```
library(reprex)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(fredr)
library(ggplot2)
library(forecast)
```

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

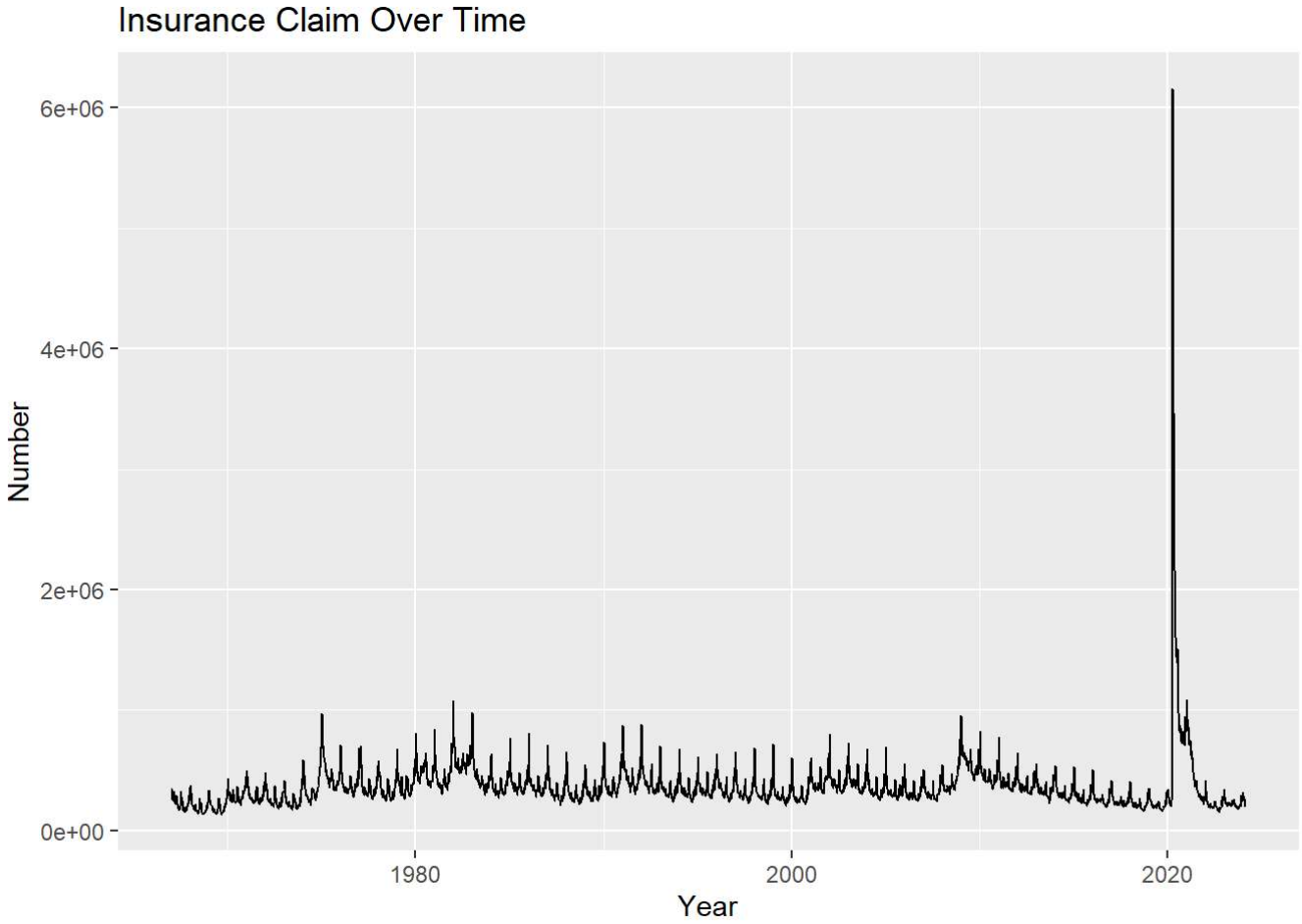
```
## Registered S3 methods overwritten by 'forecast':
##   method       from
##   fitted.Arima TSA
##   plot.Arima   TSA
```

```
library(tseries)
library(urca)
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ forcats   1.0.0     ✓ stringr   1.5.1
## ✓ lubridate 1.9.3     ✓ tibble    3.2.1
## ✓ purrr     1.0.2     ✓ tidyr     1.3.1
## ✓ readr     2.1.5
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ✖ readr::spec()   masks TSA::spec()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
e errors
```

```
library(readxl)
library(forecast)

fredr_set_key("360481124fc765b815de2697f1bf8d62")

icnsa <- fredr(series_id = "ICNSA")

icnsa$date <- as.Date(icnsa$date)
ggplot(icnsa, aes(x = date, y = value)) + geom_line() + labs(title = "Insurance Claim Over Tim
e",x = "Year",y = "Number")
```

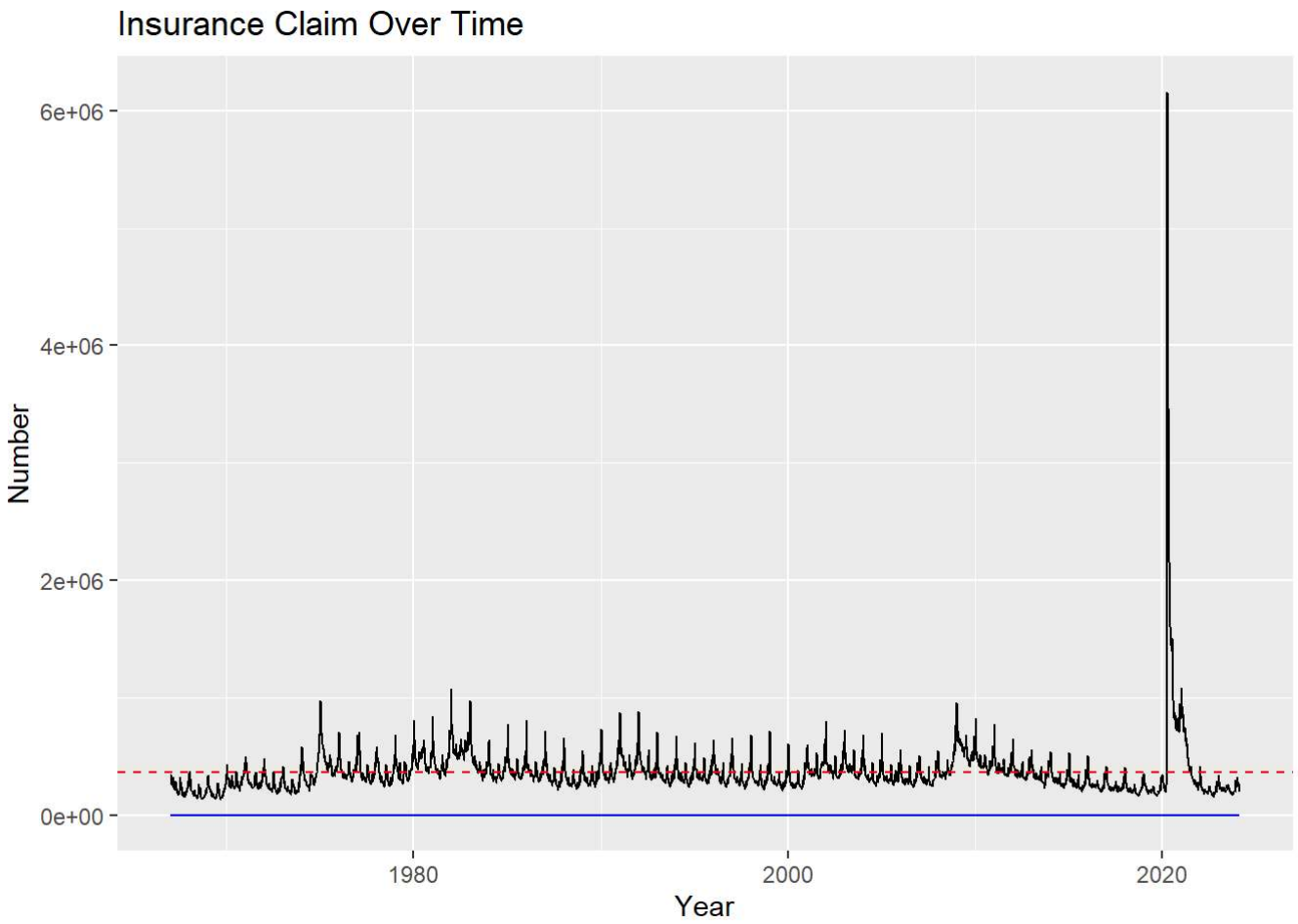



```
str(icnsa)
```

```
## tibble [2,981 × 5] (S3: tbl_df/tbl/data.frame)
##  $ date          : Date[1:2981], format: "1967-01-07" "1967-01-14" ...
##  $ series_id     : chr [1:2981] "ICNSA" "ICNSA" "ICNSA" "ICNSA" ...
##  $ value         : num [1:2981] 346000 334000 277000 252000 274000 276000 247000 248000 32600
0 240000 ...
##  $ realtime_start: Date[1:2981], format: "2024-02-28" "2024-02-28" ...
##  $ realtime_end  : Date[1:2981], format: "2024-02-28" "2024-02-28" ...
```

```
lower_limit <- 100
upper_limit <- 200
mean_value <- mean(icnsa$value)
# Filter the data within the range defined by the blue lines
icnsa_filtered <- icnsa %>%
  filter(date >= min(date) & date <= max(date), value >= lower_limit & value <= upper_limit)

# Calculate mean and standard deviation for the filtered data
mean_value_filtered <- mean(icnsa_filtered$value)
sd_value_filtered <- sd(icnsa_filtered$value)

# Plot for line representing the mean value and blue lines for the lower and upper limits of the
standard deviation
ggplot(icnsa, aes(x = date, y = value)) +
  geom_line() +
  geom_hline(yintercept = mean_value, linetype = "dashed", color = "red") +
  geom_segment(aes(x = min(date), y = lower_limit, xend = max(date), yend = lower_limit), color
= "blue") +
  geom_segment(aes(x = min(date), y = upper_limit, xend = max(date), yend = upper_limit), color
= "blue") +
  geom_hline(yintercept = mean_value_filtered, linetype = "dashed", color = "green") +  # Add me
an line for filtered data
  labs(title = "Insurance Claim Over Time", x = "Year", y = "Number")
```

```
## Warning: Removed 1 rows containing missing values (`geom_hline()`).
```




```
# Filter the data within the range defined by the blue lines
icnsa_filtered <- icnsa %>%
  filter(date >= min(date) & date <= max(date), value >= lower_limit & value <= upper_limit)

# Mean and standard deviation for the filtered data
mean_value_filtered <- mean(icnsa_filtered$value)
sd_value_filtered <- sd(icnsa_filtered$value)

print(paste("Mean value between the blue lines:", mean_value_filtered))
```

```
## [1] "Mean value between the blue lines: NaN"
```

