Time_series

Krishna Poudel

2024-04-15

```
#Loading libraries and time series data (.csv file)
library(tidyverse)#For data manipulation
```

```
## Warning: package 'tidyverse' was built under R version 4.3.3
```

```
## Warning: package 'ggplot2' was built under R version 4.3.2
```

```
## Warning: package 'lubridate' was built under R version 4.3.2
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ dplyr     1.1.2     ✓ readr     2.1.4
## ✓ forcats   1.0.0     ✓ stringr   1.5.0
## ✓ ggplot2   3.4.4     ✓ tibble    3.2.1
## ✓ lubridate 1.9.3     ✓ tidyr     1.3.0
## ✓ purrr     1.0.1
## ── 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 becom
e errors
```

```
library(forecast) # For forecasting of time series analysis
```

```
## Warning: package 'forecast' was built under R version 4.3.3
```

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

```
library(zoo)      # For time series analysis
```

```
## Warning: package 'zoo' was built under R version 4.3.3
```

```
##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
library(ggplot2)  #For data visualisation

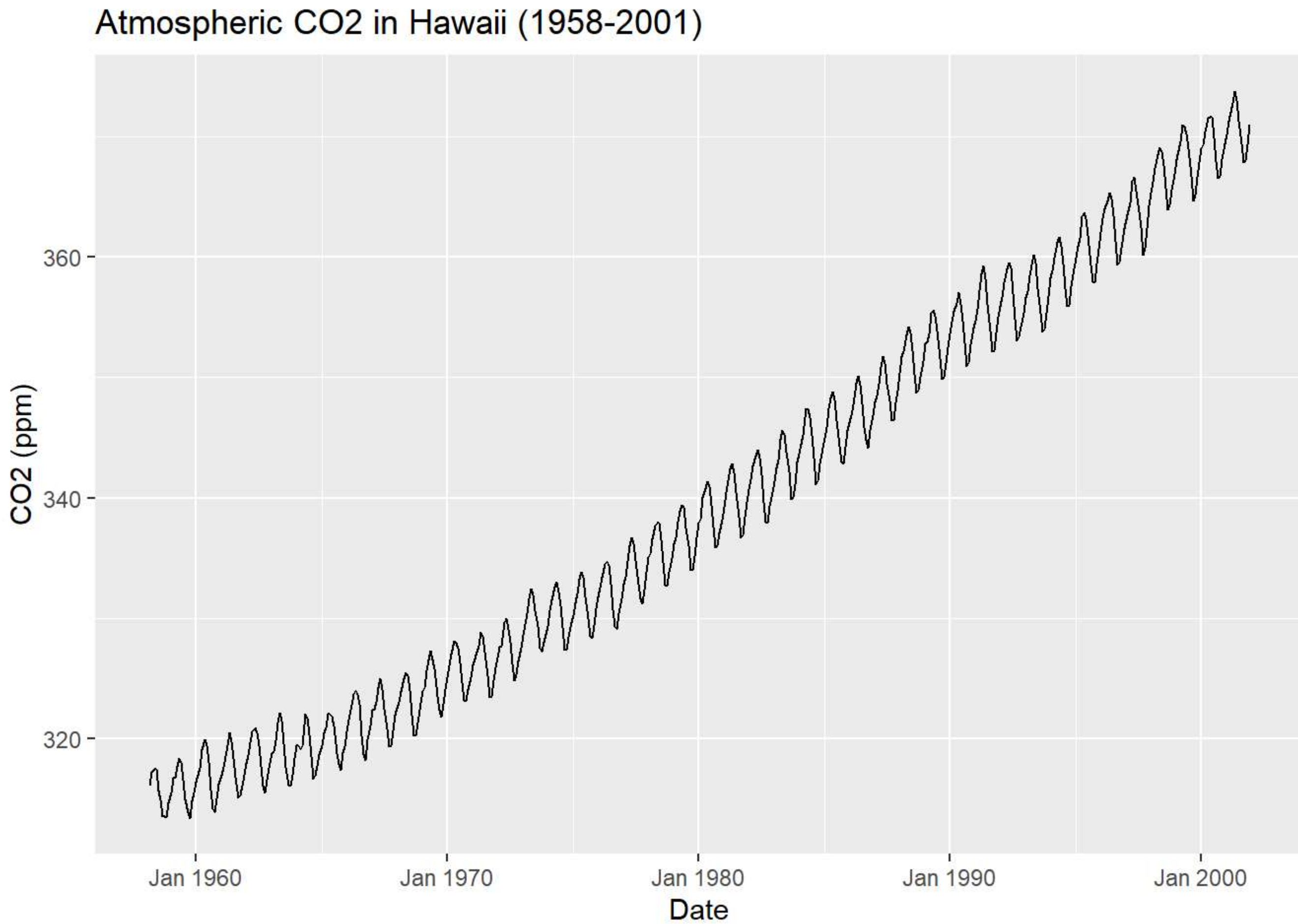
data <- read.csv("hawai.csv")

#Checking the structure of data
str(data)
```

```
