COVID Cases and Deaths

4/2/2023

Abstract

This report aims to study the seasonality patterns in COVID-19 cases and deaths in Wisconsin, USA over March 2020 to March 2023. Wisconsin was used as a proxy for a region with four seasons.

Is there a relationship between time of year and case/death frequency? Does the flu season overlap with peak COVID-19 cases and deaths?

COVID-19 was a pandemic that caused shock waves throughout societies and economies. Studying the relationship between month of year and correlations with the flu seasons will provide insights applicable to future flu-like illnesses.

Acknowledging my personal bias that there cases and deaths reported are significantly correlated and driven by the flu season, I will scrutinize the data from the view point of someone who wants to disprove seasonality with peak COVID-19 cases and deaths.

One area of potential further analysis is to compare how the seasonality had changed from 2020 to 2022, if there have been any.

Libraries Use: - dpylr, BSDA, tidyverse, lubridate, ggplot2

Importing Data

Both data sets were automatically updated csv files (now archived and deprecated) maintained by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University. The last date of update was 3/9/23.

Each row of the US Deaths data is a different city in a US State. Each say's death count were updated as a new column.

Each row of the US Cases data is also a different city in a US State. Each day's case count was updated as a new column.

```
library(tidyverse)
```

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

## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.3     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.4.3     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
deaths_url = "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_c

confirmed_url = "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/css

deaths_data = read_csv(deaths_url)
```

```
## Rows: 3342 Columns: 1155
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr    (6): iso2, iso3, Admin2, Province_State, Country_Region, Combined_Key
## dbl (1149): UID, code3, FIPS, Lat, Long_, Population, 1/22/20, 1/23/20, 1/24...
##
## 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.
```

```r
confirmed_data = read_csv(confirmed_url)
```

```
## Rows: 3342 Columns: 1154
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr    (6): iso2, iso3, Admin2, Province_State, Country_Region, Combined_Key
## dbl (1148): UID, code3, FIPS, Lat, Long_, 1/22/20, 1/23/20, 1/24/20, 1/25/20...
##
## 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.
```

Tidying Data

Transforming data prior to further analysis.

Keeping only Wisconsin cases and deaths:

```r
library(dplyr)

death <- deaths_data[deaths_data$Province_State == 'Wisconsin', ]
case <- confirmed_data[confirmed_data$Province_State == 'Wisconsin', ]
```

Aggregating all cities within Wisconsin and joining the Case and Death columns to one main data frame:

```r
# Aggregating all cities and joining cases to deaths

df <- merge(
  colSums(select(case, -1:-11)),
  colSums(select(death, -1:-12)),
  by = 0
)
colnames(df) <- c("Date", "Case", "Death")
```

Turning Date into Datetime and checking all data types:

```
library(lubridate)
df$Date <- mdy(df$Date)

df <- df[order(df$Date), ]
```

Checking missing data and how many 0 case and death days there are:

```
str(df)
```

```
## 'data.frame':    1143 obs. of  3 variables:
##  $ Date : Date, format: "2020-01-22" "2020-01-23" ...
##  $ Case : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Death: num  0 0 0 0 0 0 0 0 0 0 ...
```

```
print('Null:')
```

```
## [1] "Null:"
```

```
colSums(is.na(df))
```

```
##  Date  Case Death
##     0     0     0
```

```
print('Case or Death ==0:')
```

```
## [1] "Case or Death ==0:"
```

```
colSums(df[, c("Case", "Death")] == 0)
```

```
##  Case Death
##    48    57
```

```
df <- df[df$Case != 0, ]
```

Feature Engineering

Create a new column for new daily cases and new daily deaths: Creating a month and a year column:

```
rownames(df) <- NULL
df$New_Cases <- c(df$Case[1], diff(df$Case))
df$New_Deaths <- c(df$Death[1], diff(df$Death))
df$Month <- as.integer(format(df$Date, "%Y%m"))
df$Year <- as.integer(format(df$Date, "%Y"))
```

Bringing in Flu Season and Peak Flu Season Categorical Variable:

```
df$Flu <- ifelse(month(df$Date) %in% c(10, 11, 12, 1, 2, 3), 1, 0)
df$PeakFlu <- ifelse(month(df$Date) %in% c(12, 1, 2), 1, 0)
```

Check if New_Cases and New_Deaths both have 0 as a minimum: - a negative daily case or death is not possible - fill negative daily case or death with 0