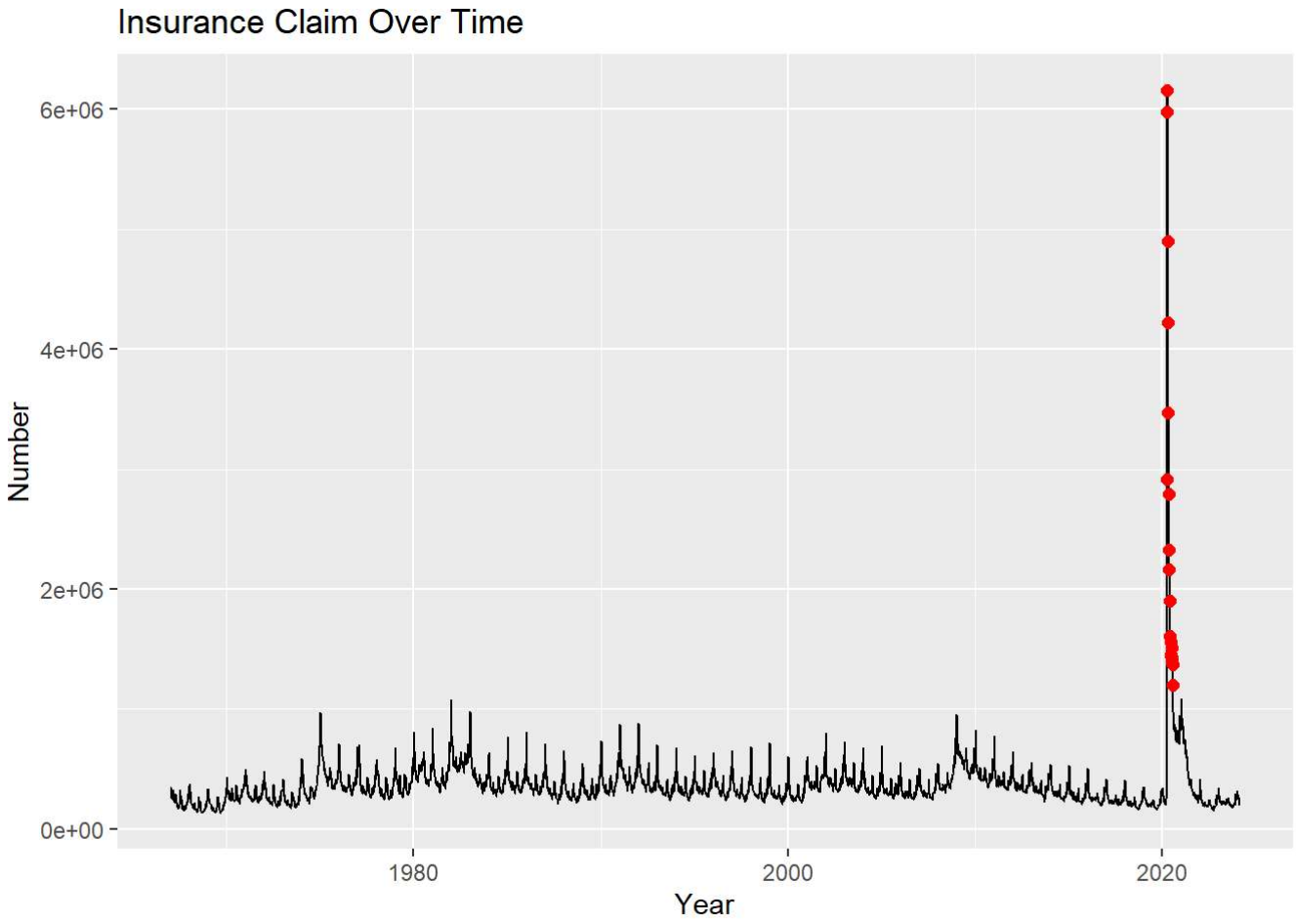
```
# All points above the higher limit of the standard deviation
above_limit_points <- icnsa %>%
  filter(value > upper_limit)

# Print the points
print(above_limit_points)
```

```
## # A tibble: 2,981 × 5
##    date       series_id  value realtime_start realtime_end
##    <date>     <chr>      <dbl> <date>         <date>
##  1 1967-01-07 ICNSA     346000 2024-02-28     2024-02-28
##  2 1967-01-14 ICNSA     334000 2024-02-28     2024-02-28
##  3 1967-01-21 ICNSA     277000 2024-02-28     2024-02-28
##  4 1967-01-28 ICNSA     252000 2024-02-28     2024-02-28
##  5 1967-02-04 ICNSA     274000 2024-02-28     2024-02-28
##  6 1967-02-11 ICNSA     276000 2024-02-28     2024-02-28
##  7 1967-02-18 ICNSA     247000 2024-02-28     2024-02-28
##  8 1967-02-25 ICNSA     248000 2024-02-28     2024-02-28
##  9 1967-03-04 ICNSA     326000 2024-02-28     2024-02-28
## 10 1967-03-11 ICNSA     240000 2024-02-28     2024-02-28
## # ℹ 2,971 more rows
```

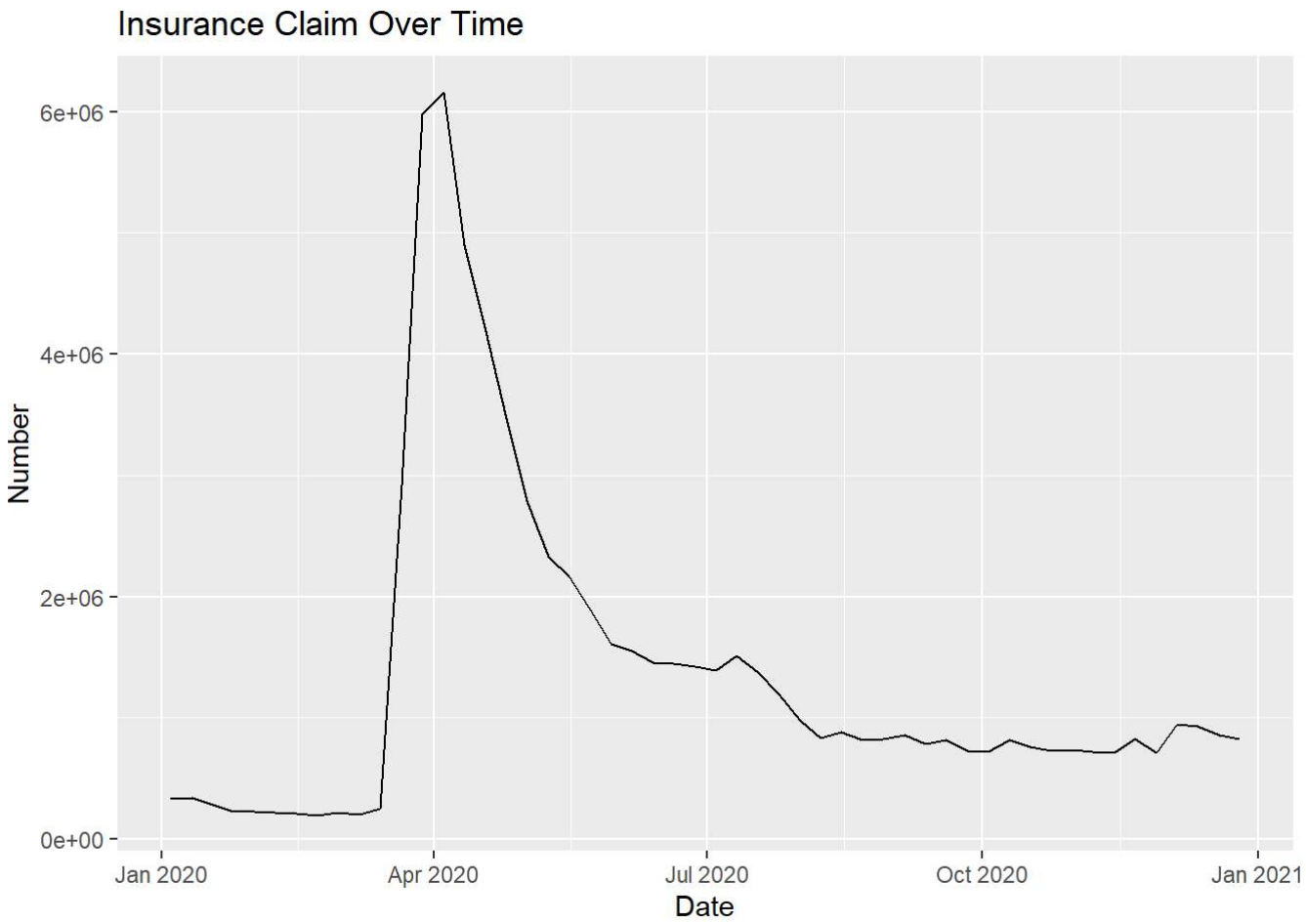
```
# Calculate Z-score
icnsa <- icnsa %>%
  mutate(z_score = abs(value - mean(value)) / sd(value))

# Set threshold for anomaly detection (e.g., 3 standard deviations)
threshold <- 3

# Identify anomalies
anomalies <- icnsa %>%
  filter(z_score > threshold)

# Print detected anomalies
print(anomalies)
```

```
## # A tibble: 19 × 6
##    date       series_id   value realtime_start realtime_end z_score
##    <date>     <chr>       <dbl> <date>         <date>         <dbl>
##  1 2020-03-21 ICNSA     2914268 2024-02-28     2024-02-28     10.1
##  2 2020-03-28 ICNSA     5981838 2024-02-28     2024-02-28     22.2
##  3 2020-04-04 ICNSA     6161268 2024-02-28     2024-02-28     22.9
##  4 2020-04-11 ICNSA     4898119 2024-02-28     2024-02-28     17.9
##  5 2020-04-18 ICNSA     4221704 2024-02-28     2024-02-28     15.2
##  6 2020-04-25 ICNSA     3466665 2024-02-28     2024-02-28     12.2
##  7 2020-05-02 ICNSA     2790860 2024-02-28     2024-02-28      9.58
##  8 2020-05-09 ICNSA     2325889 2024-02-28     2024-02-28      7.74
##  9 2020-05-16 ICNSA     2162822 2024-02-28     2024-02-28      7.10
## 10 2020-05-23 ICNSA     1902006 2024-02-28     2024-02-28      6.07
## 11 2020-05-30 ICNSA     1610909 2024-02-28     2024-02-28      4.92
## 12 2020-06-06 ICNSA     1555824 2024-02-28     2024-02-28      4.70
## 13 2020-06-13 ICNSA     1456353 2024-02-28     2024-02-28      4.31
## 14 2020-06-20 ICNSA     1446176 2024-02-28     2024-02-28      4.27
## 15 2020-06-27 ICNSA     1425969 2024-02-28     2024-02-28      4.19
## 16 2020-07-04 ICNSA     1390315 2024-02-28     2024-02-28      4.05
## 17 2020-07-11 ICNSA     1511632 2024-02-28     2024-02-28      4.53
## 18 2020-07-18 ICNSA     1372247 2024-02-28     2024-02-28      3.98
## 19 2020-07-25 ICNSA     1200962 2024-02-28     2024-02-28      3.30
```