## 'data.frame':    526 obs. of  2 variables:
##  $ time: num  1958 1958 1958 1958 1958 ...
##  $ CO2 : num  316 317 317 317 316 ...
```

```
#Step.2.Conversion to time series format
#Conversion to distinct format (YYYY-MM)
data$date <- as.yearmon(data$time, "%Y.%m")

#Data visualisation of time series plot
ggplot(data, aes(x = date, y = CO2)) + geom_line() +
  labs(title = "Atmospheric CO2 in Hawaii (1958-2001)", x = "Date",
  y = "CO2 (ppm)")
```




```
# Step 3. Split data into training and test sets (70% training)
train_size <- floor(nrow(data) * 0.7)
train_data <- data[1:train_size, ]
test_data <- data[(train_size + 1):nrow(data), ]

#Inspecting the data
head(train_data)
```

```
##       time      CO2     date
## 1 1958.167 316.1000 Mar 1958
## 2 1958.250 317.2000 Apr 1958
## 3 1958.333 317.4333 May 1958
## 4 1958.417 317.4333 Jun 1958
## 5 1958.500 315.6250 Jul 1958
## 6 1958.583 314.9500 Aug 1958
```

```
tail(test_data)
```

```
##         time     CO2     date
## 521 2001.500 371.300 Jul 2001
## 522 2001.583 369.425 Aug 2001
## 523 2001.667 367.880 Sep 2001
## 524 2001.750 368.050 Oct 2001
## 525 2001.833 369.375 Nov 2001
## 526 2001.917 371.020 Dec 2001
```

```
#checking if the test is good
nrow(train_data) / nrow(test_data) >= 0.7
```

