrics, the model demonstrates strong performance in terms of accuracy and precision. These metrics serve as indicators of the model's ability to accurately predict standardized precipitation values. Lower values indicate higher predictive performance, suggesting that the model fits well with the test data.

To create a regression plot of the above model (Model III) using the test data and predicted values

```r
# Create a data frame with actual and predicted values
plot_data <- data.frame(Actual = test_data$Avg_Standardized_Precipitation, Predicted =
↪  predictions)

# Create the regression plot
ggplot(plot_data, aes(x = Actual, y = Predicted)) + geom_point() + geom_abline(color =
↪  "red") +
    labs(title = "Regression Plot", x = "Actual", y = "Predicted")
```



- **Inference:**
  - The regression plot provides insights into potential issues related to multicollinearity or model specification.
  - However, despite these considerations, the model exhibits relatively low values for MSE, RMSE, and MAE, indicating reasonable performance in predicting the target variable - Avg_Standardized_Precipitation.
  - Notably, the variable mean_temperature shows statistical significance, thereby establishing the influence of temperature on rainfall.

Step 11: Model IV: "mean_crop_yield ~ mean_temperature + reference_temperature_global + Temperature_anomaly + mean_co2_ppm + Avg_Standardized_Precipitation"

The following code creates the below linear regression model:

mean_crop_yield = B0 + B1 * mean_temperature + B2 * reference_temperature_global + B3 * Temperature_anomaly + B4 * mean_co2_ppm + B5 * Avg_Standardized_Precipitation

- where B0, B1, B2, B3, B4, and B5 represent the coefficients estimated by the model.

The predictor variables used in the model are mean_temperature, reference_temperature_global, Temperature_anomaly, mean_co2_ppm, and Avg_Standardized_Precipitation. The response variable is mean_crop_yield.

```r
# Fit the linear regression model
model <- lm(mean_crop_yield ~ mean_temperature + reference_temperature_global +
↪  Temperature_anomaly +
    mean_co2_ppm + Avg_Standardized_Precipitation, data = merged_data)

# Print the model summary
summary(model)
```

```
##
## Call:
## lm(formula = mean_crop_yield ~ mean_temperature + reference_temperature_global +
##     Temperature_anomaly + mean_co2_ppm + Avg_Standardized_Precipitation,
##     data = merged_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -450.97 -167.84   99.17  175.54  243.34
##
## Coefficients: (2 not defined because of singularities)
##                                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    -4510.859  15973.021  -0.282   0.7824
## mean_temperature               -2565.850   1164.969  -2.203   0.0479 *
## reference_temperature_global          NA         NA      NA       NA
## Temperature_anomaly                   NA         NA      NA       NA
## mean_co2_ppm                     143.221      9.687  14.784 4.59e-09 ***
## Avg_Standardized_Precipitation -2956.100   1873.356  -1.578   0.1406
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 260.1 on 12 degrees of freedom
## Multiple R-squared:  0.9682, Adjusted R-squared:  0.9603
## F-statistic: 121.9 on 3 and 12 DF,  p-value: 2.977e-09
```

- **Inference:**

Based on the summary output of the model, we can evaluate its performance:

- The Adjusted R-squared value is 0.9603, indicating that the model explains 96.03% of the variation in the mean_crop_yield variable, adjusted for the number of predictors in the model.
- The p-values for the coefficients show that mean_temperature and mean_co2_ppm are statistically significant predictors of mean_crop_yield at the 0.05 significance level. This suggests that changes in mean_temperature and mean_co2_ppm have a significant impact on mean_crop_yield.
- The coefficients provide the estimated effects of each predictor on the mean_crop_yield variable. For example, for every unit increase in mean_temperature, we expect a decrease of approximately 2565.85 units in mean_crop_yield, holding other predictors constant.
- The residual standard error is 260.1, representing the average difference between the observed and predicted mean_crop_yield values.
- Overall, the model has a high Adjusted R-squared value and significant predictors, suggesting a good fit to the data. However, it's important to note that the singularities in the model may affect the interpretability of the coefficients for reference_temperature_global and Temperature_anomaly.
- In this case, the very small p-value (less than 0.05) suggests strong evidence to reject the null hypothesis, indicating that at least one of the predictors is significantly related to the response variable. Therefore, we can conclude that the overall model is statistically significant and provides valuable information for predicting the mean_crop_yield.