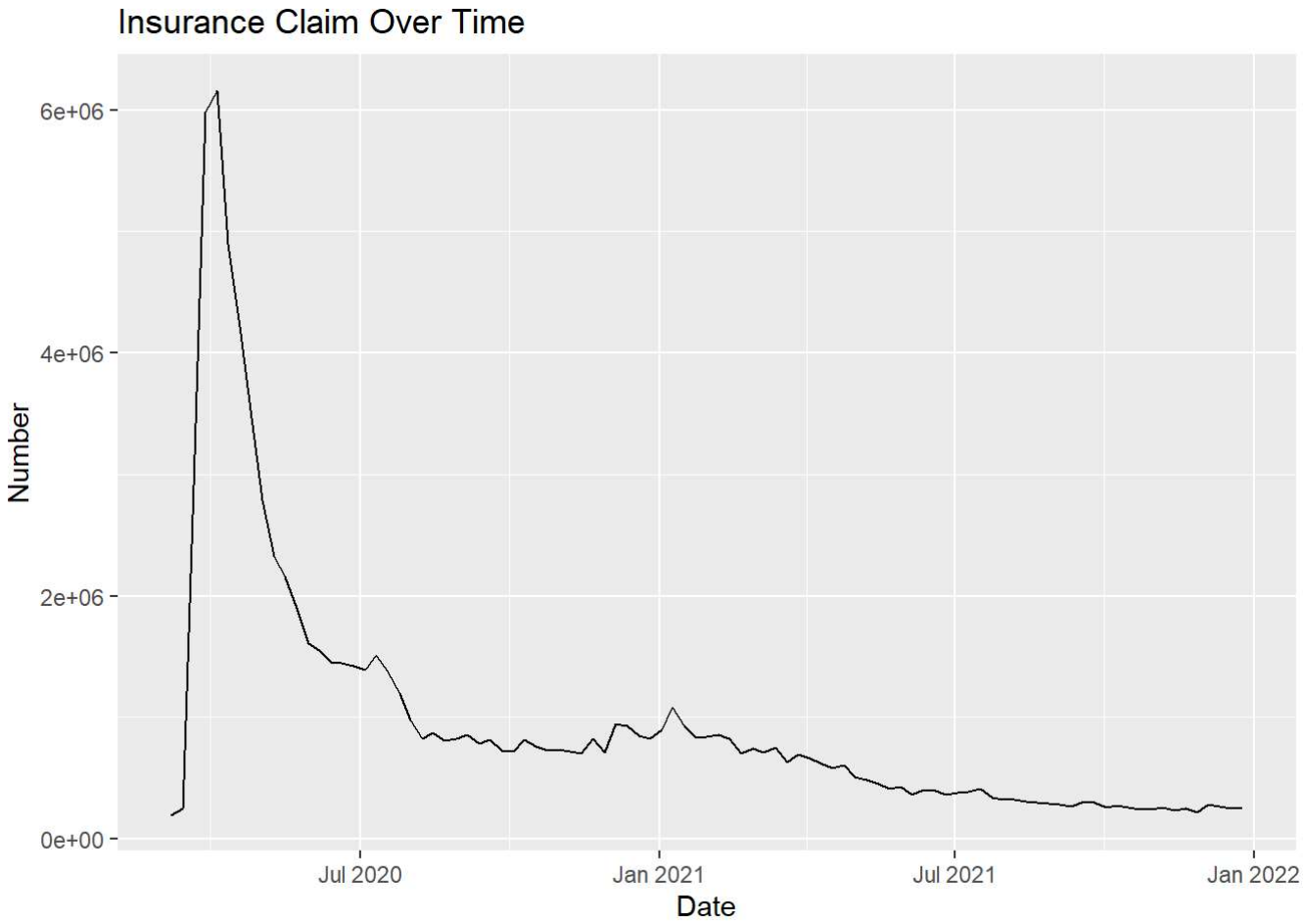
```
library(ggplot2)

# Plot the data with anomalies highlighted
ggplot(icnsa, aes(x = date, y = value)) +
  geom_line() +
  geom_point(data = anomalies, aes(x = date, y = value), color = "red", size = 2) +
  labs(title = "Insurance Claim Over Time", x = "Year", y = "Number")
```




```
start_date <- as.Date("2020-01-01")
end_date <- as.Date("2021-01-01")
# Filter the data for the specified range
filtered_icnsa <- icnsa %>%
  filter(date >= start_date & date <= end_date)

# Plot the filtered data with the specified range on the x-axis
ggplot(filtered_icnsa, aes(x = date, y = value)) +
  geom_line() +
  labs(title = "Insurance Claim Over Time", x = "Date", y = "Number") +
  scale_x_date(date_labels = "%b %Y")
```




```
start_date <- as.Date("2020-03-01")
end_date <- as.Date("2021-12-30")
# Filter the data for the specified range
filtered_icnsa <- icnsa %>%
  filter(date >= start_date & date <= end_date)

# Plot the filtered data with the specified range on the x-axis
ggplot(filtered_icnsa, aes(x = date, y = value)) +
  geom_line() +
  labs(title = "Insurance Claim Over Time", x = "Date", y = "Number") +
  scale_x_date(date_labels = "%b %Y")
```


Insurance Claim Over Time
6e+06
4e+06
2e+06
0e+00
Number
Jul 2020
Jan 2021
Jul 2021
Jan 2022
Date

```
# Defining the range
start_date_start <- as.Date("2020-02-01")
end_date_start <- as.Date("2020-03-20")

# Filter the data for the specified range
filtered_icnsa <- icnsa %>%
  filter(date >= start_date_start & date <= end_date_start)

# Calculate the differences in value and date
diff_value <- diff(filtered_icnsa$value)
diff_date <- as.numeric(diff(filtered_icnsa$date))

# Calculate the slopes
slopes <- diff_value / diff_date

# Find the index of the segment with the maximum slope
max_slope_index <- which.max(slopes)

# Get the corresponding start and end dates for the segment with the highest slope
highest_slope_start_date <- filtered_icnsa$date[max_slope_index]
highest_slope_end_date <- filtered_icnsa$date[max_slope_index + 1]

# Plot the data with the specified range and slope highlighted
ggplot(filtered_icnsa, aes(x = date, y = value)) +
  geom_line() +
  geom_segment(aes(x = filtered_icnsa$date[max_slope_index],
                   y = filtered_icnsa$value[max_slope_index],
                   xend = filtered_icnsa$date[max_slope_index + 1],
                   yend = filtered_icnsa$value[max_slope_index + 1]),
               color = "red", size = 1) +
  labs(title = "Insurance Claim Over Time", x = "Date", y = "Number")
```

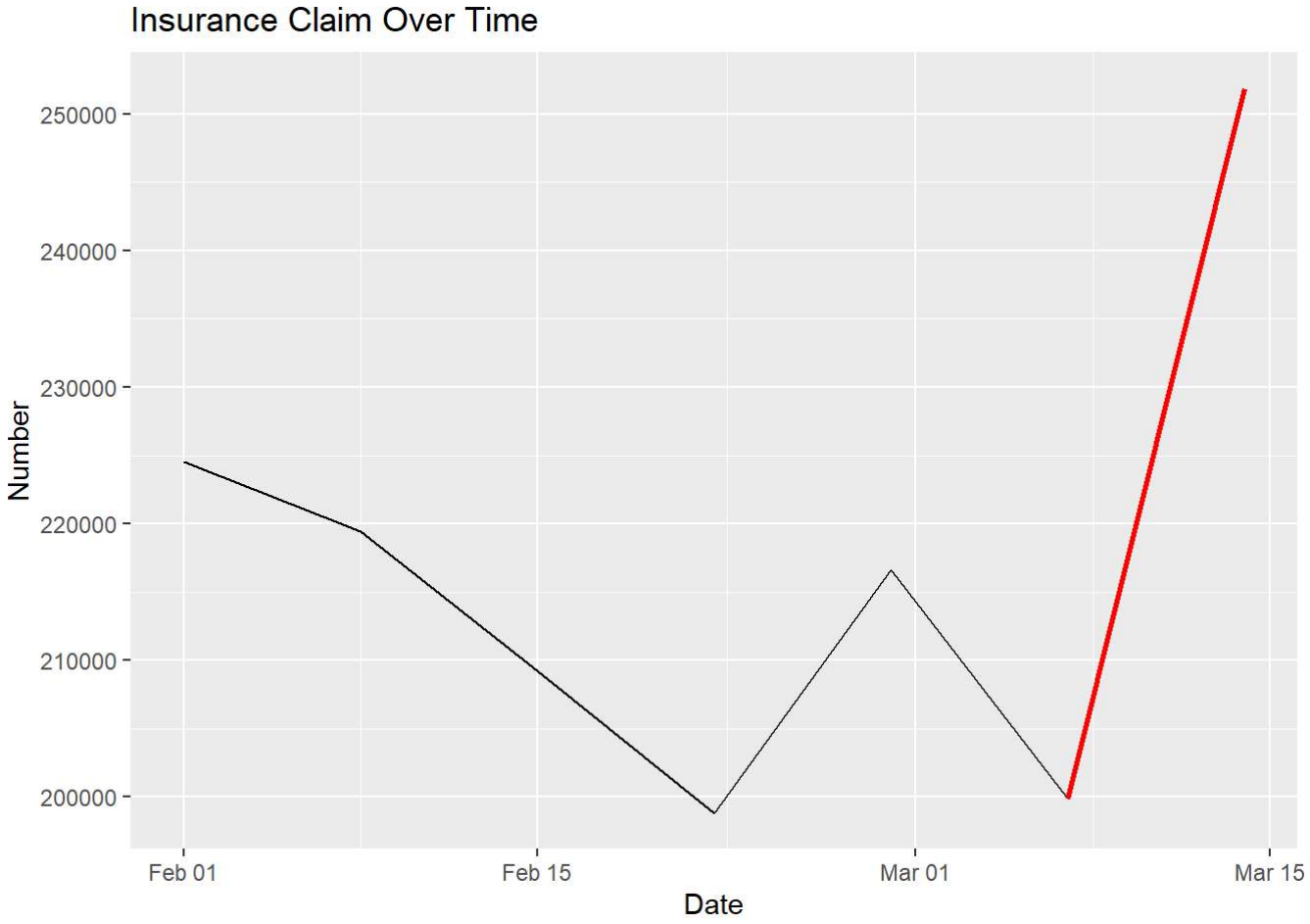
```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## ℹ Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## Warning: Use of `filtered_icnsa$date` is discouraged.
## ℹ Use `date` instead.
```

```
## Warning: Use of `filtered_icnsa$value` is discouraged.
## ℹ Use `value` instead.
```

```
## Warning: Use of `filtered_icnsa$date` is discouraged.
## ℹ Use `date` instead.
```

```
## Warning: Use of `filtered_icnsa$value` is discouraged.
## ℹ Use `value` instead.
```