```r
min(df$New_Cases)
```

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

```r
min(df$New_Deaths)
```

```
## [1] -69
```

```r
df[df$New_Deaths < 0, ]
```

```
##           Date    Case Death New_Cases New_Deaths  Month Year Flu PeakFlu
## 370 2021-03-14  625359  7176       314         -2 202103 2021   1       0
## 391 2021-04-04  639227  7340       637         -1 202104 2021   0       0
## 405 2021-04-18  651545  7420       596         -2 202104 2021   0       0
## 412 2021-04-25  656526  7473       465         -1 202104 2021   0       0
## 419 2021-05-02  661685  7567       476         -2 202105 2021   0       0
## 505 2021-07-27  684119  8184      1078        -69 202107 2021   0       0
## 515 2021-08-06  695245  8300      1312         -5 202108 2021   0       0
## 756 2022-04-04 1586767 14316       999         -1 202204 2022   0       0
## 806 2022-05-24 1661777 14527      6969         -4 202205 2022   0       0
## 990 2022-11-24 1920758 15589      1524         -5 202211 2022   1       0
```

```r
df$New_Deaths <- ifelse(df$New_Deaths < 0, 0, df$New_Deaths)

df = df[, !(names(df) %in% c("Case", "Death"))]
```

Creating a Monthly Case and Death Summary DF - This will be used for further exploration in the analysis section - Flu and Peak Flu categorical values can be aggregate and averaged as they will all either be 0 or 1

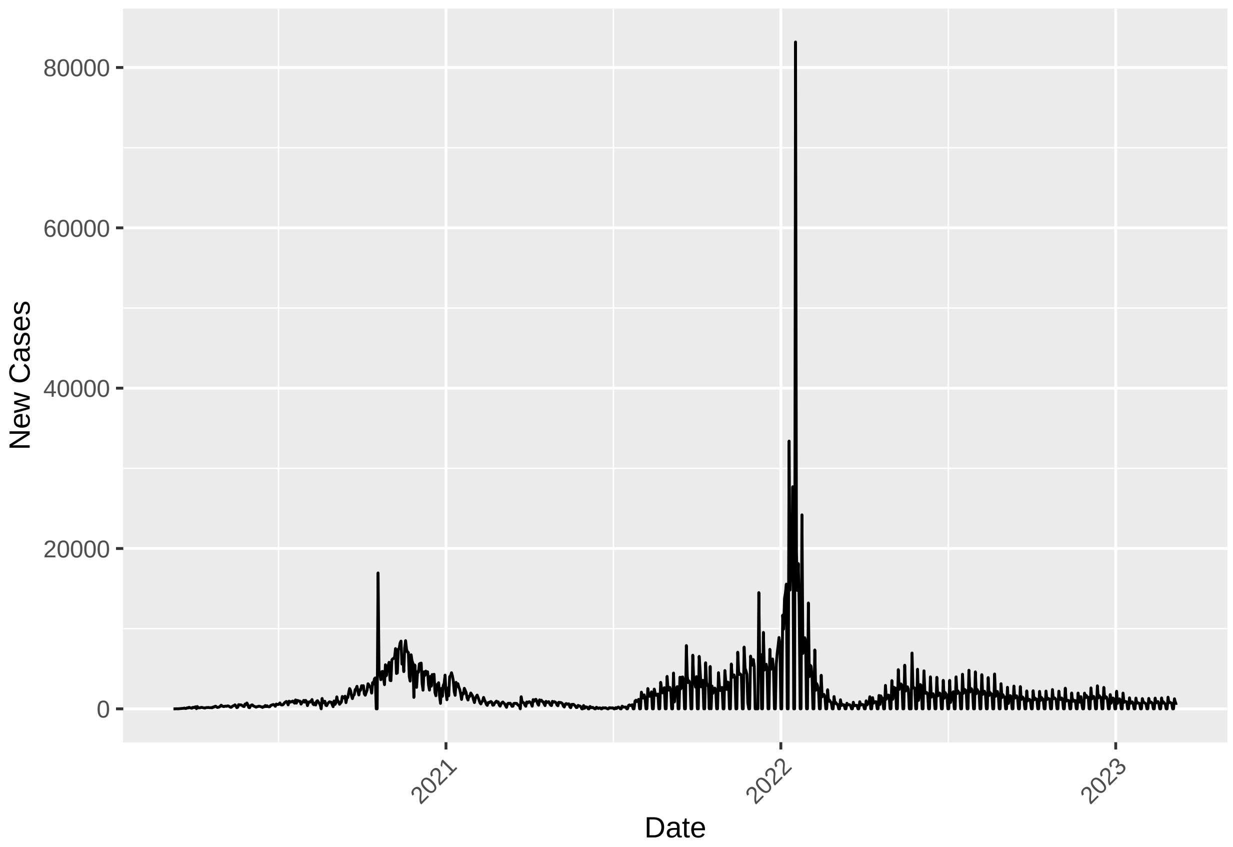
```r
df_month = df %>%
  group_by(Month) %>%
  summarise(Total_Cases = sum(New_Cases),
            Total_Deaths = sum(New_Deaths),
            Flu = mean(Flu),
            PeakFlu = mean(PeakFlu))
```

Visualyzing and Analyzing Data

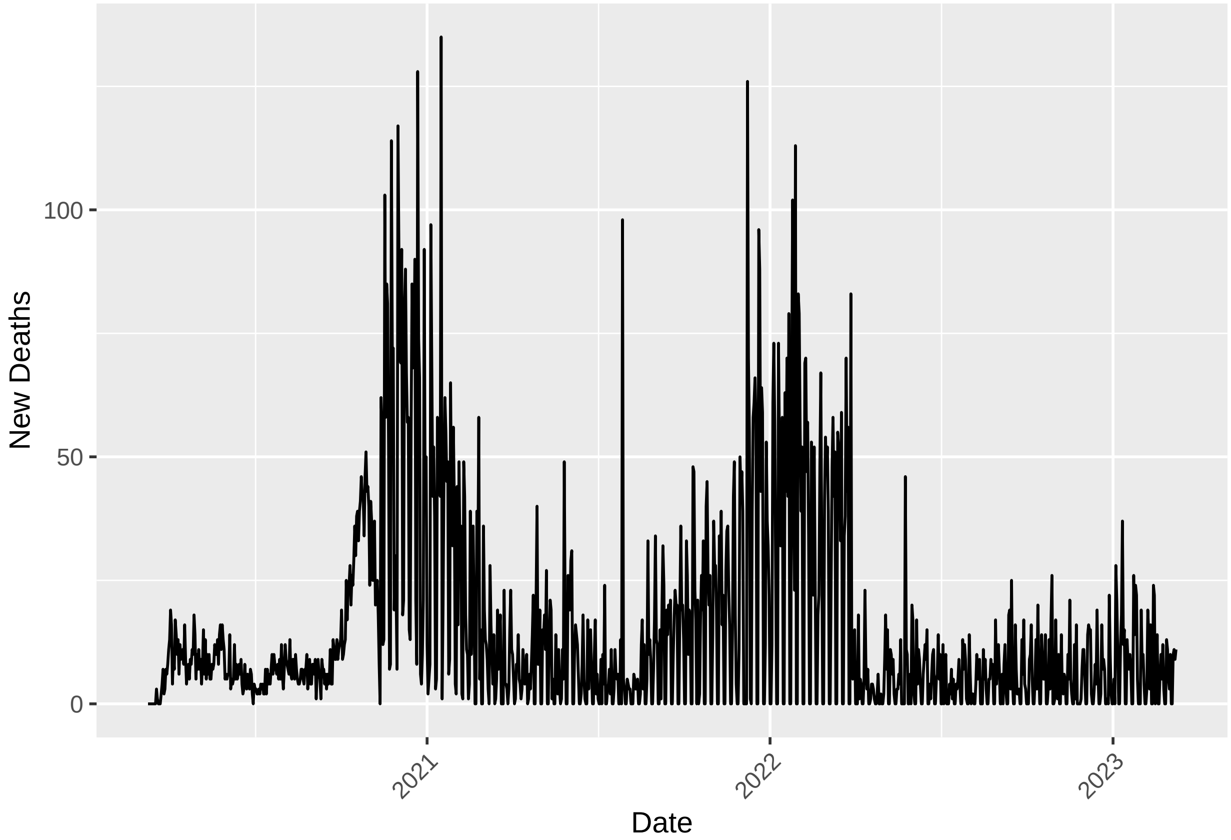
Starting off, lets take a look a the time series plot for daily cases.

- Additional Question for Further Investigation: New cases seems very messy with huge spikes in 2022. To visualize better intra-year seasonality, lets try to normalize the values per year to see if there is an intra-year seasonality. For this project we will explore min max intra-year scaling.
- New deaths seem to be better contained in a min and max range of 0 to 140. Seasonality visualization should be easier without intra-year min max scaling.

