script_PA

2025-08-11

Import the libraries.

```
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse
2.0.0 ──
## ✓ dplyr     1.1.4     ✓ readr     2.1.5
## ✓ forcats   1.0.0     ✓ stringr   1.5.1
## ✓ ggplot2   3.5.2     ✓ tibble    3.3.0
## ✓ lubridate 1.9.4     ✓ tidyr     1.3.1
## ✓ purrr     1.1.0
## ── Conflicts ────────────────────────────────────────
tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all
conflicts to become errors
```

```
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(lubridate)
library(ggplot2)
library(scales)
```

```
##
## Attaching package: 'scales'
##
## The following object is masked from 'package:purrr':
##
##     discard
##
## The following object is masked from 'package:readr':
##
##     col_factor
```

```
library(readr)
library(dplyr)
```

Load the dataset.

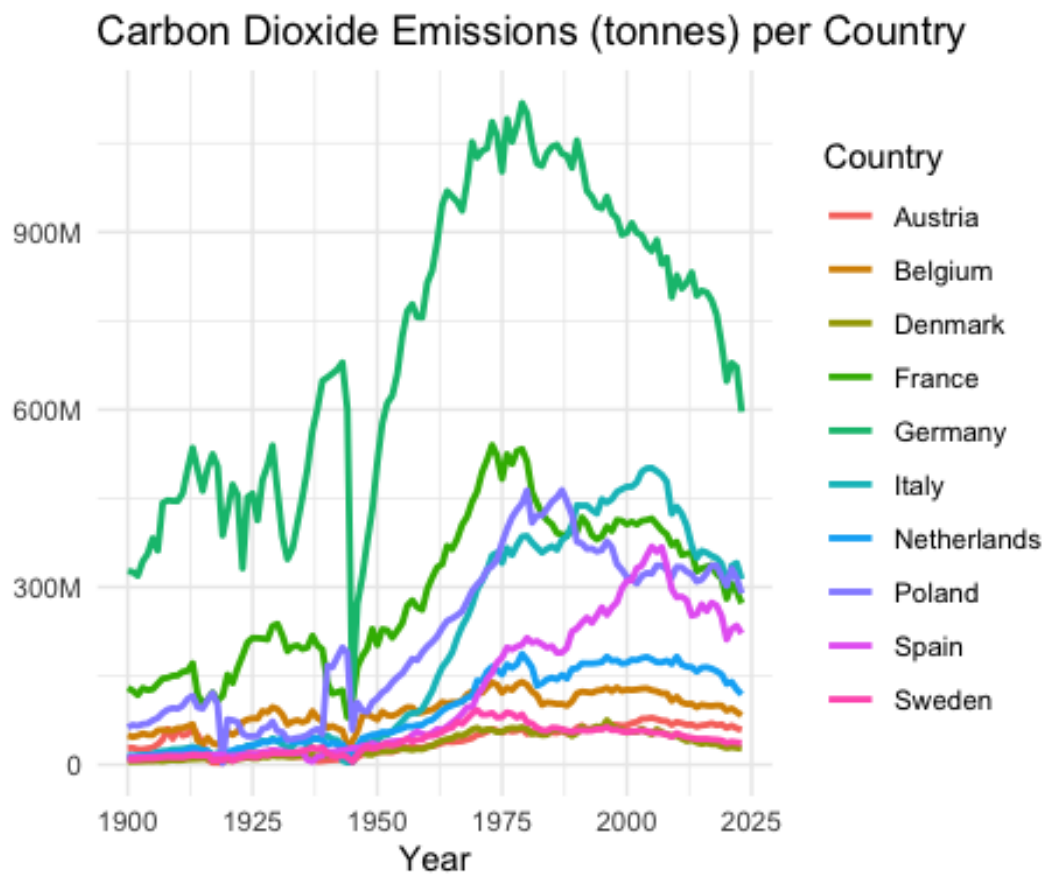
```
data <- read_csv('CO2emissions.csv')
```

```
## Rows: 1488 Columns: 4
## ── Column specification
────────────────────────────────────────────────────────
## Delimiter: ","
## chr (2): Country, Code
## dbl (2): Year, Annual CO2 emissions (tonnes)
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

General overview graph without forecasts.

```
data |>
  filter(!Country %in% c("World", "European Union (27)")) |>
  ggplot(aes(x = Year, y = `Annual CO2 emissions (tonnes)`, color = Country))
+
  geom_line(size = 1) +
  scale_y_continuous(labels = label_number(scale_cut = cut_short_scale())) +
  labs(
    title = "Carbon Dioxide Emissions (tonnes) per Country",
    x = "Year",
    y = "",
    color = "Country"
  ) +
  theme_minimal()
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## ℹ Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



Now we build a function that splits the time series into train (until 2019) and test (2020-2023) and compares the performances of ETS and ARIMA for every country. We evaluate the models using RMSE (Root Mean Squared Error).

```
compare_models <- function(df, country_name) {
  country_data <- df %>%
    dplyr::filter(Country == country_name) %>%
    arrange(Year)

  # Make it a ts
  ts_data <- ts(country_data$`Annual CO2 emissions (tonnes)`, start =
min(country_data$Year), frequency = 1)

  # Train/test split
  train_ts <- window(ts_data, end = 2019)
  test_ts  <- window(ts_data, start = 2020, end = 2023)
  h <- length(test_ts)

  # ETS
  ets_fit <- ets(train_ts)
  # ETS(ANN) returns flat predictions -> so we force ETS(AAN)
  if (grepl("ETS\\(A,N,N\\)", ets_fit$method)) {
  ets_fit <- ets(train_ts, model = "AAN")
```

```
  }
  ets_fc <- forecast(ets_fit, h = h)
  ets_rmse <- sqrt(mean((test_ts - ets_fc$mean)^2))
  ets_label <- ets_fit$method

  # ARIMA
  arima_fit <- auto.arima(train_ts, stepwise = FALSE, approximation = FALSE)
  arima_order <- arimaorder(arima_fit)

  # If ARIMA(0,1,0), then force a drift to avoid flat forecasts
  if (all(arima_order == c(0, 1, 0))) {
    arima_fit <- Arima(train_ts, order = c(0, 1, 0), include.drift = TRUE)
  }

  arima_fc   <- forecast(arima_fit, h = h)
  arima_rmse <- sqrt(mean((test_ts - arima_fc$mean)^2))
  arima_order <- arimaorder(arima_fit)
  has_drift <- "drift" %in% names(coef(arima_fit))

  # Best model
  if (ets_rmse < arima_rmse) {
    best_model <- ets_label
  } else {
    suffix <- if (has_drift) ") + drift" else ")"
    best_model <- paste0("ARIMA(", arima_order[1], ",", arima_order[2], ",",
arima_order[3], suffix)
  }

  # Return the table
  tibble(
    Country = country_name,
    ETS_RMSE = ets_rmse,
    ARIMA_RMSE = arima_rmse,
    Best_Model = best_model
  )
}
```