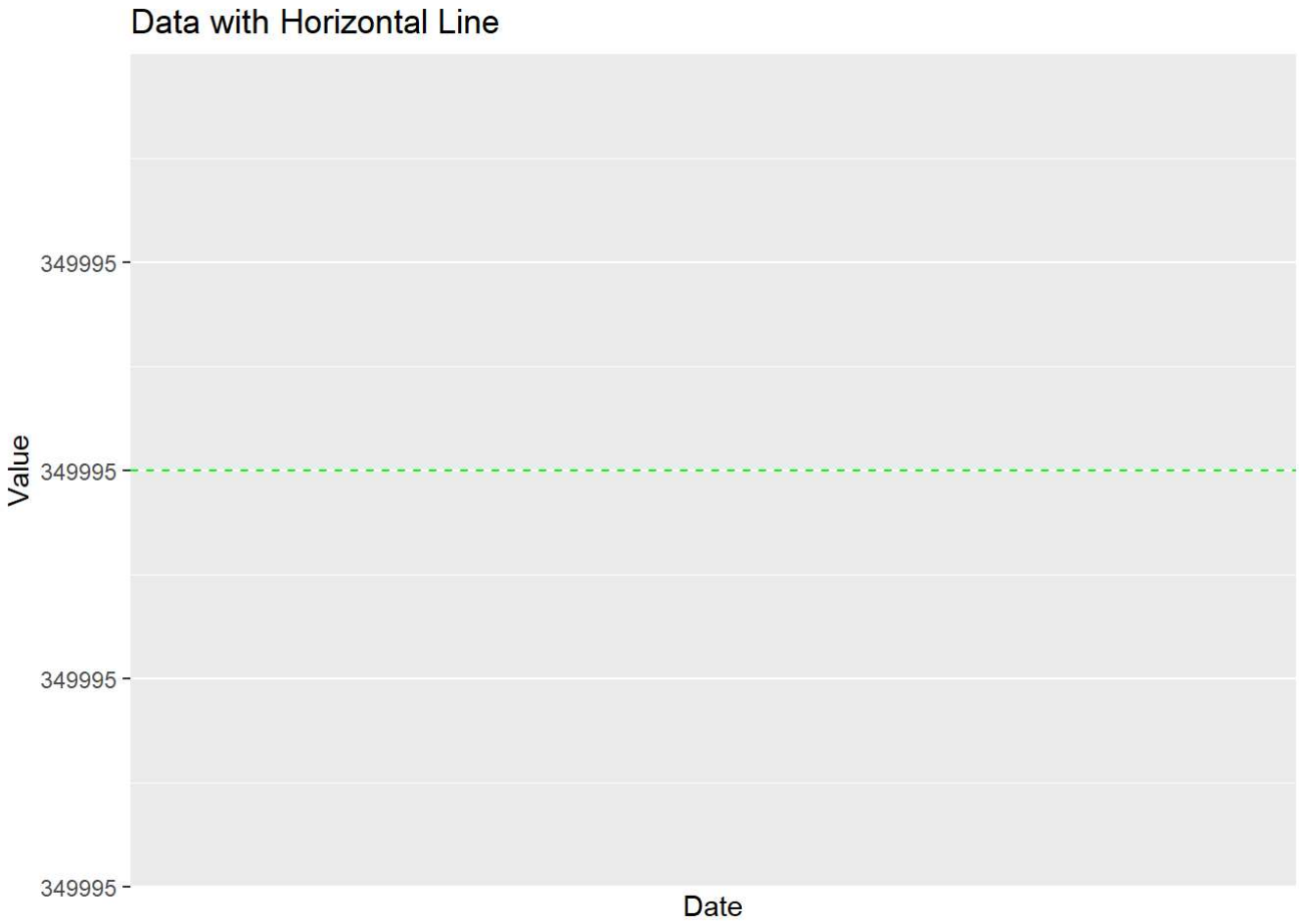



```
# Print the value of the x-axis where the slope starts
print(paste("X-axis value where slope starts:", highest_slope_start_date))
```

```
## [1] "X-axis value where slope starts: 2020-03-07"
```

#Start Date: 2020-03-07

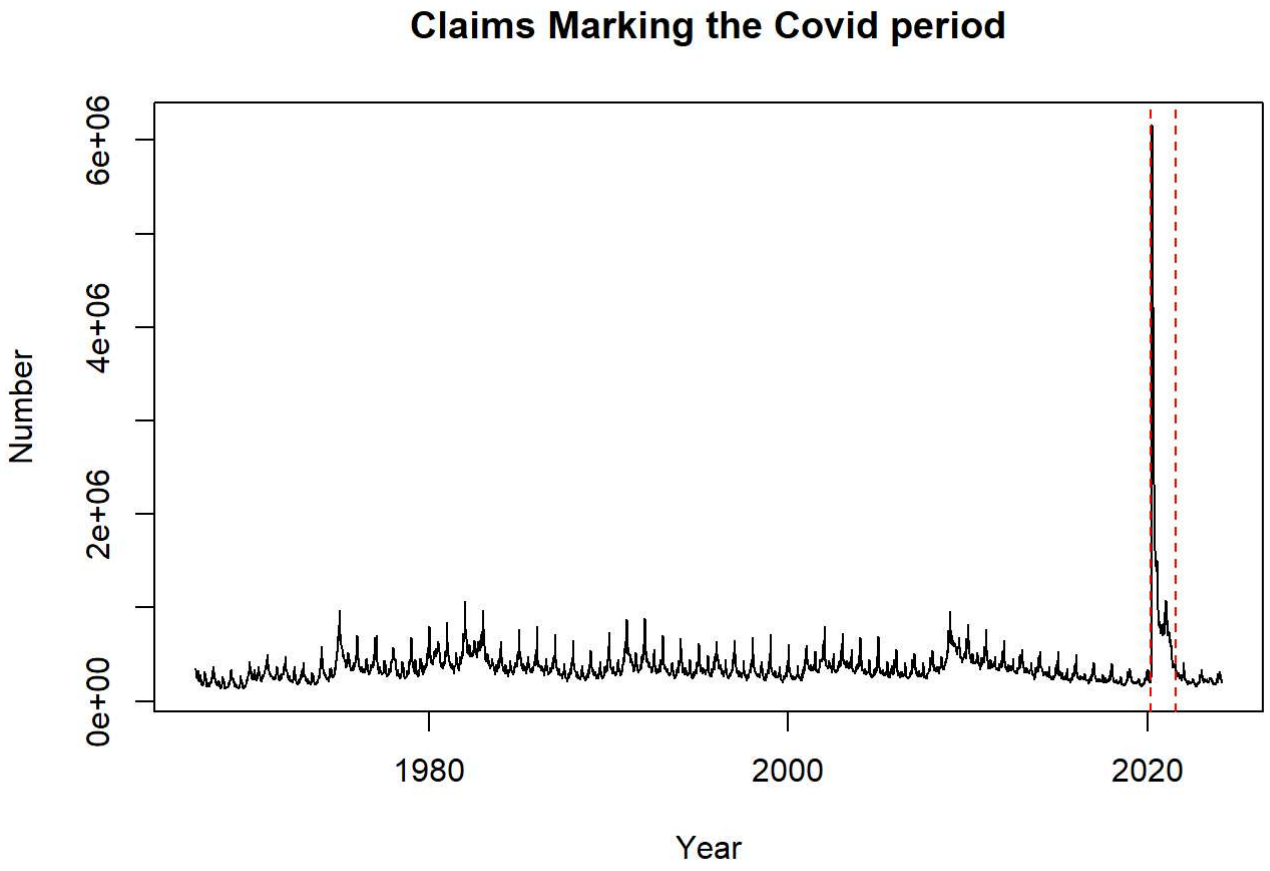
```
# Define the start and end dates
start_date_end <- as.Date("2021-01-01")
end_date_end <- as.Date("2021-12-31")

# Define the specific value (y-value)
specific_value <- 349995
# Filter the data for the specified date range
filtered_data <- icnsa_filtered %>%
  filter(date >= start_date_end & date <= end_date_end)

# Plot the filtered data with a horizontal line at the specific value
ggplot(filtered_data, aes(x = date, y = value)) +
  geom_line(color = "blue") +  # Adding color to the line
  geom_hline(yintercept = specific_value, linetype = "dashed", color = "green") +
  labs(title = "Data with Horizontal Line", x = "Date", y = "Value")
```



#COVID End date:2021-07-31

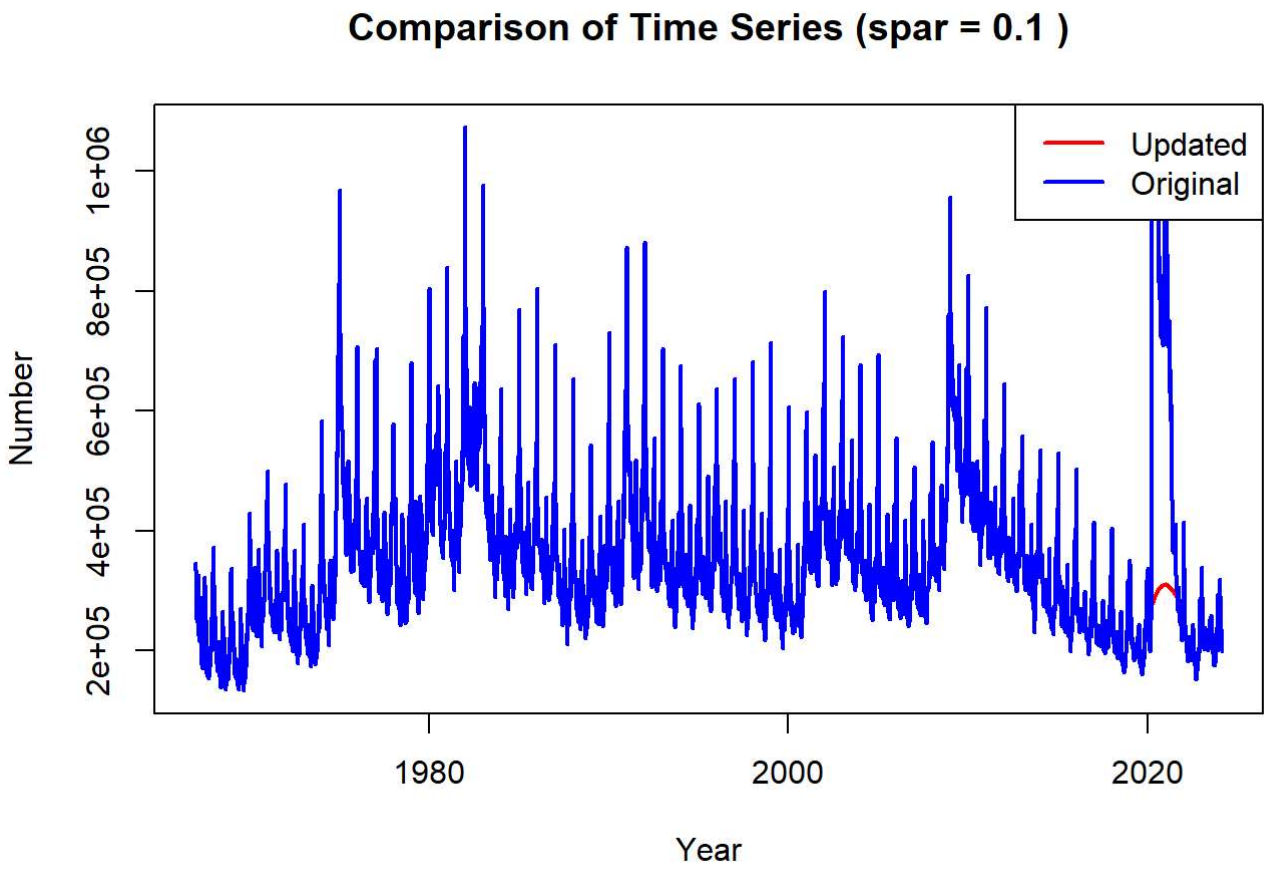
```
start_date <- as.Date("2020-03-07")
end_date <- as.Date("2021-07-31")
```