```r
library(ggplot2)

ggplot(df, aes(x = Date, y = New_Cases)) +
  geom_line() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(x = "Date", y = "New Cases")
```

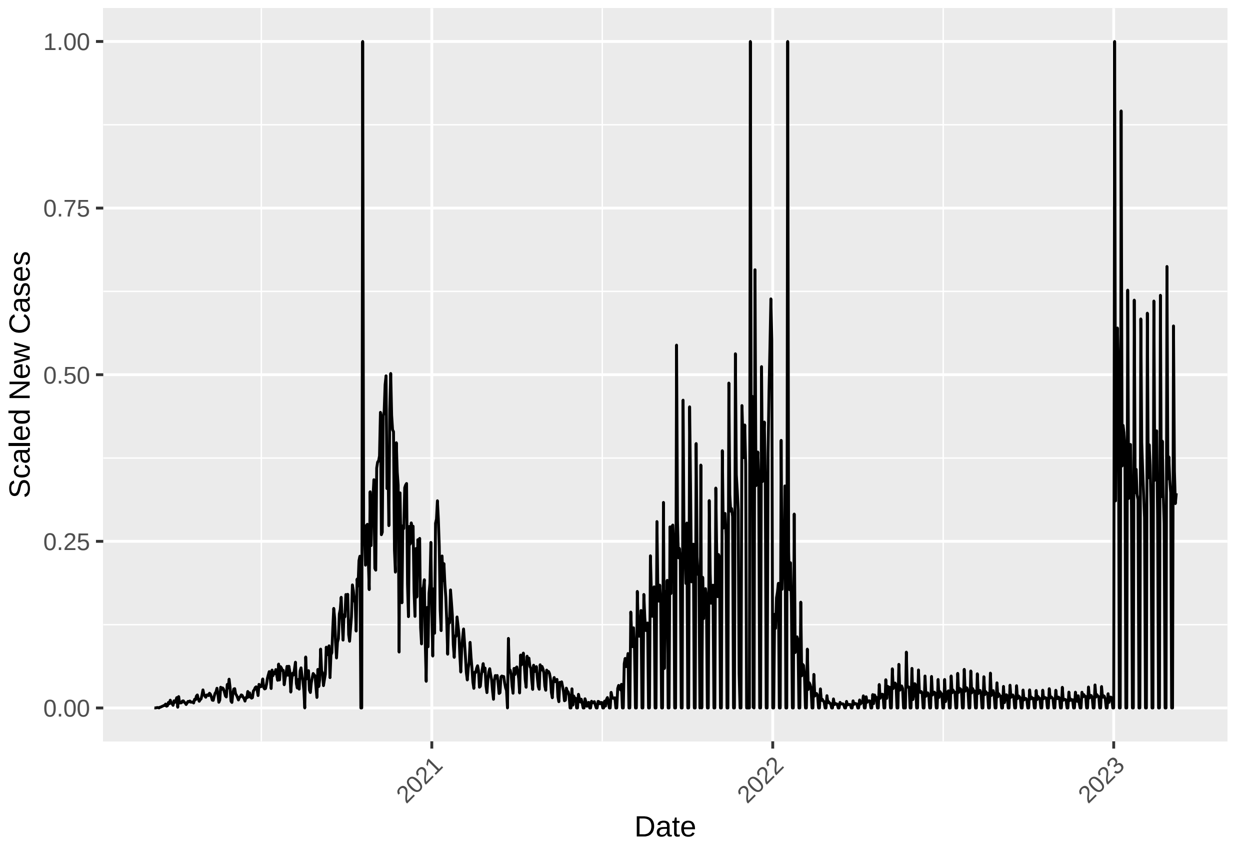


```
ggplot(df, aes(x = Date, y = New_Deaths)) +
  geom_line() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(x = "Date", y = "New Deaths")
```



Normalizing intra-year cases to explore seasonality patterns within a calendar year. - Note: There may be patterns that span across calendar years (flu season) that by normalizing, we may not see.