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

```
#Checking row counts
nrow(train_data) + nrow(test_data) == nrow(data)
```

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

```
#Step.4.Fitting in Seasonal ARIMA model
model <- auto.arima(train_data$CO2)

# Forecasting for the test period
forecasts <- forecast(model, h = nrow(test_data))
plot_data <- data.frame(date = test_data$date, actual = test_data$CO2,
forecast = forecasts$mean)



# Evaluate the Model

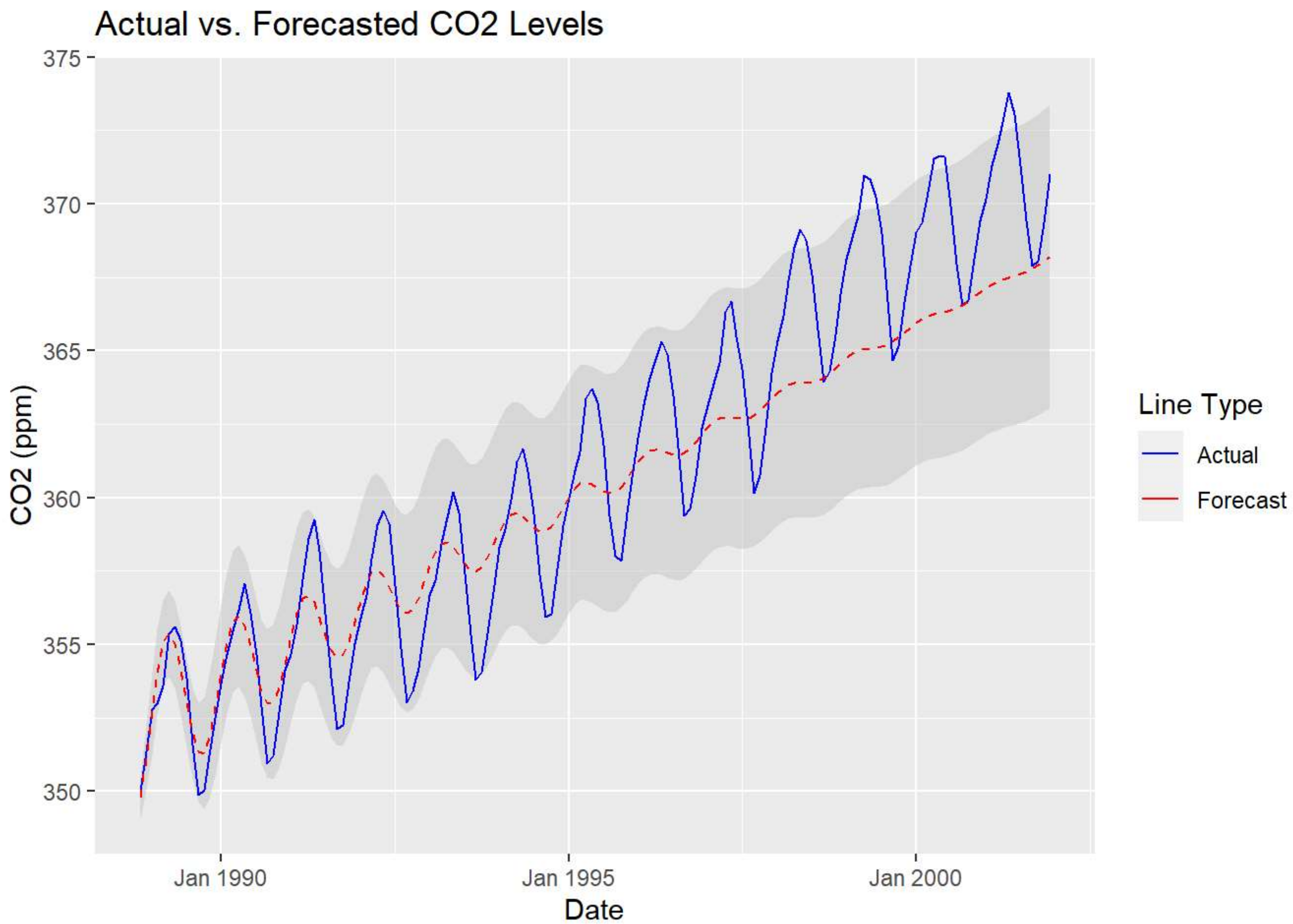
comparison <- data.frame(Actual = test_data$CO2, Forecasted = forecasts$mean)

accuracy_metrics <- accuracy(forecasts, test_data$CO2)

# Plotting forecast value vs actual value
plot_data <- data.frame(
  date = test_data$date,
  actual = test_data$CO2,
  forecast = forecasts$mean,
  lower = forecasts[["lower"]][, "80%"],
  upper = forecasts[["upper"]][, "80%"])

ggplot(data = plot_data, aes(x = date)) +
  geom_ribbon(aes(ymin = lower, ymax = upper), fill = "grey80", alpha = 0.5) +
  geom_line(aes(y = actual, color = "Actual"), linetype = "solid") +
  geom_line(aes(y = forecast, color = "Forecast"), linetype = "dashed") +
  labs(title = "Actual vs. Forecasted CO2 Levels", x = "Date", y = "CO2 (ppm)",
 color = "Line Type") + scale_color_manual(values =
  c("Actual" = "blue", "Forecast" = "red"))
```