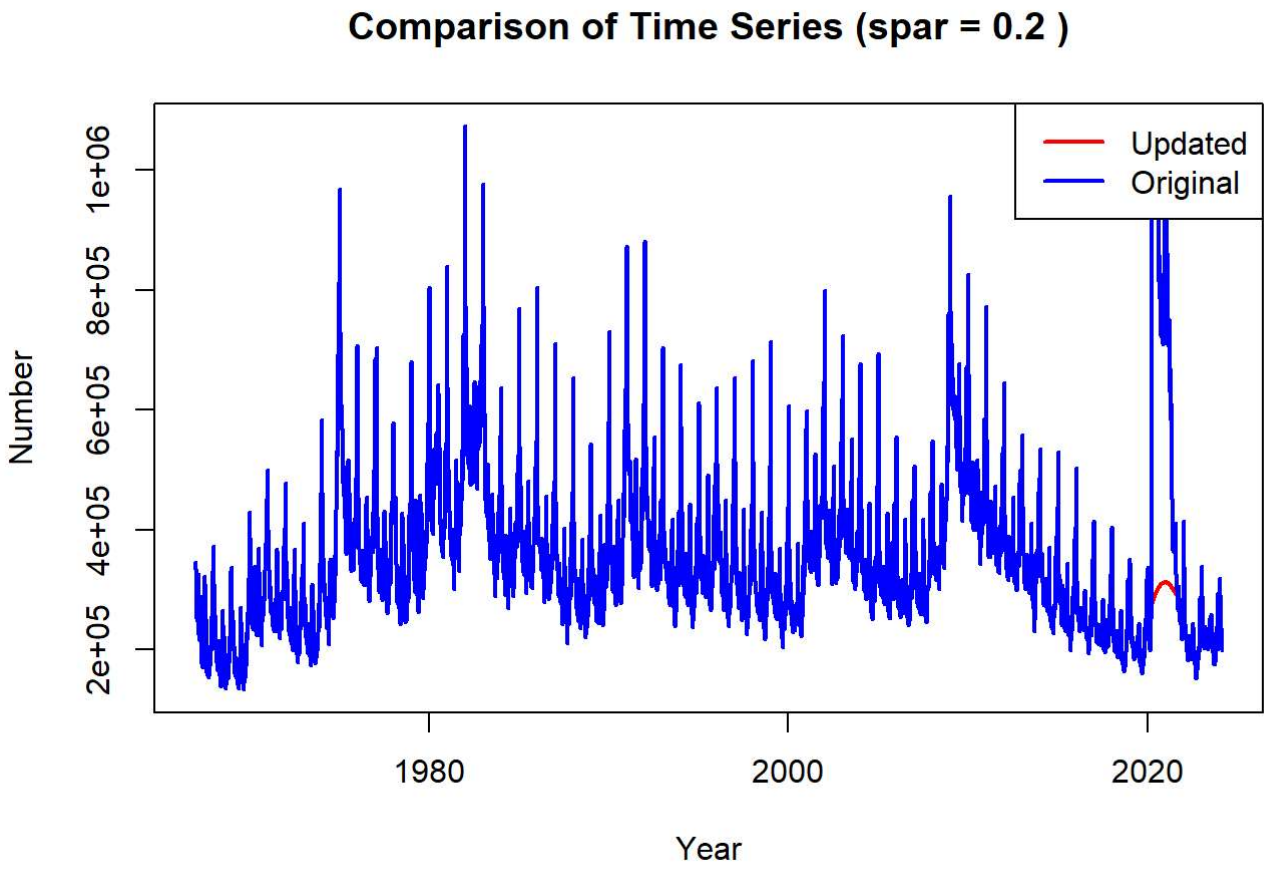
```
# Plotting Covid period data with vertical lines to visulize the covid region
plot(icnsa$date, icnsa$value, type = "l",
     main = "Claims Marking the Covid period",
     xlab = "Year",
     ylab = "Number")

abline(v = as.numeric(start_date), col = "red", lty = 2)
abline(v = as.numeric(end_date), col = "red", lty = 2)
```

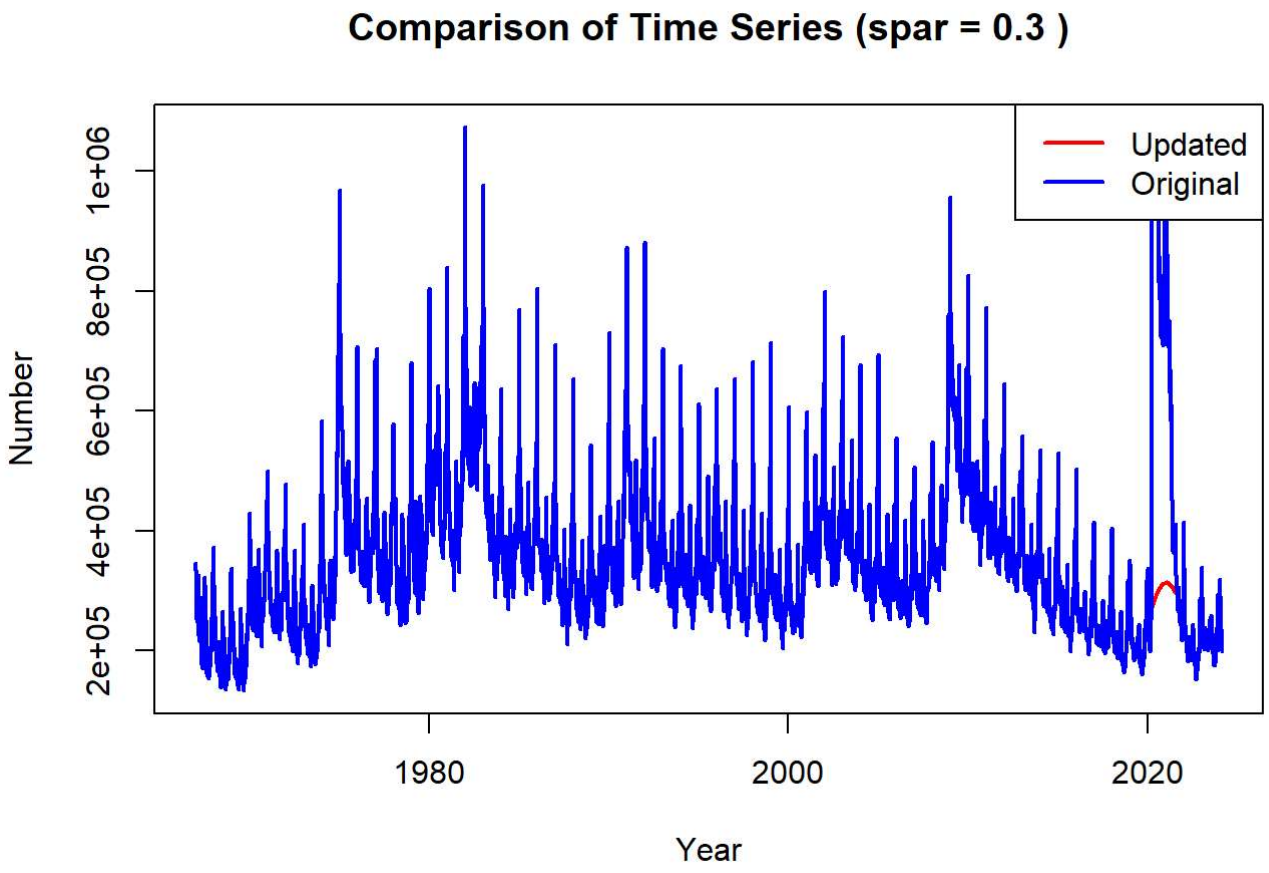


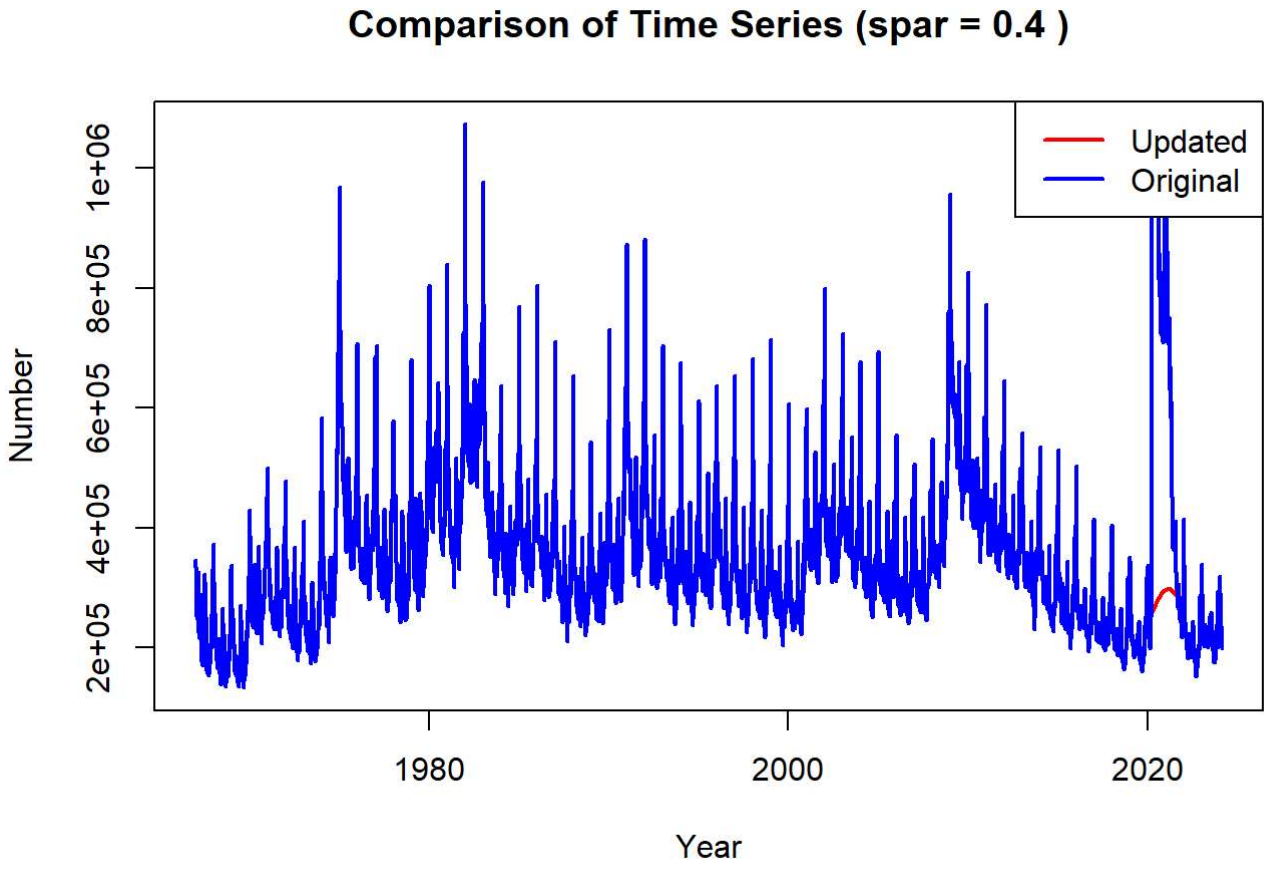
3. Use a cubic spline methodology to impute new values for the Covid period you decided on in #2. Discuss the λ value chosen and the overall fit.