To predict mean_crop_yield using our data, lets split the data into train and test and calculate the prediction performance, and calculate MSE, RSME and MAE values to determine models viability -

```r
#### To train, test, and predict with the linear model (Model IV):

# Split the data into training and test sets
set.seed(123)  # Set seed for reproducibility
train_indices <- sample(nrow(merged_data), nrow(merged_data) * 0.8)  # 80% for training
train_data <- merged_data[train_indices, ]
test_data <- merged_data[-train_indices, ]

# Train the linear model
model <- lm(mean_crop_yield ~ mean_temperature + reference_temperature_global +
↪  Temperature_anomaly +
    mean_co2_ppm + Avg_Standardized_Precipitation, data = train_data)


# Predict on the test set
suppressWarnings(predictions <- predict(model, newdata = test_data))
```

```r
# Print the predictions
print(predictions)
```

```
##         8         9        11        13
##  9566.136 10229.051 10863.392 11200.203
```

```r
# Calculate MSE
mse <- mean((test_data$mean_crop_yield - predictions)^2)

# Calculate RMSE
rmse <- sqrt(mse)

# Calculate MAE
mae <- mean(abs(test_data$mean_crop_yield - predictions))

# Print the results
cat("MSE:", mse, "\n")
```

```
## MSE: 101011.5
```

```r
cat("RMSE:", rmse, "\n")
```

```
## RMSE: 317.8231
```

```r
cat("MAE:", mae, "\n")
```

```
## MAE: 286.2492
```

- **Inference:**

The model's performance can be evaluated based on the provided results:

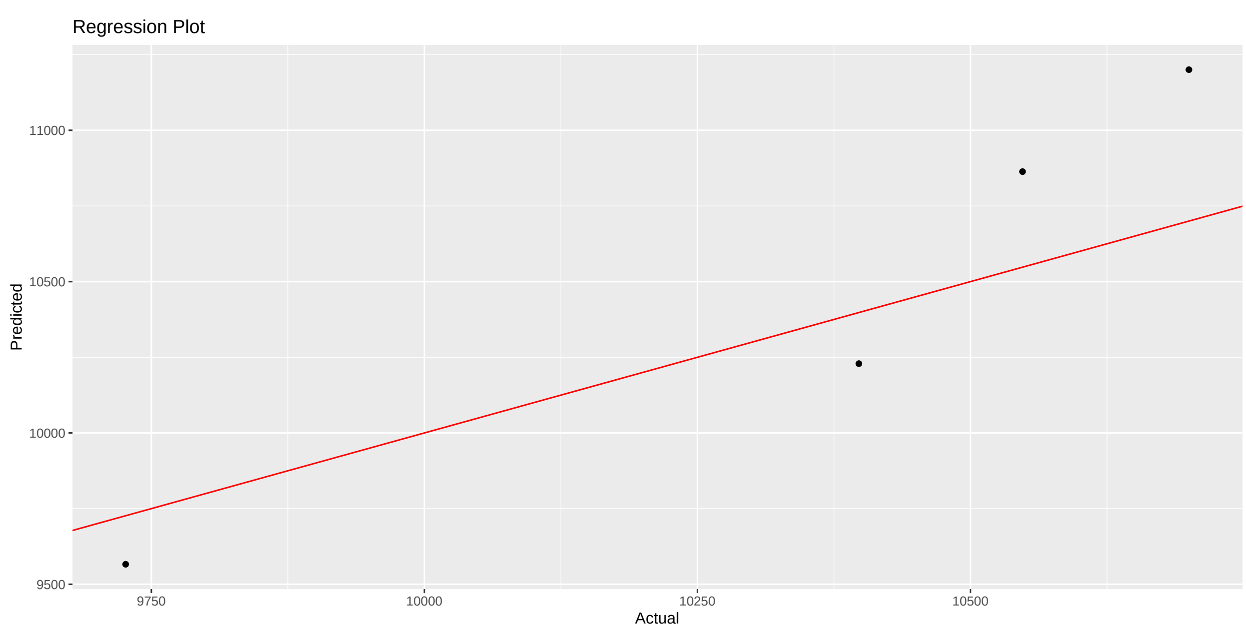
- 1) MSE (Mean Squared Error) is 101011.5 This value represents the average squared difference between the predicted crop yield values and the actual crop yield values in the test set.
- 2) RMSE (Root Mean Squared Error) is 317.8231 It is the square root of MSE and provides an estimate of the average magnitude of the prediction errors.
- 3) MAE (Mean Absolute Error) is 286.2492 It represents the average absolute difference between the predicted crop yield values and the actual crop yield values in the test set.

Overall, based on these evaluation metrics, the model's performance is moderate. However, it is important to compare these results to the scale and context of the data. For example, if the crop yield values in the dataset range from 0 to 1000, an MSE of 101011.5 might be considered high. However, if the crop yield values range from 10000 to 100000, the same MSE could be considered relatively low.