Apply the function to all the countries.

```
# Make a list of all the countries
country_list <- data %>% distinct(Country) %>% pull(Country)

# See results
results <- map_dfr(country_list, ~compare_models(data, .x))
print(results)
```

```
## # A tibble: 12 × 4
##    Country               ETS_RMSE  ARIMA_RMSE Best_Model
##    <chr>                    <dbl>       <dbl> <chr>
##  1 Austria                6471551.    6106282. ARIMA(0,1,4)
```

```
##  2 Belgium                13397039.   11184008. ARIMA(1,1,3)
##  3 Denmark                 1640563.    3078411. ETS(M,A,N)
##  4 European Union (27)  165355401.  191969299. ETS(A,A,N)
##  5 France                 25176203.   25347771. ETS(A,Ad,N)
##  6 Germany                72036682.   77315014. ETS(A,Ad,N)
##  7 Italy                  18389382.   18386440. ARIMA(1,1,1)
##  8 Netherlands            25178780.   26491215. ETS(A,A,N)
##  9 Poland                 26505648.   21423190. ARIMA(0,1,0) + drift
## 10 Spain                  35705898.   22040784. ARIMA(3,1,0)
## 11 Sweden                  3018231.    4260719. ETS(A,Ad,N)
## 12 World                1627139527. 1593542557. ARIMA(0,2,2)
```

Now let's compute forecasts (2024-2030) for every country using the respective best model.

```
forecast_2030 <- function(df, country_name, best_model) {
  country_data <- df %>%
    dplyr::filter(Country == country_name) %>%
    arrange(Year)

  ts_data <- ts(country_data$`Annual CO2 emissions (tonnes)`, start =
min(country_data$Year), frequency = 1)

  h <- 2030 - 2023  # Years to forecast

  # ETS
  if (startsWith(best_model, "ETS")) {
  fit <- ets(ts_data)

  # Avoid ETS(A,N,N), brings to flat forecasts. Force ETS(A,A,N) instead
  if (grepl("ETS\\(A,N,N\\)", fit$method)) {
    fit <- ets(ts_data, model = "AAN")
  }

  fc <- forecast(fit, h = h)
  model_label <- fit$method
}

  # ARIMA (with drift if (0,1,0))
  else if (startsWith(best_model, "ARIMA")) {
    drift <- grepl("\\+ drift$", best_model)

    # Extract parameters
    order_vals <- gsub("ARIMA\\(|\\)|\\+ drift", "", best_model)
    order_vec <- as.integer(strsplit(order_vals, ",")[[1]])

    fit <- Arima(ts_data, order = order_vec, include.drift = drift)
    fc <- forecast(fit, h = h)
```

```
    model_label <- best_model
  }

  # Extract forecasts
  years <- as.character(2024:2030)
  preds <- as.numeric(fc$mean)
  tibble_row <- tibble(
    Country = country_name,
    Best_Model = model_label,
    !!!set_names(as.list(preds), years)
  )

  return(tibble_row)
}
```

Let's do it for all the countries.

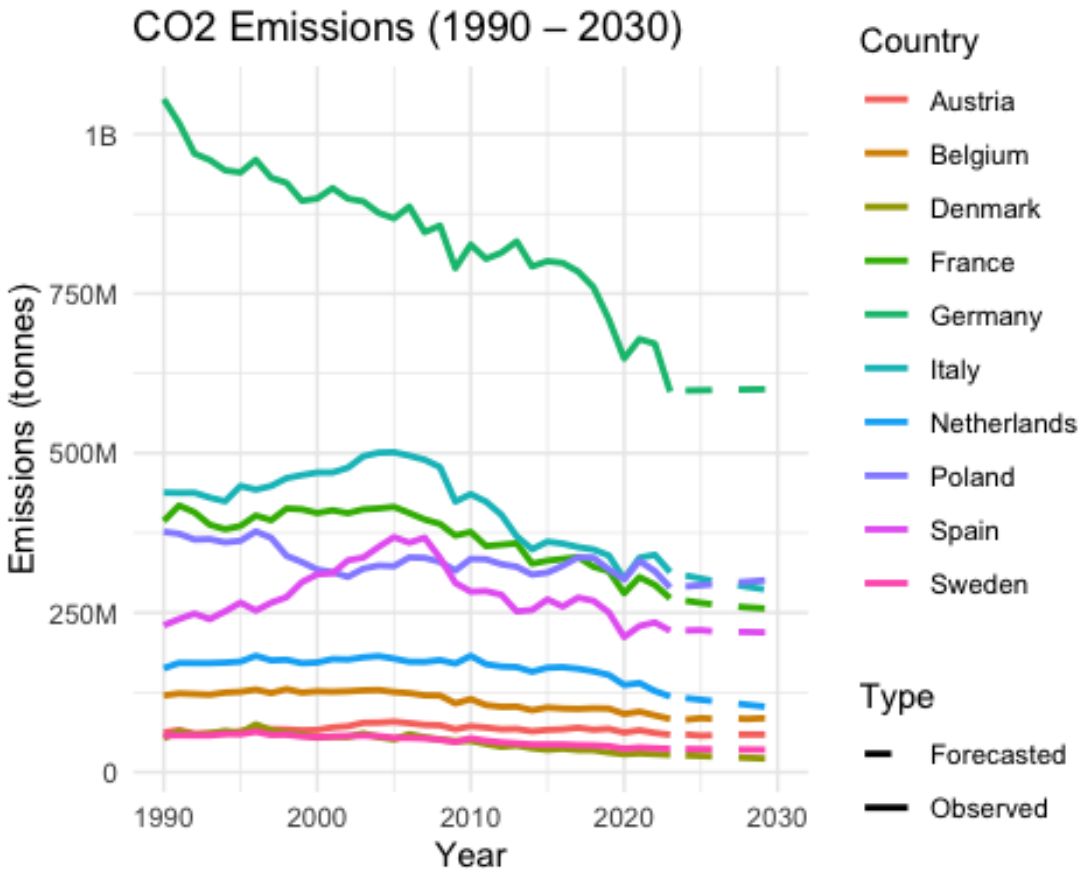
```
forecast_table <- pmap_dfr(
  list(
    country_name = results$Country,
    best_model   = results$Best_Model
  ),
  ~forecast_2030(data, ..1, ..2)
)

# Show table with best models
print(forecast_table)
```

```
## # A tibble: 12 × 9
##    Country    Best_Model  `2024`  `2025`  `2026`  `2027`  `2028`  `2029`
`2030`
##    <chr>      <chr>        <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
<dbl>
##  1 Austria    ARIMA(0,1… 5.91e 7 5.72e 7 5.82e 7 5.90e 7 5.90e 7 5.90e 7
5.90e 7
##  2 Belgium    ARIMA(1,1… 8.23e 7 8.47e 7 8.33e 7 8.46e 7 8.34e 7 8.45e 7
8.35e 7
##  3 Denmark    ETS(M,A,N) 2.64e 7 2.54e 7 2.44e 7 2.35e 7 2.25e 7 2.16e 7
2.06e 7
##  4 European … ETS(A,A,N) 2.45e 9 2.39e 9 2.33e 9 2.27e 9 2.21e 9 2.15e 9
2.09e 9
##  5 France     ETS(A,Ad,… 2.69e 8 2.65e 8 2.62e 8 2.60e 8 2.58e 8 2.57e 8
2.56e 8
##  6 Germany    ETS(A,Ad,… 5.98e 8 5.99e 8 5.99e 8 5.99e 8 6.00e 8 6.00e 8
6.00e 8
##  7 Italy      ARIMA(1,1… 3.08e 8 3.03e 8 2.99e 8 2.95e 8 2.91e 8 2.87e 8
2.84e 8
##  8 Netherlan… ETS(A,A,N) 1.16e 8 1.14e 8 1.11e 8 1.08e 8 1.06e 8 1.03e 8
1.01e 8
##  9 Poland     ARIMA(0,1… 2.91e 8 2.93e 8 2.95e 8 2.97e 8 2.98e 8 3.00e 8
```

```
3.02e 8
## 10 Spain      ARIMA(3,1… 2.22e 8 2.23e 8 2.20e 8 2.20e 8 2.20e 8 2.19e 8
2.19e 8
## 11 Sweden     ETS(A,Ad,… 3.63e 7 3.60e 7 3.57e 7 3.55e 7 3.53e 7 3.52e 7
3.50e 7
## 12 World      ARIMA(0,2… 3.82e10 3.86e10 3.91e10 3.95e10 3.99e10 4.03e10
4.07e10
```