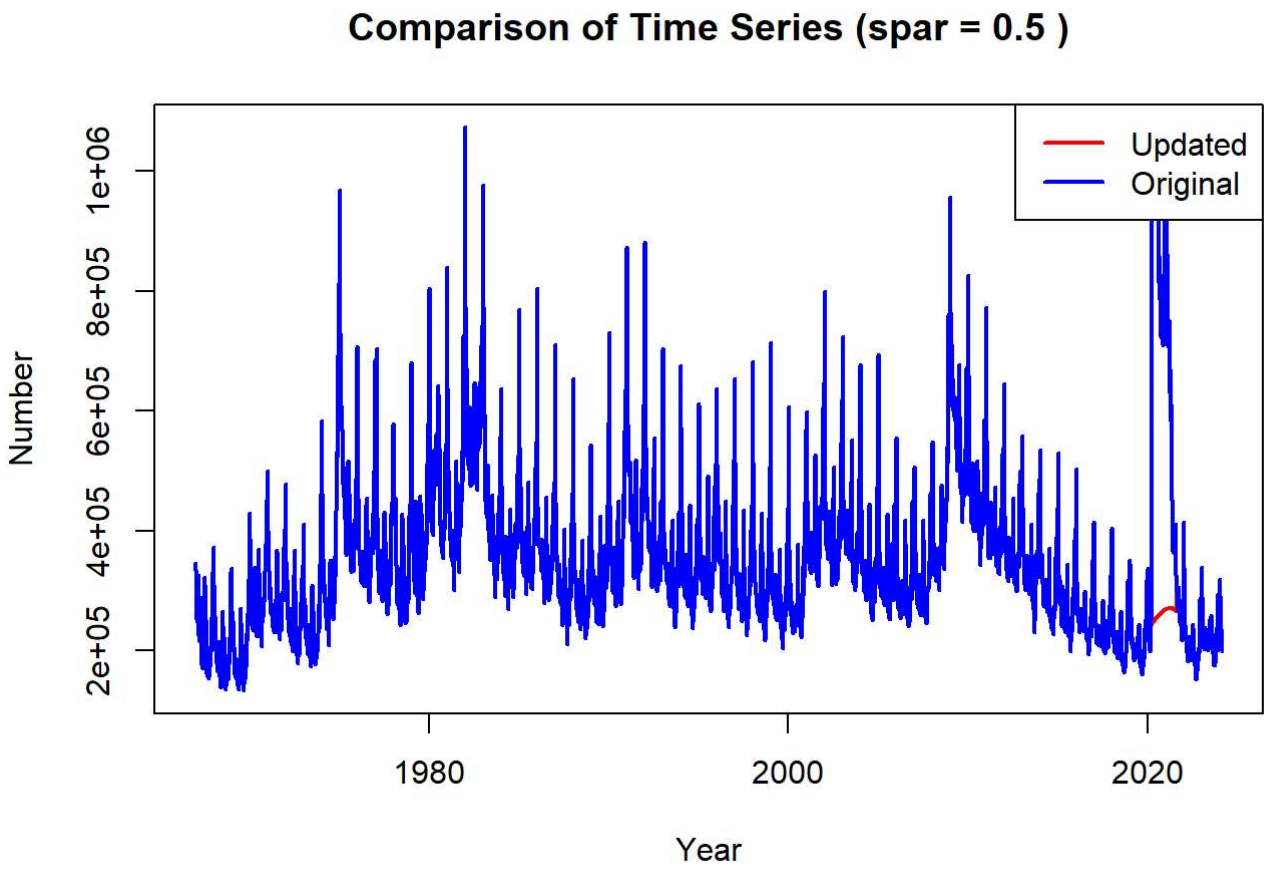
```
# Define non-Covid and Covid periods
non_covid_period <- icnsa[icnsa$date < start_date | icnsa$date > end_date, ]
covid_period <- icnsa[icnsa$date >= start_date & icnsa$date <= end_date, ]

# Initialize an empty list to store updated data frames for each spar value
updated_data_list <- list()

# Initialize an empty data frame to store lambda and MSE values
lambda_mse_df <- data.frame(lambda = numeric(), MSE = numeric())
spar_values <- seq(0.1, 1.0, by = 0.1)
# Implementing cubic spline with different values of spar
lambda_values <- seq(0.1, 1.0, by = 0.1)
for (lambda in lambda_values) {
  # Fit cubic spline to non-Covid period with current spar value
  spline_fit_covid <- smooth.spline(x = as.numeric(non_covid_period$date), y = non_covid_period
$value, spar = lambda)

  # Impute new values using spline fit
  imputed_values <- predict(spline_fit_covid, x = as.numeric(covid_period$date))$y

  # Updating Covid period data with imputed values
  covid_period$value <- imputed_values

  # Combining non-Covid and imputed values
  updated_icnsa_data <- rbind(non_covid_period, covid_period)
  updated_icnsa_data <- updated_icnsa_data %>% arrange(date)

  # Store updated data frame in the list
  updated_data_list[[as.character(lambda)]] <- updated_icnsa_data

  # Plotting updated data
  plot(updated_icnsa_data$date, updated_icnsa_data$value, type = "l", col = "red", lwd = 2,
       main = paste("Comparison of Time Series (spar =", lambda, ")"), xlab = "Year", ylab = "Nu
mber")

  # Add original data for comparison
  lines(icnsa$date, icnsa$value, col = "blue", lwd = 2)

  # Add legend
  legend("topright", legend = c("Updated", "Original"), col = c("red", "blue"), lty = 1, lwd =
2)
}
```

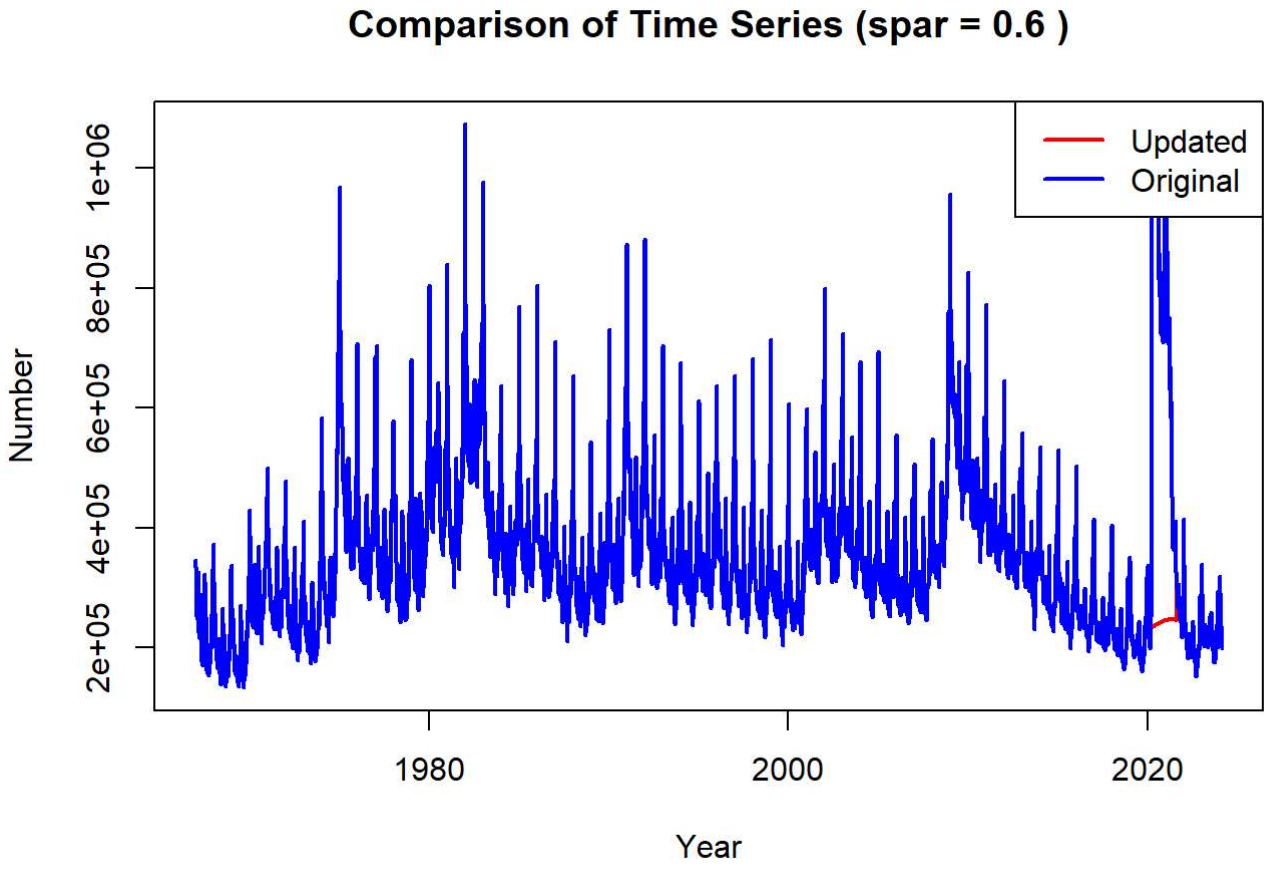

Comparison of Time Series (spar = 0.1)
Updated
Original
Number
2e+05
4e+05
6e+05
8e+05
1e+06
1980
2000
2020
Year