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




```
```r
# Step 5. Residual analysis

residuals <- test_data$CO2 - forecasts$mean

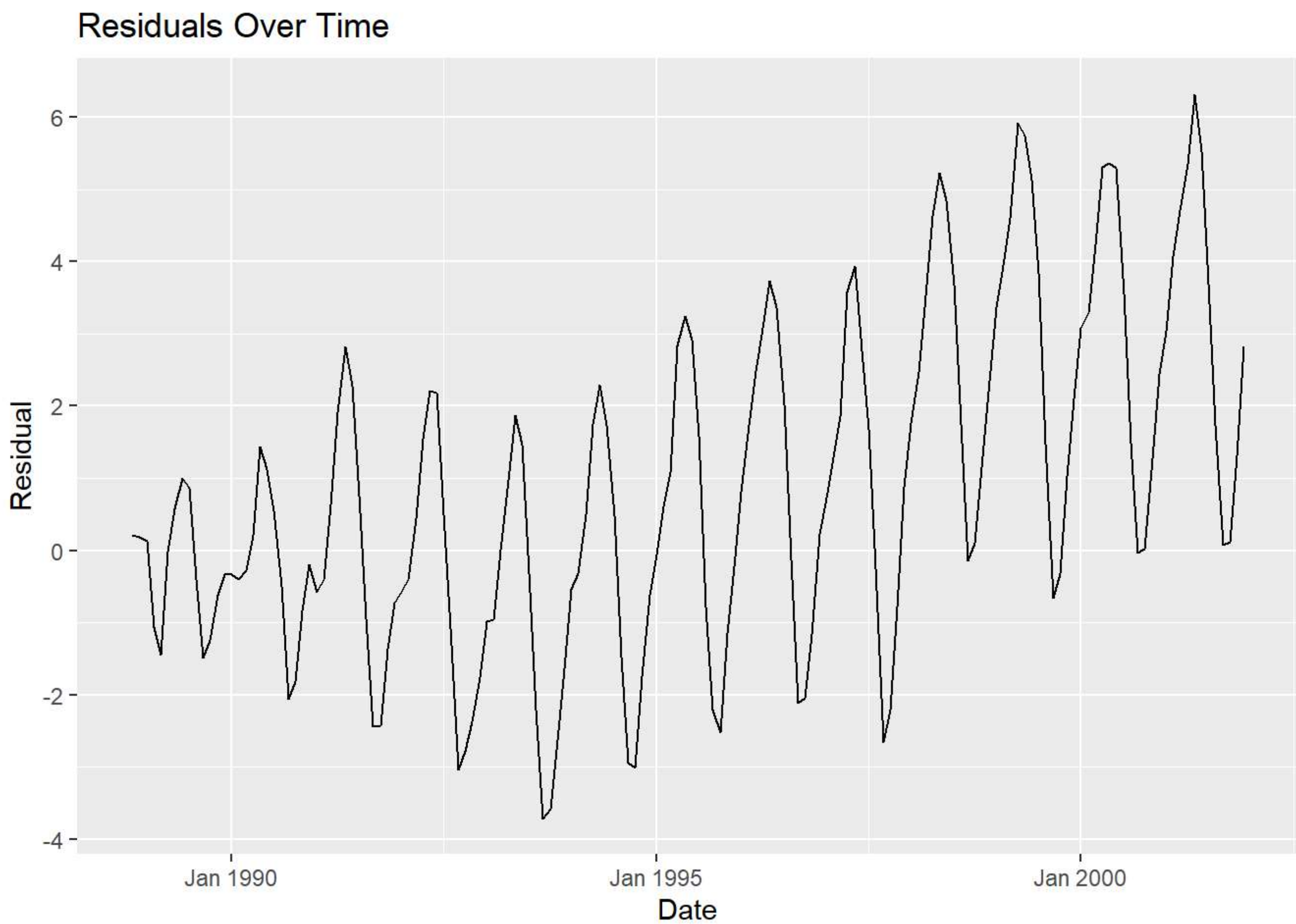
Residual_df <- data.frame(Date = test_data$date, Residuals = residuals)

#Checking for normality of residuals
shapiro.test(residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals
## W = 0.98024, p-value = 0.02307
```
```

```
#Plotting residuals vs time
ggplot(Residual_df, aes(x = Date, y = Residuals)) + geom_line() +
  labs(title = "Residuals Over Time", x = "Date", y = "Residual")
```