Now, build a general line chart with forecasts.

```
historical_df <- data %>%
  filter(Year >= 1990, Year <= 2023,
         !Country %in% c("World", "European Union (27)")) %>%
  select(Country, Year, Emissions = `Annual CO2 emissions (tonnes)`) %>%
  mutate(Forecast = FALSE)

# Forecasts for countries
forecast_long <- forecast_table %>%
  filter(!Country %in% c("World", "European Union (27)")) %>%
  pivot_longer(cols = `2024`:`2030`, names_to = "Year", values_to =
"Emissions") %>%
  mutate(Year = as.integer(Year),
         Forecast = TRUE)

# Join observed data and forecasts (not at a country-level)
full_df <- bind_rows(historical_df, forecast_long) %>%
  mutate(Forecast = ifelse(Forecast, "Forecasted", "Observed"))

# Plot the linechart
ggplot(full_df, aes(x = Year, y = Emissions, color = Country, linetype =
Forecast)) +
  geom_line(size = 1) +
  scale_linetype_manual(values = c("Observed" = "solid", "Forecasted" =
"dashed")) +
  scale_y_continuous(labels = label_number(scale_cut = cut_short_scale())) +
  labs(title = "CO2 Emissions (1990 – 2030)",
       x = "Year",
       y = "Emissions (tonnes)",
       color = "Country",
       linetype = "Type") +
  theme_minimal()
```



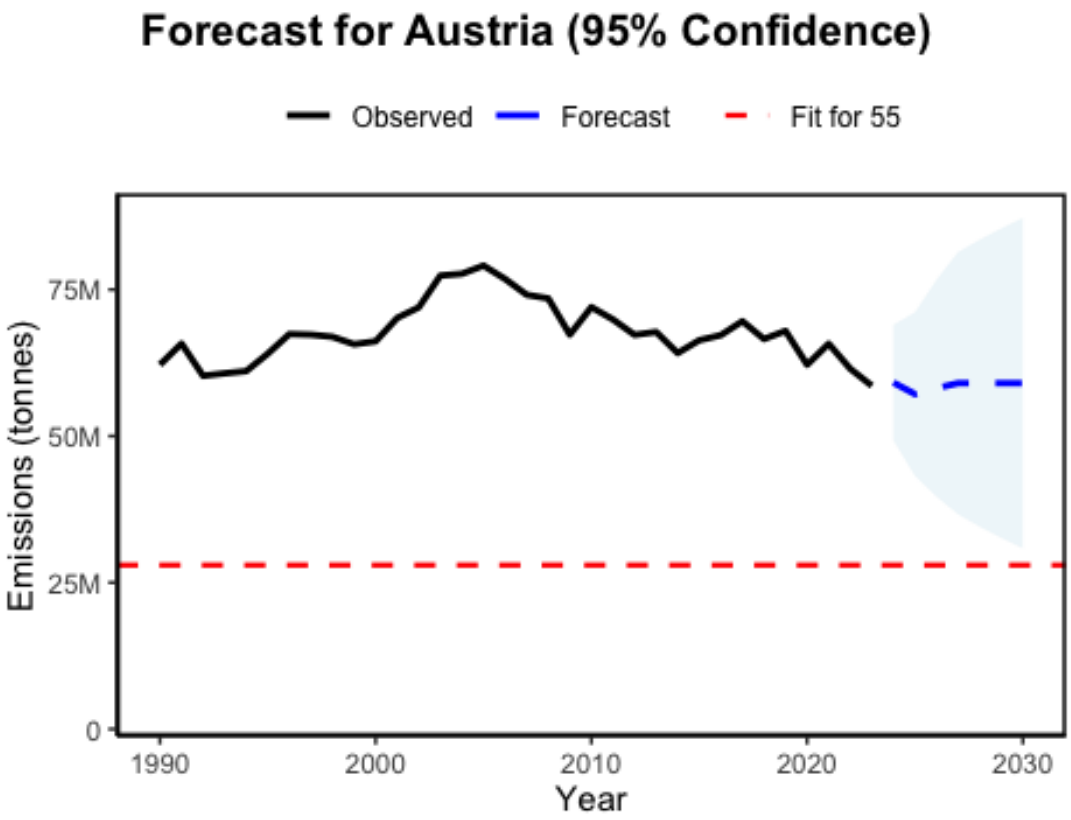
Now let's see the forecast more in detail for every country. Also visualizing the Fit for 55 target.

```
# Fit for 55 target
get_target_value <- function(ts_data) {
  ts_data[which(time(ts_data) == 1990)] * 0.45
}

# A df for each country
generate_country_forecast_df <- function(df, country_name, best_model) {
  country_data <- df %>%
    filter(Country == country_name) %>%
    arrange(Year)

  ts_data <- ts(country_data$`Annual CO2 emissions (tonnes)`, start =
min(country_data$Year), frequency = 1)
  h <- 2030 - 2023

  # Target
  fit_for_55 <- get_target_value(ts_data)

  # Re-Fit models
  if (startsWith(best_model, "ETS")) {
```