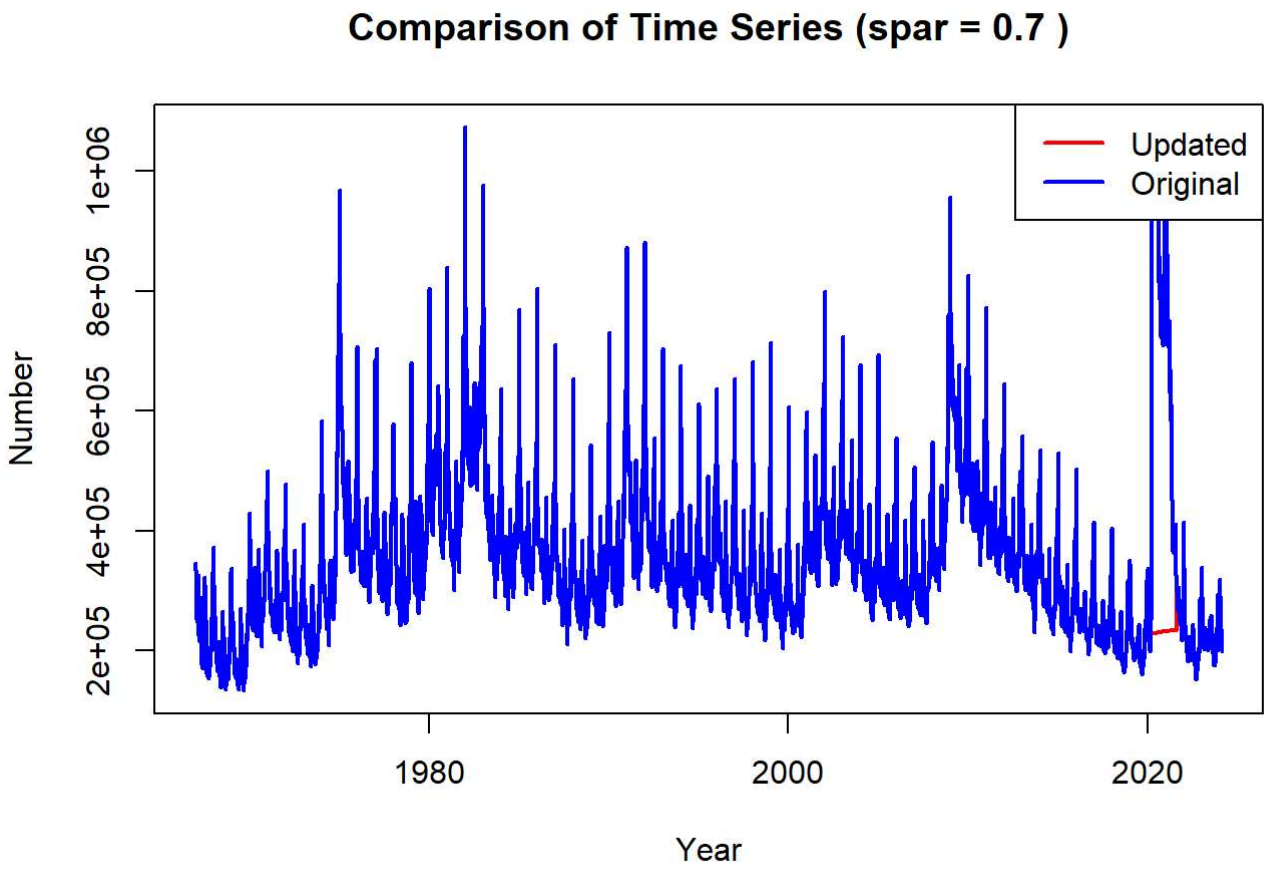

Comparison of Time Series (spar = 0.2)
Updated
Original
Number
2e+05
4e+05
6e+05
8e+05
1e+06
1980
2000
2020
Year

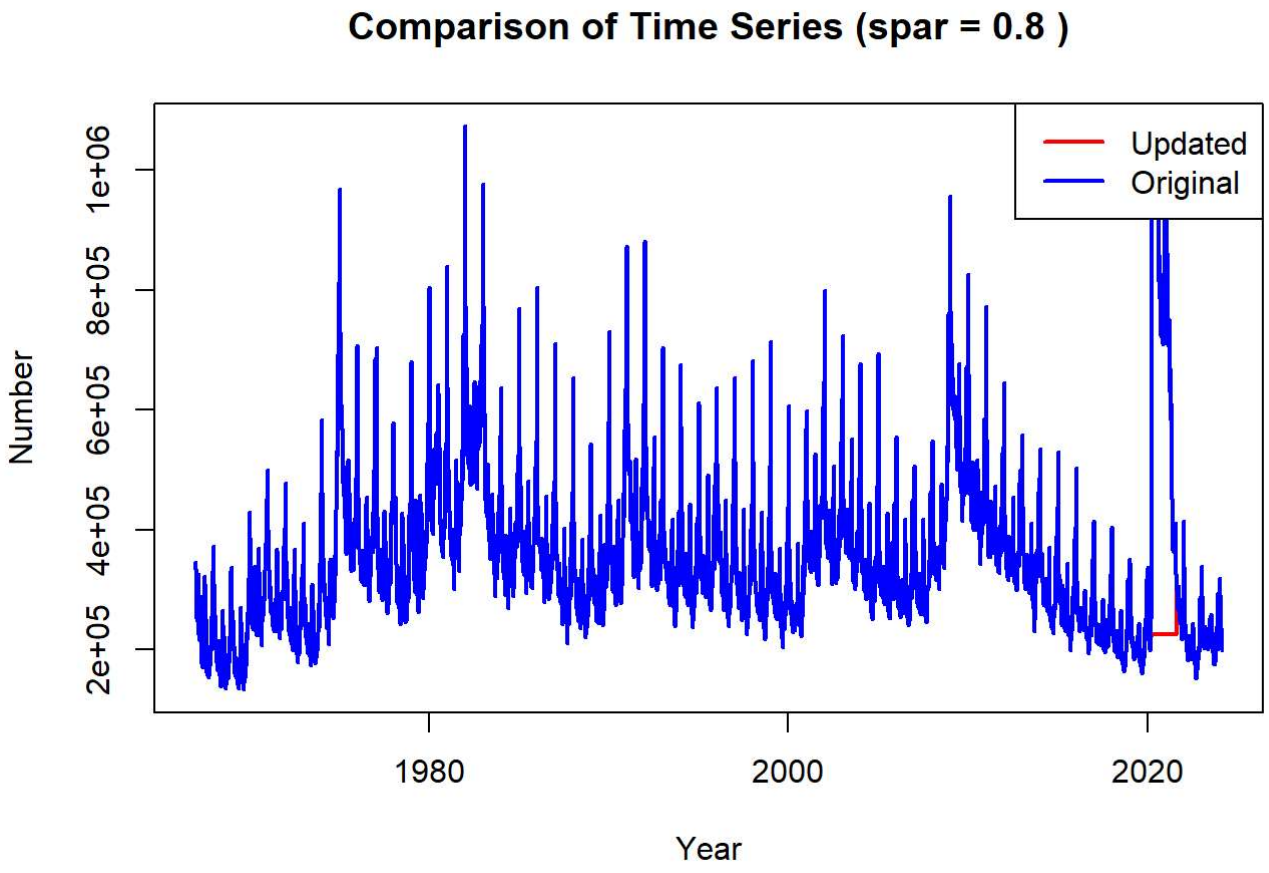

Comparison of Time Series (spar = 0.3)
Updated
Original
Number
2e+05
4e+05
6e+05
8e+05
1e+06
1980
2000
2020
Year

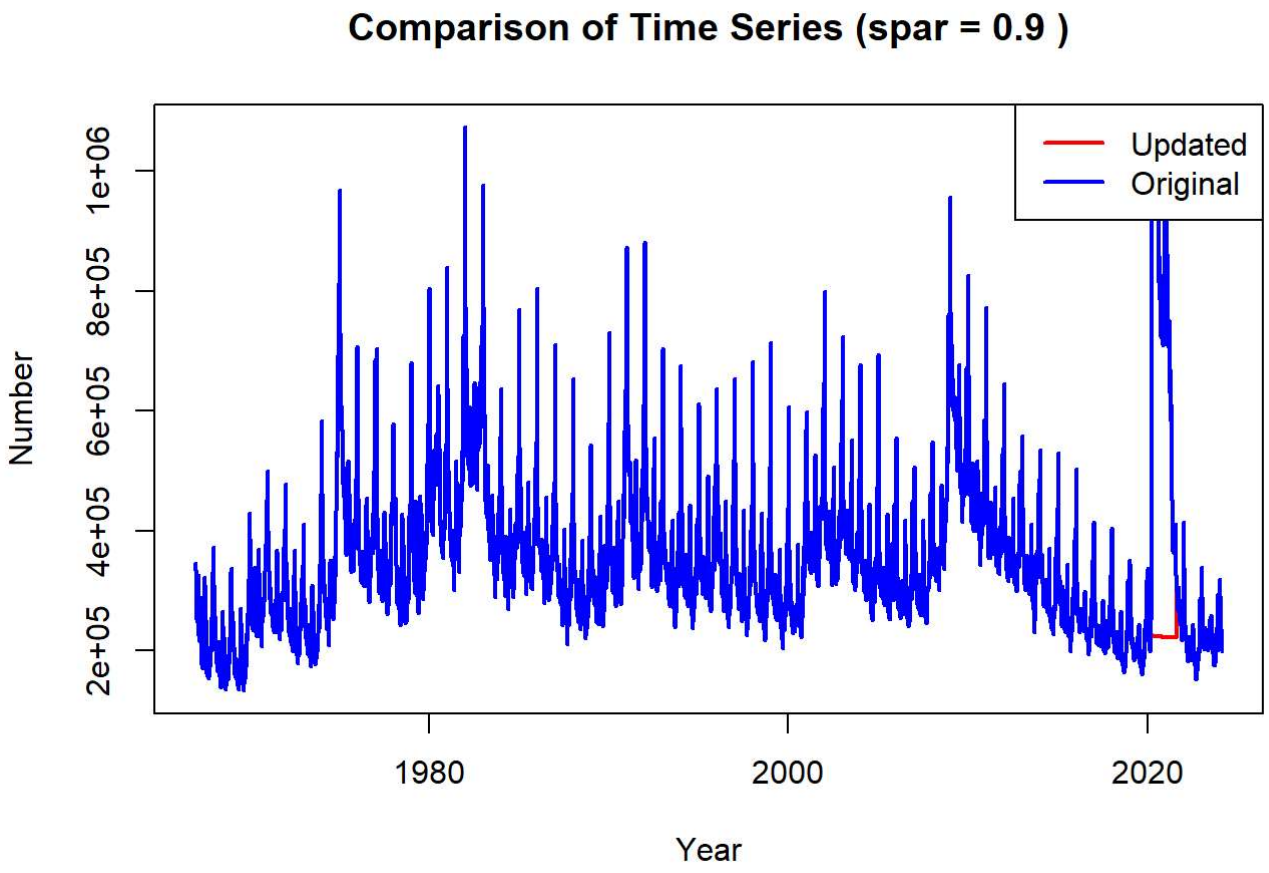

Comparison of Time Series (spar = 0.4)
Updated
Original
Number
2e+05
4e+05
6e+05
8e+05
1e+06
1980
2000
2020
Year