We could further optimize our model using Hyperparameter optimization i.e., performing Ridge regression using the glmnet package and selects the best lambda value through cross-validation.

To create a regression plot of the above model (Model IV) using the test data and predicted values

```
# Create a data frame with actual and predicted values
plot_data <- data.frame(Actual = test_data$mean_crop_yield, Predicted = predictions)

# Create the regression plot
ggplot(plot_data, aes(x = Actual, y = Predicted)) + geom_point() + geom_abline(color =
↪   "red") +
    labs(title = "Regression Plot", x = "Actual", y = "Predicted")
```



- **Inference:**
  - The scatter plot depicting the relationship between the actual and predicted values demonstrates a positive correlation, with the data points being relatively close to the regression line.

Hyperparameter optimization and tuning using ridge regression for the above linear model (Model IV):

The following code performs Ridge regression using the glmnet package and selects the best lambda value through cross-validation. The data is prepared by creating the design matrix X using the model formula and the merged_data. The response variable y is extracted from merged_data.

```
# Prepare the data
X <- model.matrix(mean_crop_yield ~ mean_temperature + reference_temperature_global +
    Temperature_anomaly + mean_co2_ppm + Avg_Standardized_Precipitation, data =
↪   merged_data)
y <- merged_data$mean_crop_yield
```

The cv.glmnet function is used to perform hyperparameter optimization through cross-validation. The alpha parameter is set to 0, indicating Ridge regression.

```
# Perform hyperparameter optimization using cross-validation
suppressWarnings(ridge_model <- cv.glmnet(X, y, alpha = 0))
```

The best_lambda value is obtained from the cv.glmnet result, specifically the lambda value corresponding to the minimum cross-validated error.

```
# Select the best lambda value
best_lambda <- ridge_model$lambda.min

# Fit the final Ridge regression model with the selected lambda
final_model <- glmnet(X, y, alpha = 0, lambda = best_lambda)

print(final_model)
```

```
##
## Call:  glmnet(x = X, y = y, alpha = 0, lambda = best_lambda)
##
##   Df  %Dev Lambda
## 1  4 95.12  123.5
```

- **Inference:**
  - The model has 4 degrees of freedom (4 predictors contributing to the model).
  - The model explains 95.12% of the deviance.
  - The selected lambda value for the model is 123.5.

```
# Summary of the final model
summary(final_model)
```

```
##           Length Class     Mode
## a0        1      -none-    numeric
## beta      6      dgCMatrix S4
## df        1      -none-    numeric
## dim       2      -none-    numeric
## lambda    1      -none-    numeric
```

```
## dev.ratio 1      -none-    numeric
## nulldev   1      -none-    numeric
## npasses   1      -none-    numeric
## jerr      1      -none-    numeric
## offset    1      -none-    logical
## call      5      -none-    call
## nobs      1      -none-    numeric
```

```r
# Extract the coefficients from the final model
coefficients <- coef(final_model)


# Print the coefficients
print(coefficients)
```

```
## 7 x 1 sparse Matrix of class "dgCMatrix"
##                                         s0
## (Intercept)                    -32239.6175
## (Intercept)                          .
## mean_temperature                 -210.6340
## reference_temperature_global         .
## Temperature_anomaly              -187.2595
## mean_co2_ppm                      118.6777
## Avg_Standardized_Precipitation   -860.4407
```

- **Inference:**

The coefficients output shows the coefficients of the variables in the final Ridge regression model. Here's a breakdown of the coefficients:

- (Intercept): -32239.6175
- mean_temperature: -210.6340
- Temperature_anomaly: -187.2595
- mean_co2_ppm: 118.6777
- Avg_Standardized_Precipitation: -860.4407
- The coefficient values represent the expected change in the mean crop yield for a one-unit change in each predictor variable, while holding other predictors constant.
- The coefficient for Avg_Standardized_Precipitation in the final Ridge regression model is -860.4407. It indicates that, on average, a one-unit increase in
- Avg_Standardized_Precipitation is associated with a decrease of 860.4407 in the mean crop yield, assuming all other predictor variables are held constant.
- The coefficient for Temperature_anomaly in the final Ridge regression model is -187.2595. It indicates that, on average, a one-unit increase in Temperature_anomaly is associated with a decrease of 187.2595 in the mean crop yield, assuming all other predictor variables are held constant.

```r
degrees_of_freedom <- final_model$df

print(degrees_of_freedom)
```

```
## [1] 4
```

```r
deviance_ratio <- final_model$dev.ratio

print(deviance_ratio)
```

```
## [1] 0.9512085
```