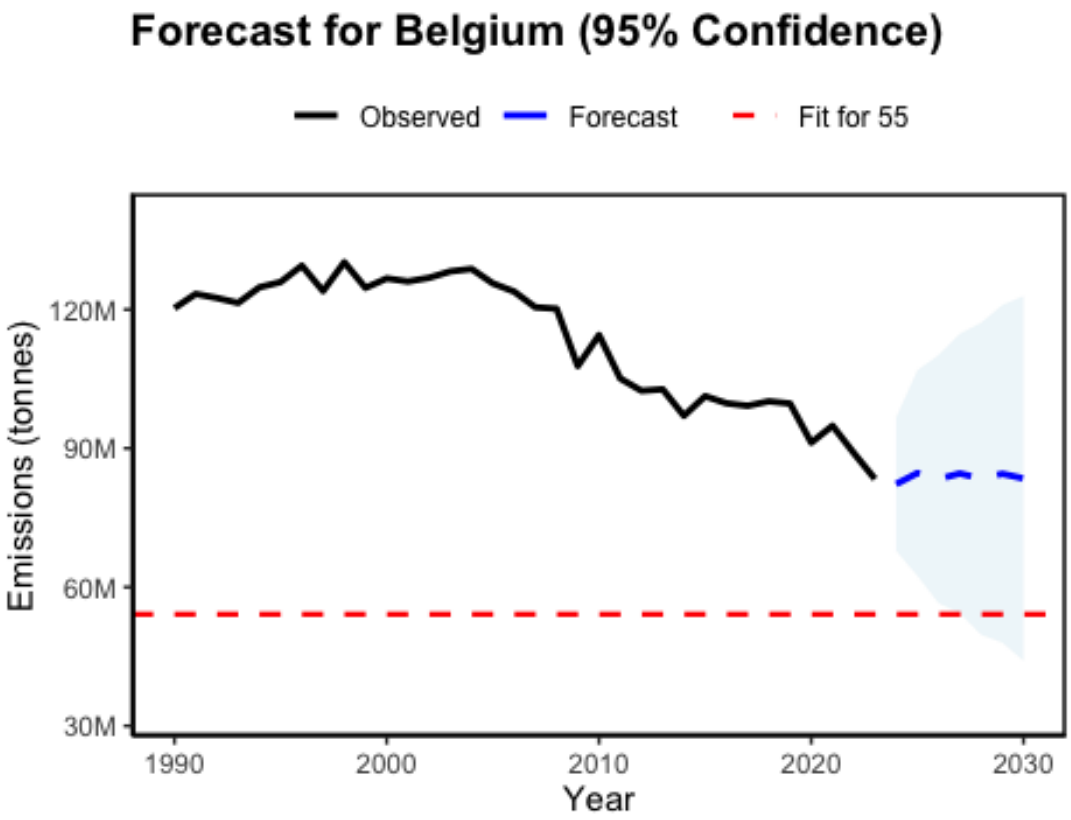
```
    fit <- ets(ts_data)
    if (grepl("ETS\\(A,N,N\\)", fit$method)) {
      fit <- ets(ts_data, model = "AAN")
    }
    fc <- forecast(fit, h = h, level = 95)
  } else if (startsWith(best_model, "ARIMA")) {
    drift <- grepl("\\*drift$", best_model)
    order_vals <- gsub("ARIMA\\(|\\)|\\+ drift", "", best_model)
    order_vec <- as.integer(strsplit(order_vals, ",")[[1]])
    fit <- Arima(ts_data, order = order_vec, include.drift = drift)
    fc <- forecast(fit, h = h, level = 95)
  }

  # Observed data
  observed_df <- tibble(
    Year = as.integer(time(ts_data)),
    Emissions = as.numeric(ts_data),
    Forecast = FALSE,
    Lower = NA,
    Upper = NA,
    Country = country_name,
    Fit55 = fit_for_55
  )

  # Forecasted data with 95% intervals
  forecast_df <- tibble(
    Year = 2024:2030,
    Emissions = as.numeric(fc$mean),
    Lower = as.numeric(fc$lower[, 1]),
    Upper = as.numeric(fc$upper[, 1]),
    Forecast = TRUE,
    Country = country_name,
    Fit55 = fit_for_55
  )

  # Join observed and forecasted values for each country
  bind_rows(observed_df, forecast_df)
}

forecast_plot_data <- pmap_dfr(
  list(
    country_name = results$Country,
    best_model = results$Best_Model
  ),
  ~generate_country_forecast_df(data, ..1, ..2)
)

plot_country_forecast <- function(country_name, save = FALSE) {
  df_country <- forecast_plot_data %>% filter(Country == country_name)
```

```r
  p <- ggplot(df_country, aes(x = Year, y = Emissions)) +
    geom_ribbon(
      data = df_country %>% filter(Forecast == TRUE),
      aes(x = Year, ymin = Lower, ymax = Upper),
      fill = "lightblue", alpha = 0.2
    ) +
    geom_line(aes(color = Forecast, linetype = Forecast), size = 1) +
geom_hline(aes(yintercept = Fit55, linetype = "Fit for 55"), color = "red",
linewidth = 0.8) +
    scale_color_manual(values = c("FALSE" = "black", "TRUE" = "blue"),
                       labels = c("Observed", "Forecast")) +
    scale_linetype_manual(values = c("FALSE" = "solid", "TRUE" = "dashed",
"Fit for 55" = "dashed"), breaks = c("Observed", "Forecast", "Fit for 55")) +
    scale_x_continuous(limits = c(1990, 2030)) +
    scale_y_continuous(labels = label_number(scale_cut = cut_short_scale()))
+
    labs(
      title = paste0("Forecast for ", country_name, " (95% Confidence)"),
      x = "Year",
      y = "Emissions (tonnes)",
      color = "",
      linetype = ""
    ) +
    theme_classic() +
    theme(
      panel.border = element_rect(color = "black", fill = NA, linewidth = 1),
      legend.position = "top",
      legend.justification = "center",
      legend.direction = "horizontal",
      legend.title = element_blank(),
      legend.text = element_text(size = 9),
      legend.spacing.x = unit(5, "pt"),
      plot.title = element_text(hjust = 0.5, size = 14, face = "bold"),
      plot.title.position = "plot",
      plot.margin = margin(t = 20, r = 10, b = 10, l = 10)
    )

  return(p)
}

# Print chart for every country
for (country in country_list) {
  print(plot_country_forecast(country))
}
```

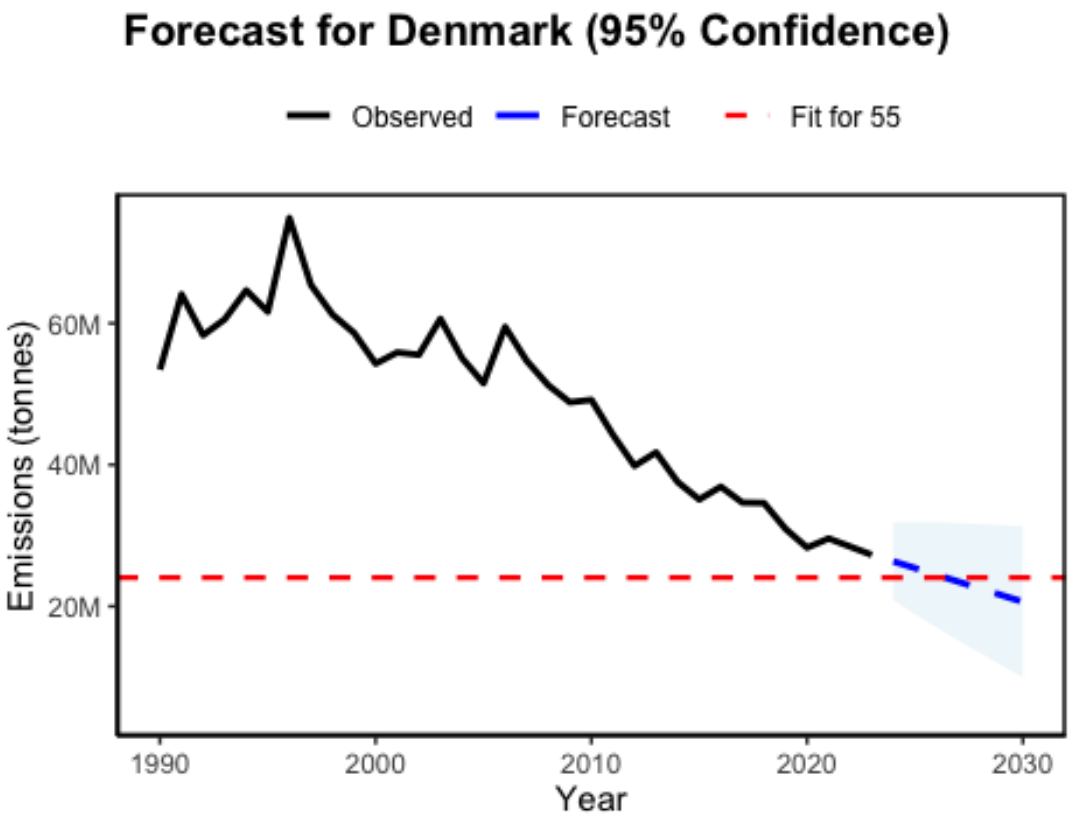
```
## Warning: Removed 90 rows containing missing values or values outside the
scale range
## (`geom_line()`).
```