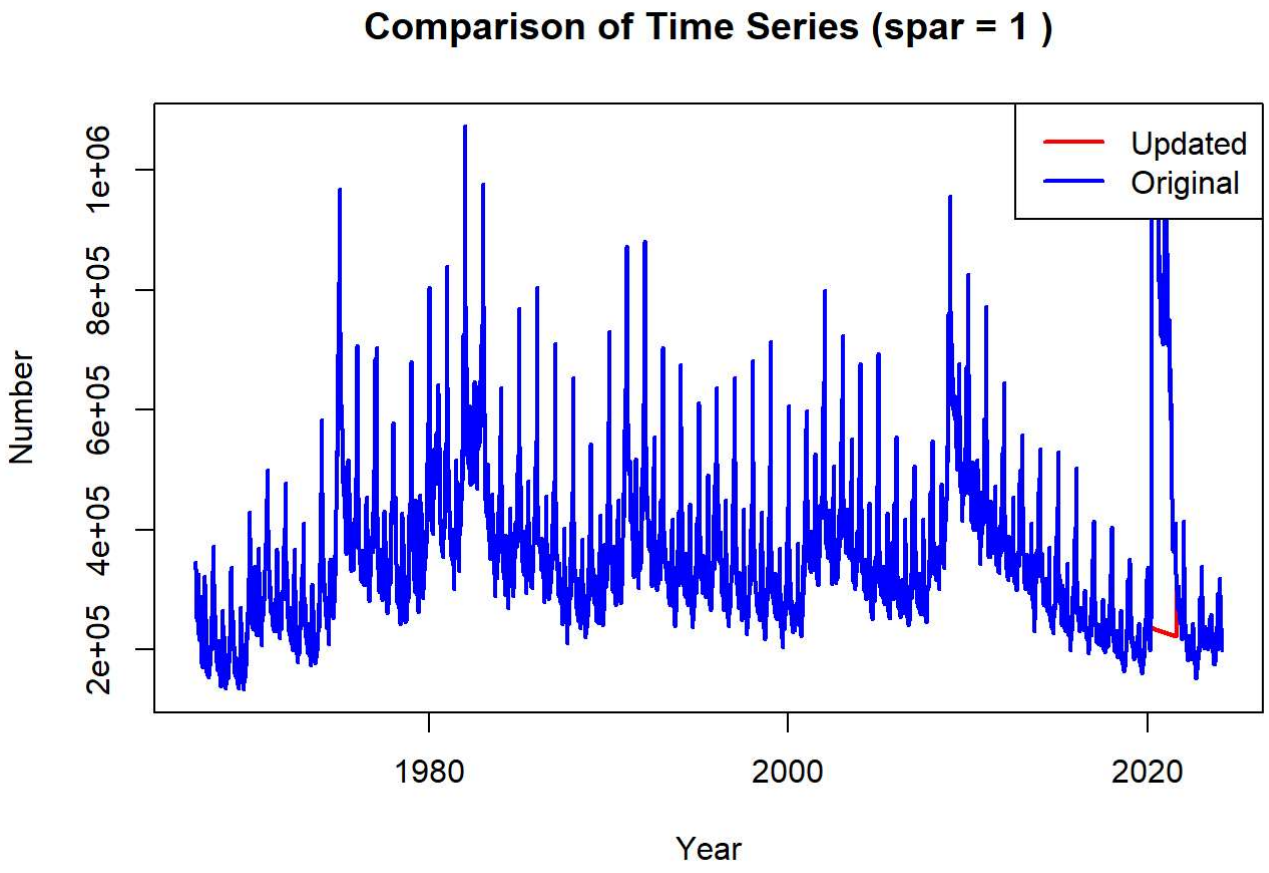

Comparison of Time Series (spar = 0.5)
Updated
Original
Number
2e+05
4e+05
6e+05
8e+05
1e+06
1980
2000
2020
Year


Comparison of Time Series (spar = 0.6)
Updated
Original
Number
2e+05
4e+05
6e+05
8e+05
1e+06
1980
2000
2020
Year


Comparison of Time Series (spar = 0.7)
Updated
Original
Number
2e+05
4e+05
6e+05
8e+05
1e+06
1980
2000
2020
Year


Comparison of Time Series (spar = 0.8)
Updated
Original
Number
2e+05
4e+05
6e+05
8e+05
1e+06
1980
2000
2020
Year


Comparison of Time Series (spar = 0.9)
Updated
Original
Number
2e+05
4e+05
6e+05
8e+05
1e+06
1980
2000
2020
Year


Comparison of Time Series (spar = 1)
Updated
Original
Number
2e+05
4e+05
6e+05
8e+05
1e+06
1980
2000
2020
Year

```
# Print the table of lambda and MSE values
print(lambda_mse_df)
```

```
## [1] lambda MSE
## <0 rows> (or 0-length row.names)
```

```
# Initialize empty vectors to store AIC and BIC values
aic_values <- numeric()
bic_values <- numeric()

# Calculate AIC and BIC for the current model
residuals <- covid_period$value - imputed_values
rss <- sum(residuals^2)
n <- length(covid_period$value)
k <- 3  # Number of parameters in the model (intercept, lambda, and degree of freedom)
aic_values <- c(aic_values, n * log(rss/n) + 2 * k)
bic_values <- c(bic_values, n * log(rss/n) + k * log(n))

# Find the index of the minimum AIC and BIC values
min_aic_index <- which.min(aic_values)
min_bic_index <- which.min(bic_values)

# Get the corresponding lambda values
best_lambda_aic <- lambda_values[min_aic_index]
best_lambda_bic <- lambda_values[min_bic_index]

# Print the lambda values corresponding to the minimum AIC and BIC
cat("Lambda value corresponding to minimum AIC:", best_lambda_aic, "\n")
```

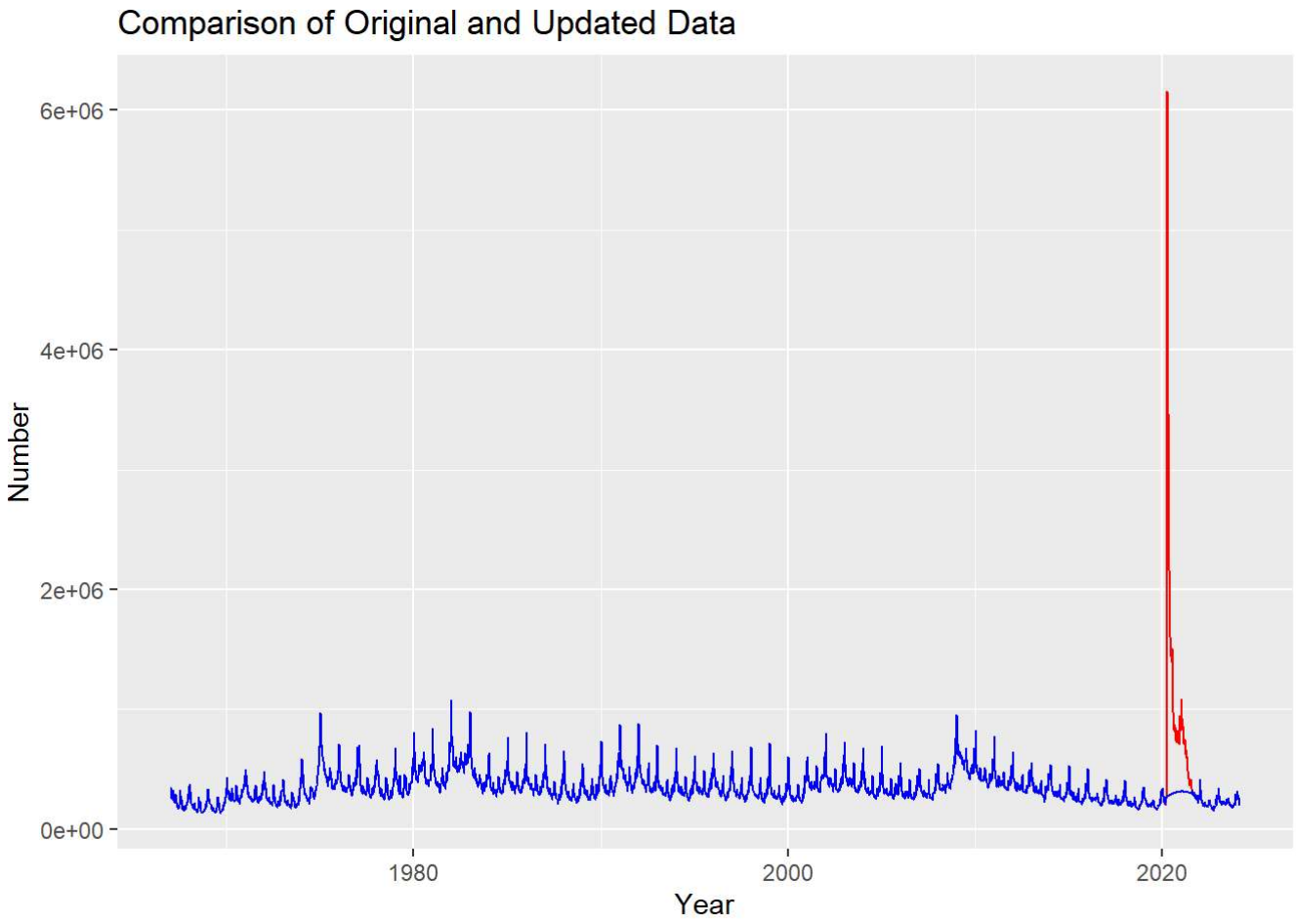
```
## Lambda value corresponding to minimum AIC: 0.1
```

```
cat("Lambda value corresponding to minimum BIC:", best_lambda_bic, "\n")
```

```
## Lambda value corresponding to minimum BIC: 0.1
```

Choosing λ = 0.3 achieves a delicate equilibrium between smoothing and preserving data trends. While lower values such as 0.1 minimize RMSE, they raise concerns of overfitting to noise. Conversely, higher values like 1.0 tend to oversmooth, sacrificing data nuances. λ = 0.3 effectively tackles overfitting while still capturing essential trends, making it a judicious choice for interpolation.

```
# Fit cubic spline to non-Covid period with spar value of 0.3
spline_fit_covid <- smooth.spline(x = as.numeric(non_covid_period$date), y = non_covid_period$va
lue, spar = 0.3)

# Impute new values using spline fit
imputed_values <- predict(spline_fit_covid, x = as.numeric(covid_period$date))$y

# Updating Covid period data with imputed values
covid_period$value <- imputed_values

# Combining non-Covid and imputed values
updated_icnsa_data <- rbind(non_covid_period, covid_period)
updated_icnsa_data <- updated_icnsa_data %>% arrange(date)

# Plotting original and updated data together
ggplot() +
  geom_line(data = icnsa, aes(x = date, y = value), color = "red") +
  geom_line(data = updated_icnsa_data, aes(x = date, y = value), color = "blue") +
  labs(title = "Comparison of Original and Updated Data", x = "Year", y = "Number")
```



```
icnsa_ts <- ts(updated_icnsa_data$value, frequency = 52)

#Applied Holt-Winters models both  additive and multiplicative
hw_add <- HoltWinters(icnsa_ts, seasonal = "additive")
hw_mult <- HoltWinters(icnsa_ts, seasonal = "multiplicative")

#Forecasted next value using both models
forecast_add <- forecast::forecast(hw_add, h = 1)
forecast_mult <- forecast::forecast(hw_mult, h = 1)

point_forecast_add <- forecast_add$mean
point_forecast_mult <- forecast_mult$mean


#Printed Point Forecast value of both model
cat("Forecasted Unemployment using Holt-Winters additive Model:", point_forecast_add, "\n")
```

```
## Forecasted Unemployment using Holt-Winters additive Model: 206758.9
```

```
cat("Forecasted Unemployment using Holt-Winters Multiplicative Model:", point_forecast_mult,
"\n")
```

```
## Forecasted Unemployment using Holt-Winters Multiplicative Model: 192799.6
```