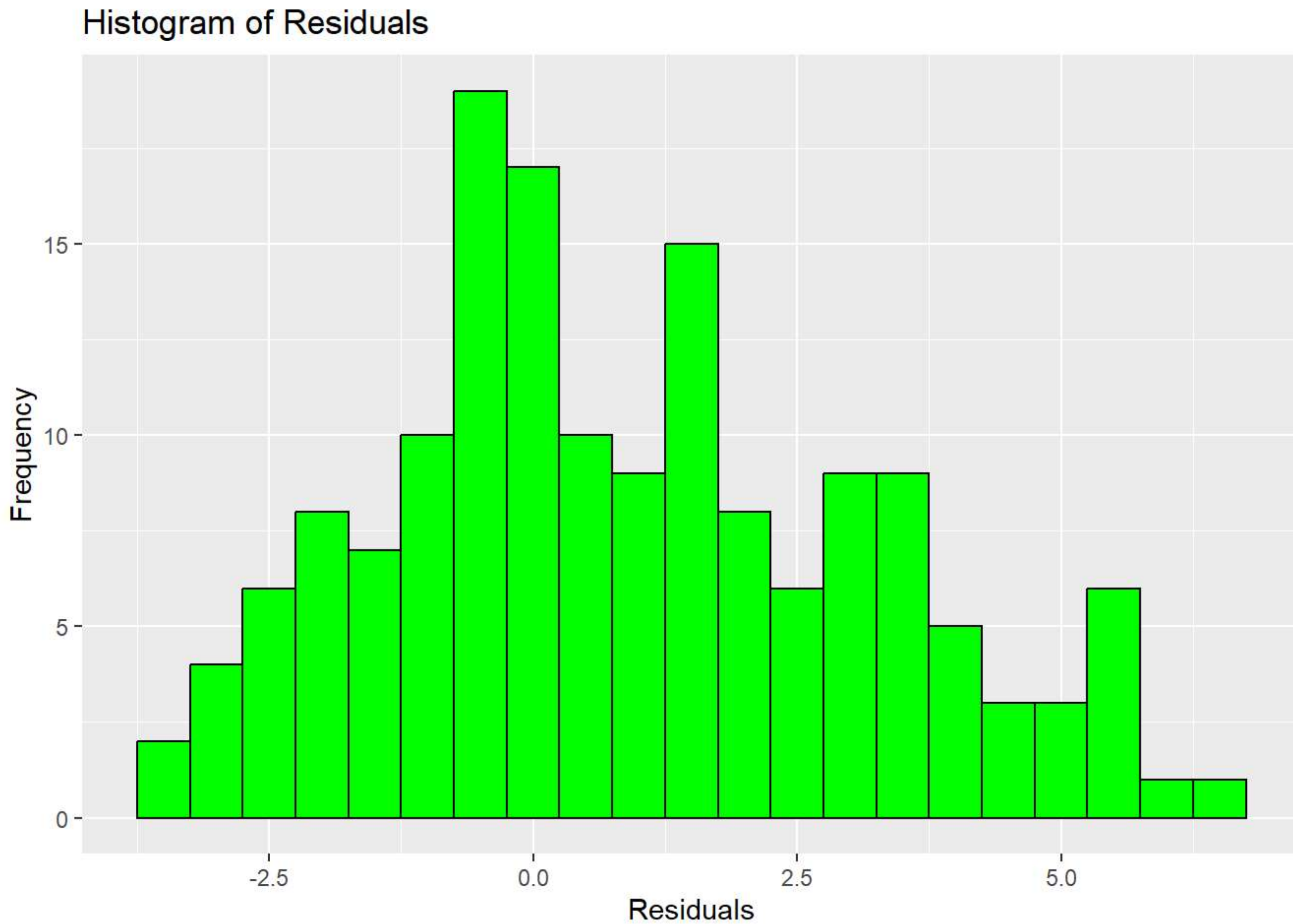
```
## Don't know how to automatically pick scale for object of type <ts>. Defaulting
## to continuous.
```




```
#Checking for normality of residual distribution

ggplot(Residual_df, aes(x = Residuals)) +
  geom_histogram(binwidth = 0.5, fill = "Green", color = "black") +
  labs(title = "Histogram of Residuals", x = "Residuals", y = "Frequency")
```

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



Histogram of Residuals
Frequency
0
5
10
15
-2.5
0.0
2.5
5.0
Residuals