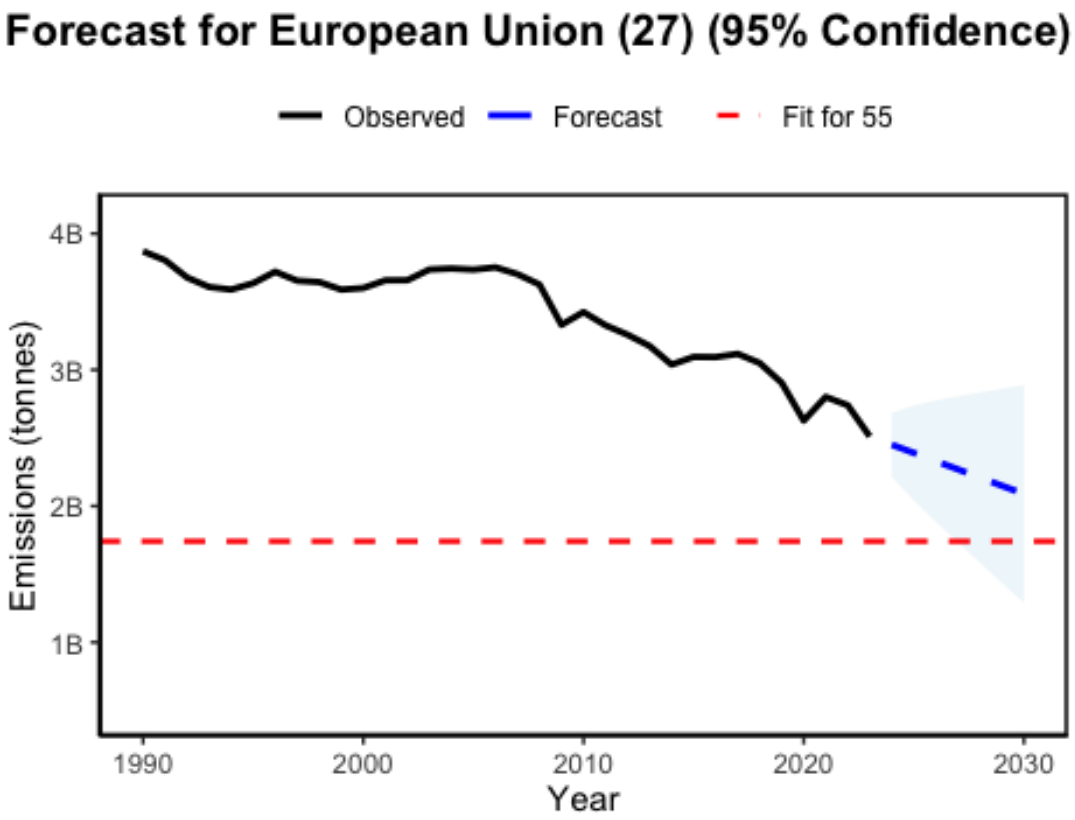

```
## Warning: Removed 90 rows containing missing values or values outside the
scale range
## (`geom_line()`).
```



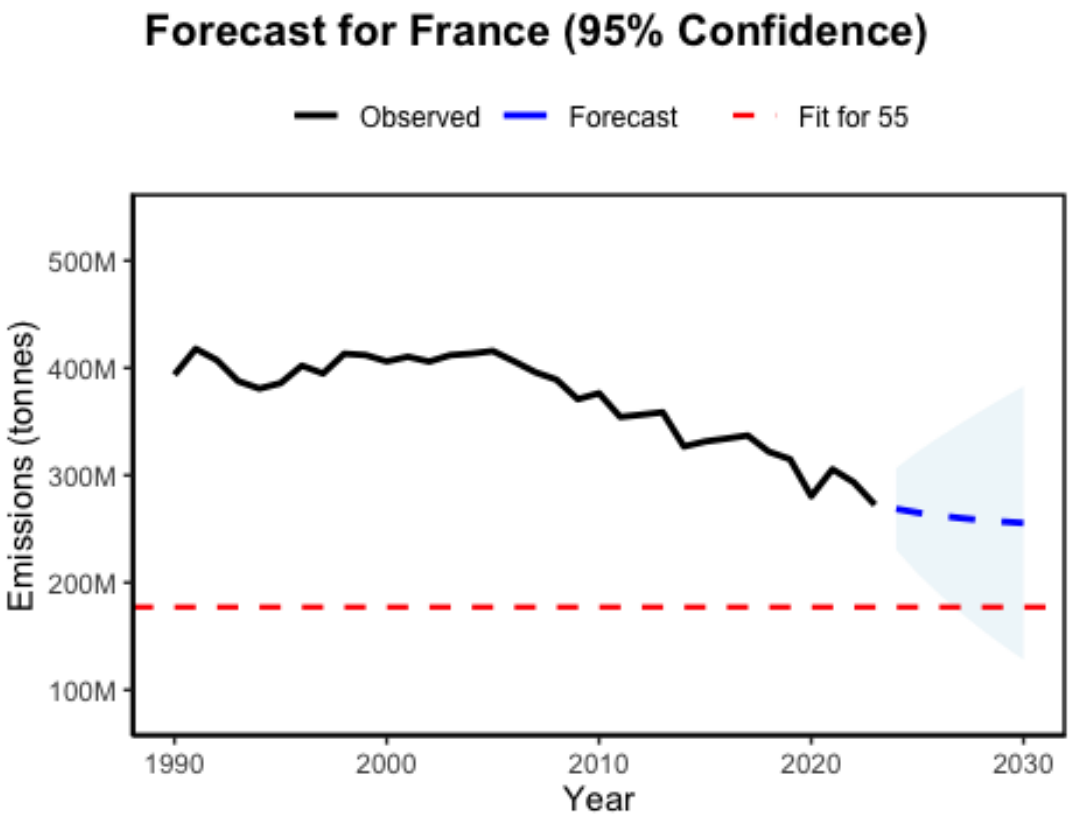

```
## Warning: Removed 90 rows containing missing values or values outside the
scale range
## (`geom_line()`).
```