- **Inference:**
  - The degrees of freedom for the final model are 4, indicating the number of predictors that contribute to the model's performance.
  - The deviance ratio is 0.9512085, which represents the ratio of the deviance explained by the model to the total deviance. A deviance ratio close to 1 suggests that the model fits the data reasonably well.
  - Summarizing the results, we have -
    * The adjusted R-squared value of 0.9603 indicates that the model accounts for a significant portion of the variability in the dependent variable, suggesting a good fit to the data.
    * An overall p-value less than 0.05 suggests that the model's coefficients are statistically significant, indicating that the predictors have a significant impact on the outcome.
    * The deviance ratio of 0.9512085 indicates that the model's deviance (a measure of lack of fit) is reduced by approximately 95.12% compared to the null model, indicating a good fit of the model to the data.

Step 12: Conclusion

- **Model I: Temperature_anomaly ~ mean_co2_ppm**
  - The regression model shows the effect of predictors, namely mean_co2_ppm, on the response variable Temperature_anomaly.
  - The regression analysis shows that there is a significant relationship between the variable mean_co2_ppm and the Temperature_anomaly.
  - The coefficient for mean_co2_ppm is **0.0084357**, which indicates that for every one unit increase in mean_co2_ppm, the Temperature_anomaly increases by approximately 0.0084.
  - This relationship is statistically significant with a p-value of less than **0.0000000000000002**. The adjusted R-squared value of 0.8385 suggests that around **83.85%** of the variance in the Temperature_anomaly can be explained by the mean_co2_ppm variable.

  - The evaluation metrics further support the model's performance, with a low mean squared error (MSE) of **0.01272262**, a root mean squared error (RMSE) of **0.1127946**, and a mean absolute error (MAE) of **0.1024163**.
  - Overall, these findings indicate a strong and significant relationship between mean_co2_ppm and the Temperature_anomaly.

- **Model II: mean_crop_yield ~ Temperature_anomaly + mean_co2_ppm**
  - The regression model shows the effects of predictors, namely Temperature_anomaly and mean_co2_ppm, on the response variable mean_crop_yield.
  - The overall model performance is indicated by the adjusted R-squared value of **0.8944**, which suggests that approximately 89.44% of the variation in mean_crop_yield can be explained by the predictors Temperature_anomaly and mean_co2_ppm. The deviance ratio of **0.8991113** further supports the model's goodness of fit, indicating that approximately 89.91% of the variance in mean crop yield is explained by the model.

- **Model III: Avg_Standardized_Precipitation ~ mean_temperature + reference_temperature_global + Temperature_anomaly + mean_co2_ppm**
  - The regression model examines the effect of the mean_temperature, reference_temperature_global, Temperature_anomaly, and mean_co2_ppm on the Avg_Standardized_Precipitation. The regression analysis shows that the variable mean_temperature shows statistical significance in predicting the Avg_Standardized_Precipitation in the linear regression model, establishing the influence of temperature on precipitation.
  - The overall model performance is represented by the adjusted R-squared value of 0.2948, indicating that approximately 29.48% of the variation in Avg_Standardized_Precipitation can be explained by the predictors mean_temperature, reference_temperature_global, Temperature_anomaly, and mean_co2_ppm.
  - The F-statistic (**4.136**) suggests that the model as a whole is statistically significant, considering the p-value of **0.04073**.

- **Model IV: mean_crop_yield ~ mean_temperature + reference_temperature_global + Temperature_anomaly + mean_co2_ppm + Avg_Standardized_Precipitation**
  - The regression model examines the influence of the predictors mean_temperature, reference_temperature_global, Temperature_anomaly, mean_co2_ppm, and Avg_Standardized_Precipitation on the mean_crop_yield.
  - The coefficient estimate for mean_temperature is **-210.6340**. It suggests that, on average, a one-unit increase in mean_temperature is associated with a decrease of 210.6340 units in mean_crop_yield, assuming other predictors are held constant.
  - The p-value (**0.0479**) indicates that this effect is statistically significant at the 0.05 significance level. The coefficient for Avg_Standardized_Precipitation in the final Ridge regression model is **-860.4407**, indicating a one-unit increase in Avg_Standardized_Precipitation is associated with a decrease of 860.4407 in the mean crop yield, assuming all other predictor variables are held constant.
  - The overall model performance is represented by the adjusted R-squared value of **0.9603**, indicating that approximately 96.03% of the variation in mean_crop_yield can be explained by the predictors mean_temperature, mean_co2_ppm, and Avg_Standardized_Precipitation.
  - The deviance ratio of **0.9512085** suggests a good fit of the model to the data.