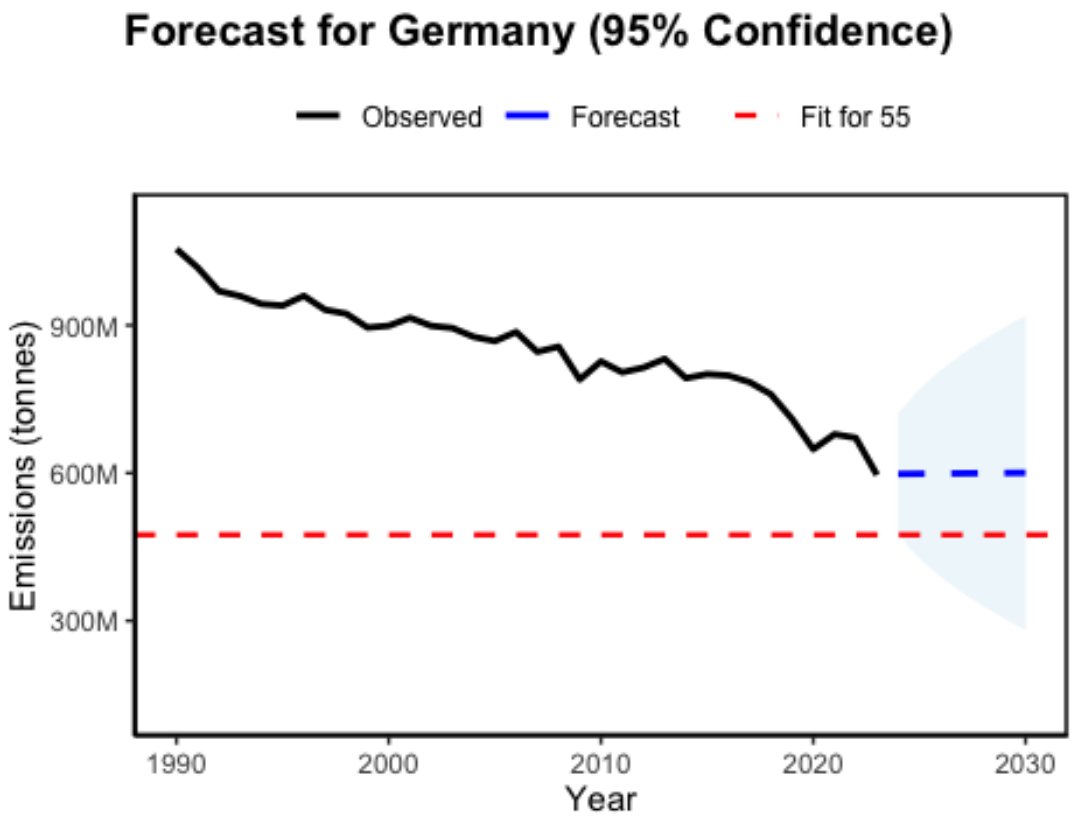

```
## Warning: Removed 90 rows containing missing values or values outside the
scale range
## (`geom_line()`).
```



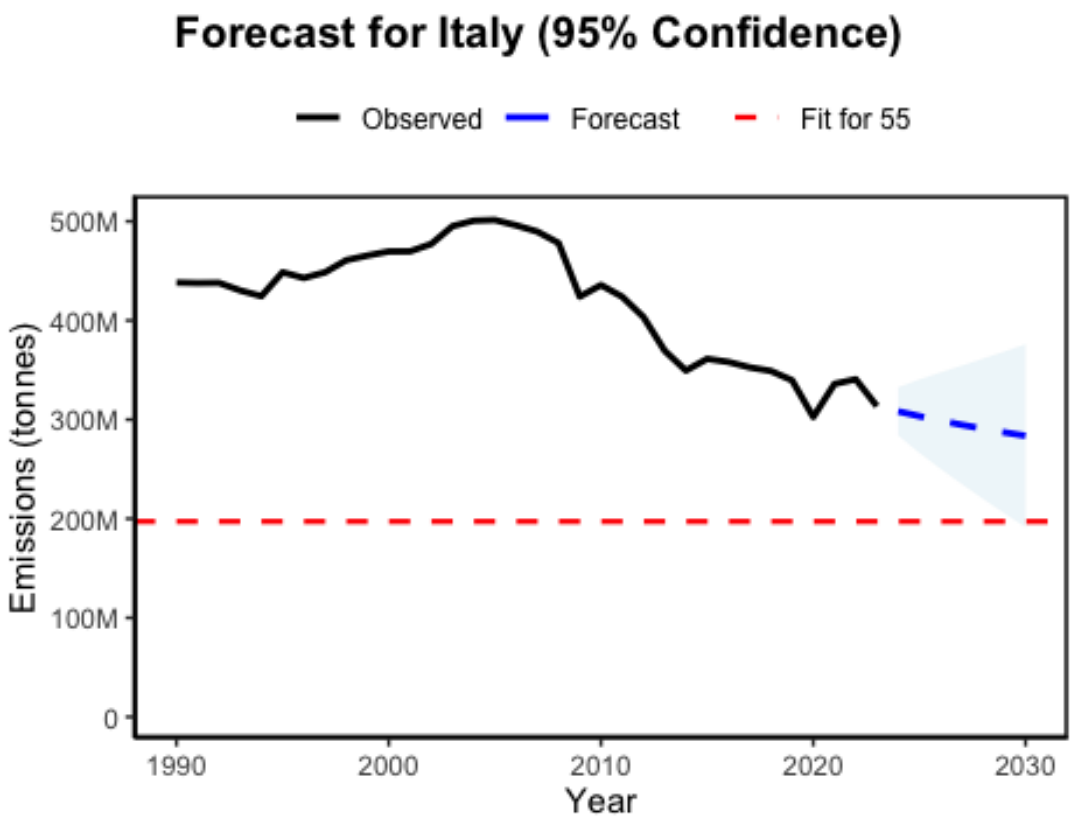

```
## Warning: Removed 90 rows containing missing values or values outside the
scale range
## (`geom_line()`).
```



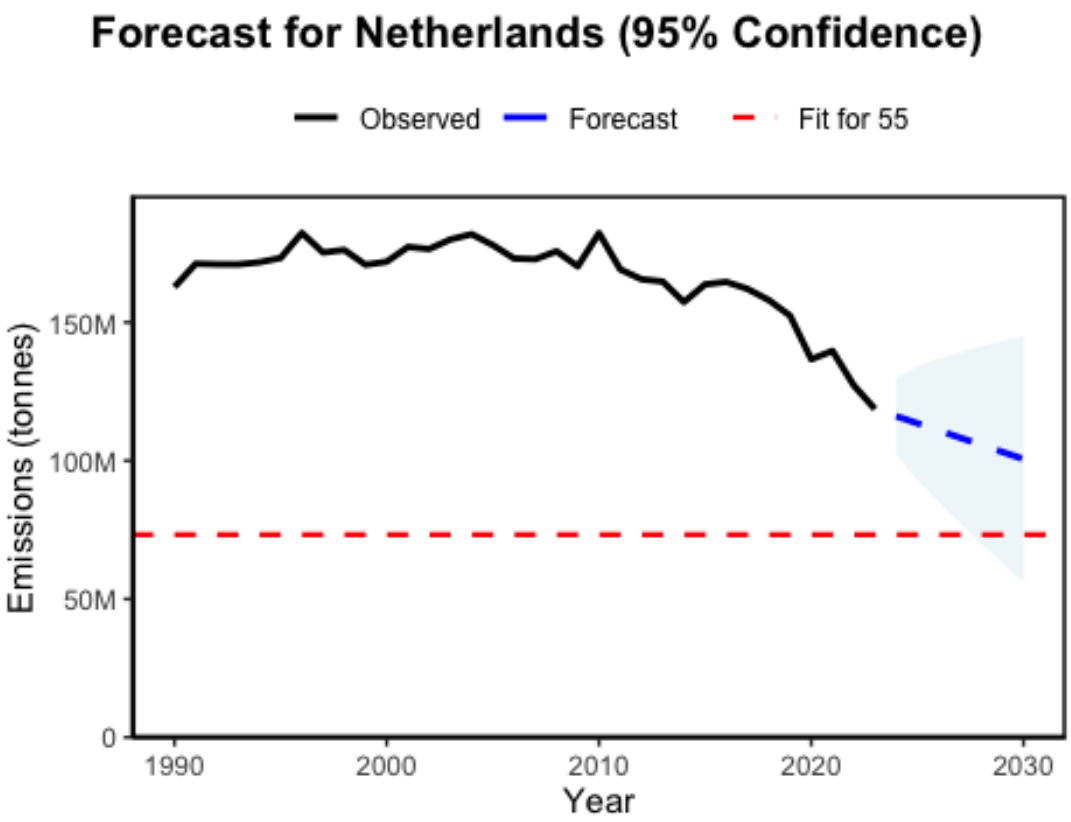

```
## Warning: Removed 90 rows containing missing values or values outside the
scale range
## (`geom_line()`).
```



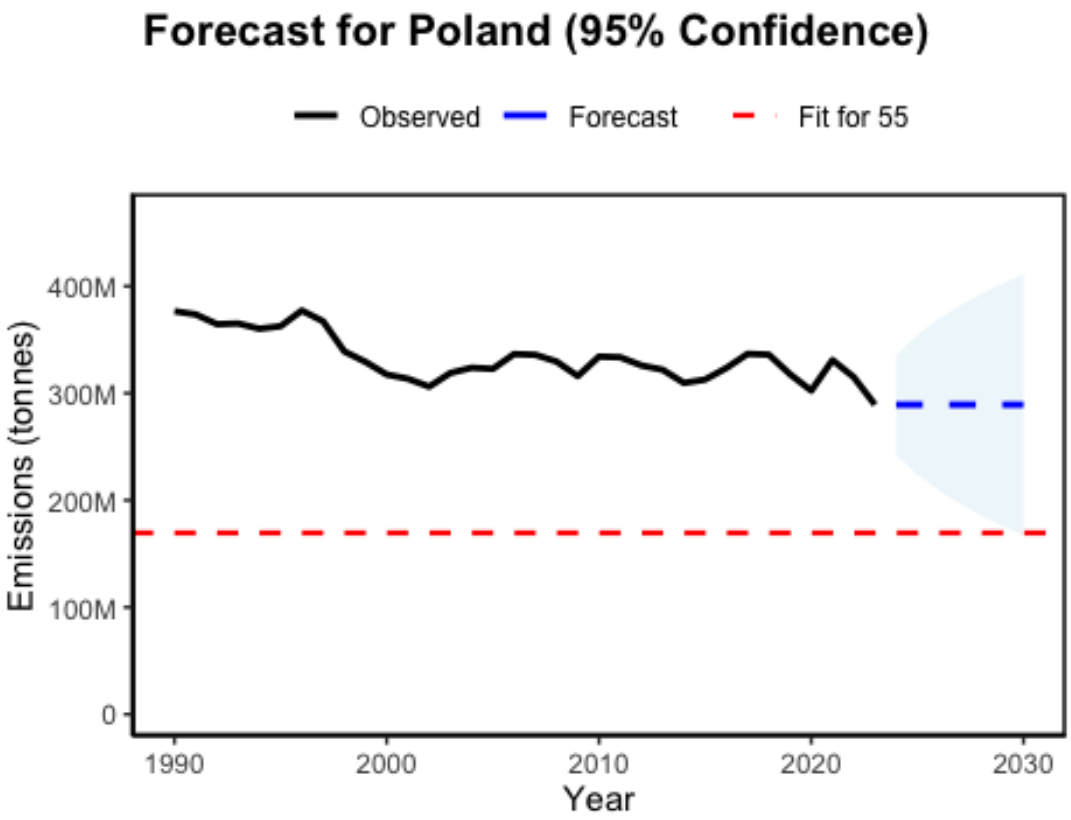

```
## Warning: Removed 90 rows containing missing values or values outside the
scale range
## (`geom_line()`).
```



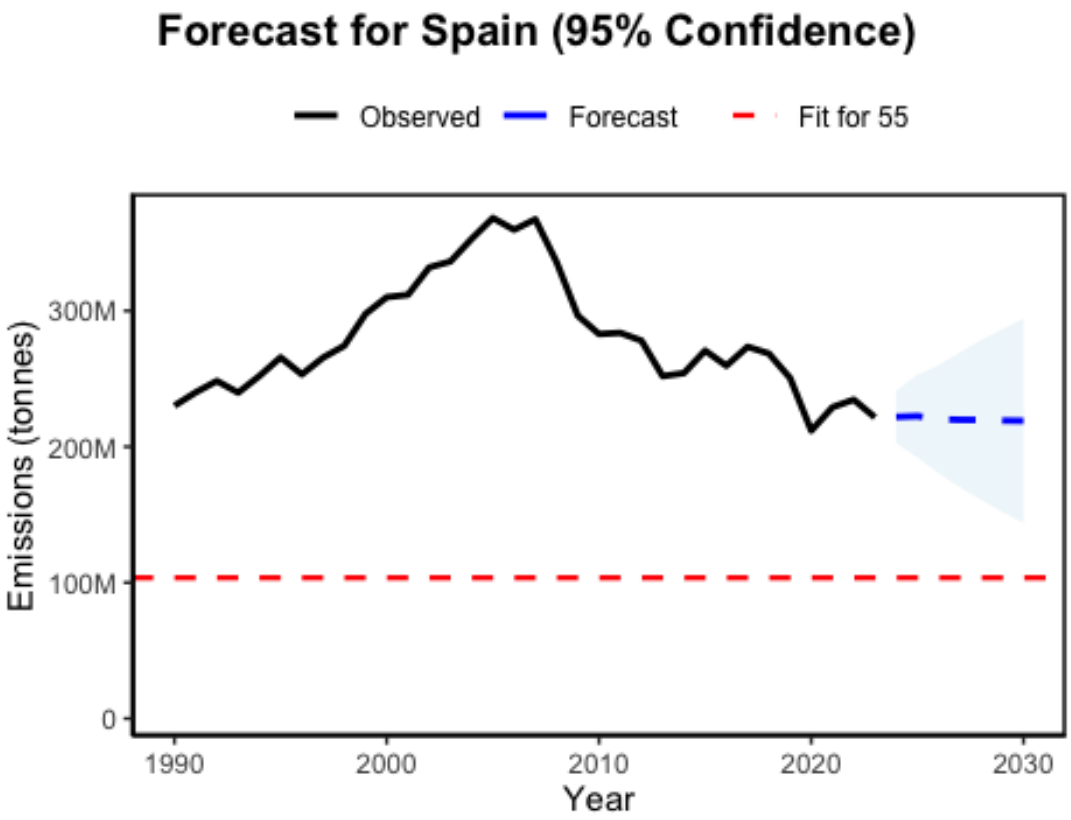

```
## Warning: Removed 90 rows containing missing values or values outside the
scale range
## (`geom_line()`).
```



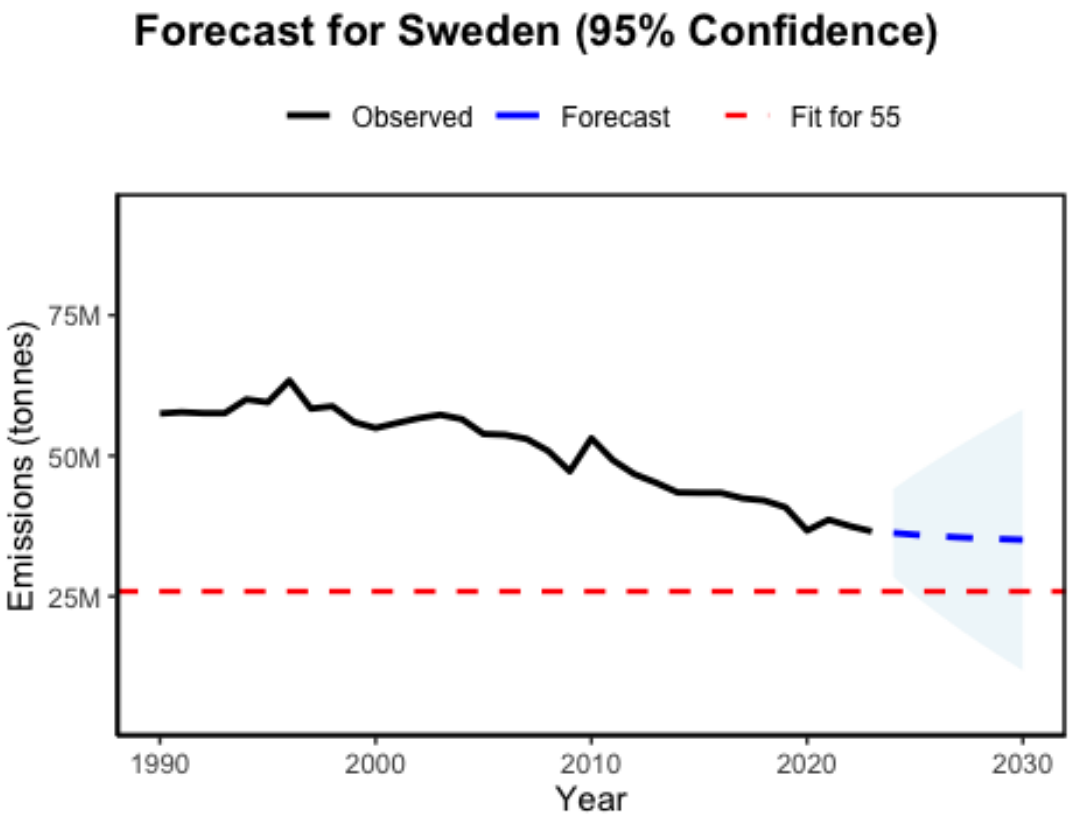

```
## Warning: Removed 90 rows containing missing values or values outside the
scale range
## (`geom_line()`).
```



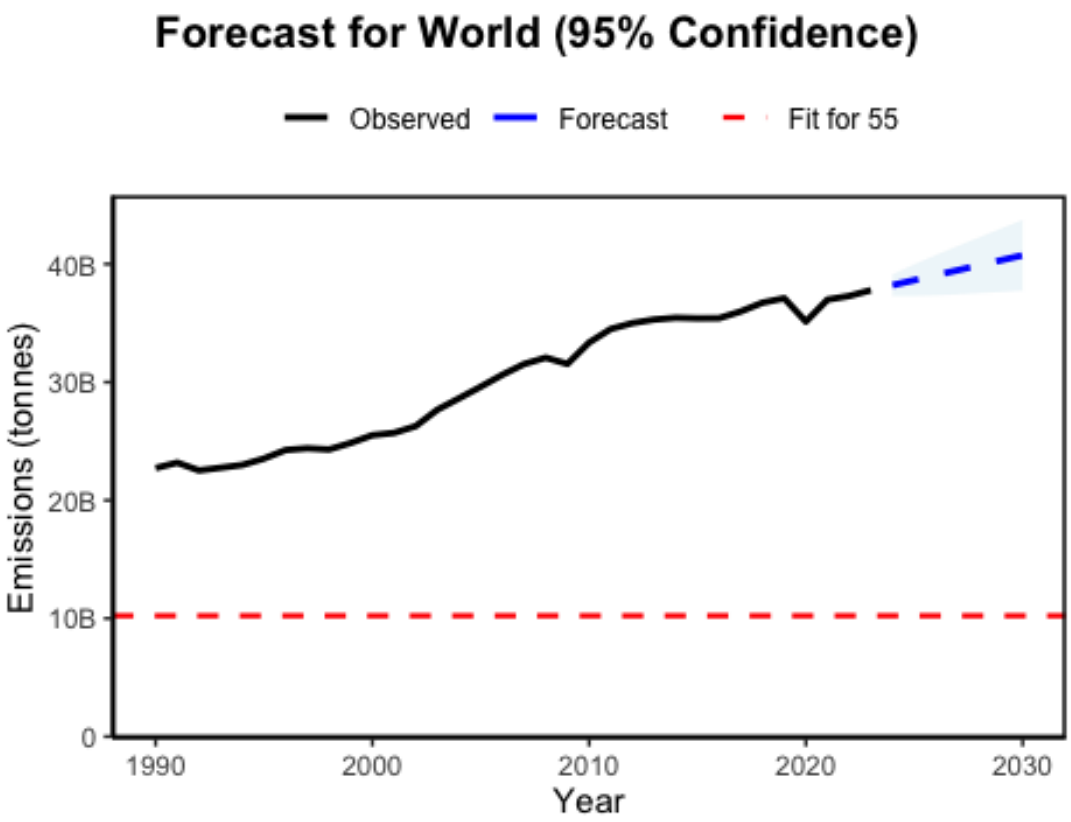

```
## Warning: Removed 90 rows containing missing values or values outside the
scale range
## (`geom_line()`).
```




```
## Warning: Removed 90 rows containing missing values or values outside the
scale range
## (`geom_line()`).
```




```
## Warning: Removed 90 rows containing missing values or values outside the
scale range
## (`geom_line()`).
```


Forecast for World (95% Confidence)
Observed
Forecast
Fit for 55
Emissions (tonnes)
40B
30B
20B
10B
0
1990
2000
2010
2020
2030
Year