```
# Function to calculate annual min-max scaling
annual_min_max <- function(df0, col) {
  df <- df0

  # Calculate minimum and maximum values for each year
  x_min <- aggregate(df[[col]], by = list(Year = df$Year), FUN = min)
  colnames(x_min) <- c("Year", "Min")

  x_max <- aggregate(df[[col]], by = list(Year = df$Year), FUN = max)
  colnames(x_max) <- c("Year", "Max")

  # Merge minimum and maximum values back into the original data frame
  df_minmax <- merge(merge(df, x_min, by = "Year"), x_max, by = "Year")

  # Calculate scaled column
  df_minmax[[paste0("Scaled_", col)]] <- (df_minmax[[col]] - df_minmax$Min) / (df_minmax$Max - df_minmax

  return(df_minmax)
}

df_minmax <- annual_min_max(df, "New_Cases")

ggplot(df_minmax, aes(x = Date, y = Scaled_New_Cases)) +
  geom_line() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(x = "Date", y = "Scaled New Cases")
```



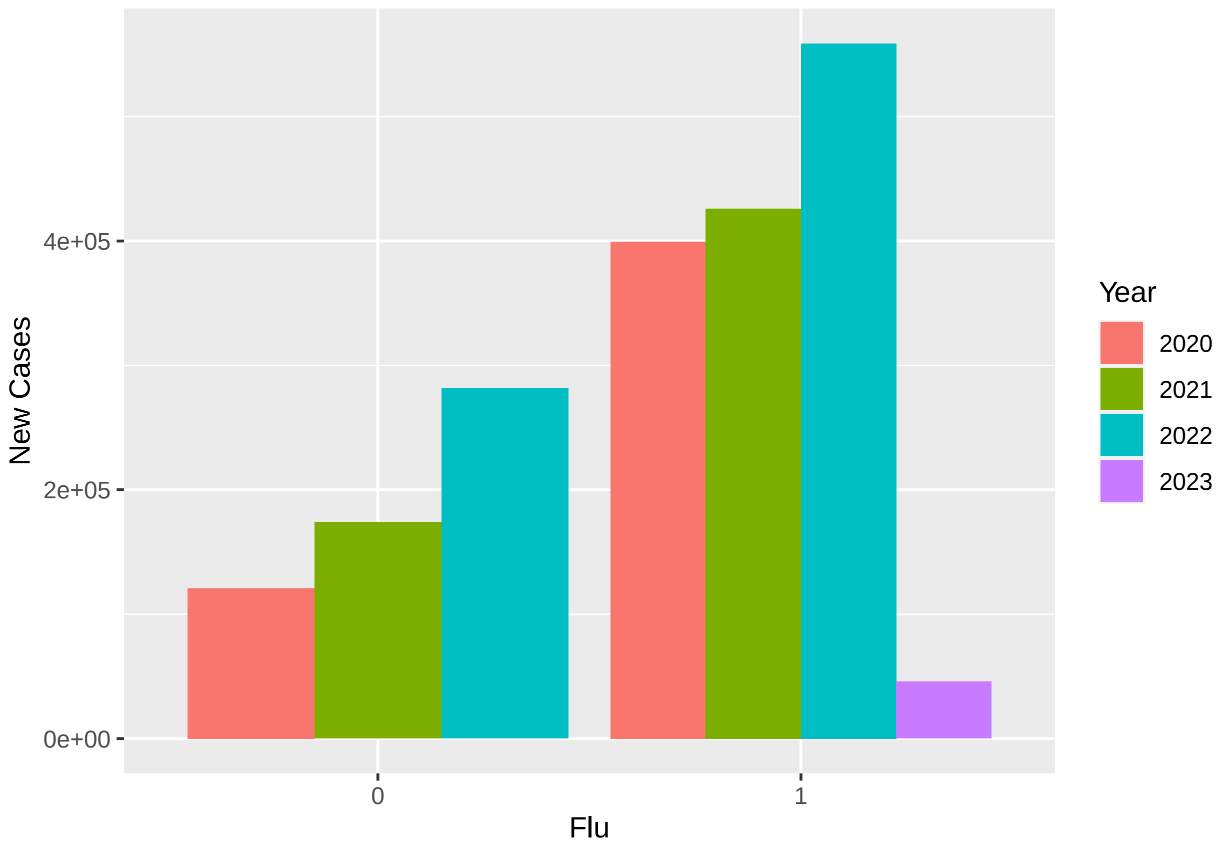
Interesting to see spikes closer to the end of the year in 2021 and the start of 2022. Let's continue exploring through various models and plots.

Visualize the amount of Cases and Deaths during the flu season and not during the flu season:

```
grouped_case = df %>%
  group_by(Year, Flu) %>%
  summarise(New_Cases = sum(New_Cases))
```

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

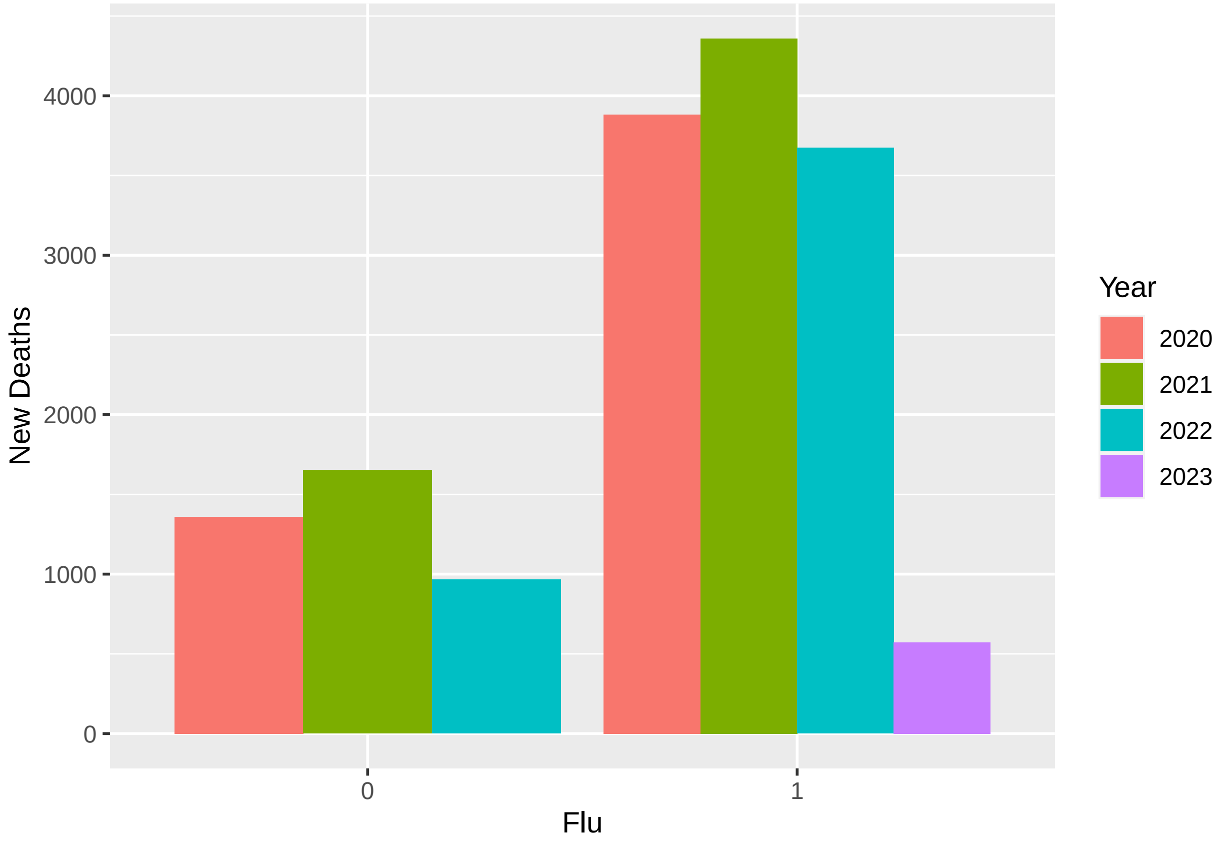
```
ggplot(grouped_case, aes(x = factor(Flu), y = New_Cases, fill = factor(Year))) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(x = "Flu", y = "New Cases", fill = "Year")
```



```
grouped_death= df %>%
  group_by(Year, Flu) %>%
  summarise(New_Deaths = sum(New_Deaths))
```

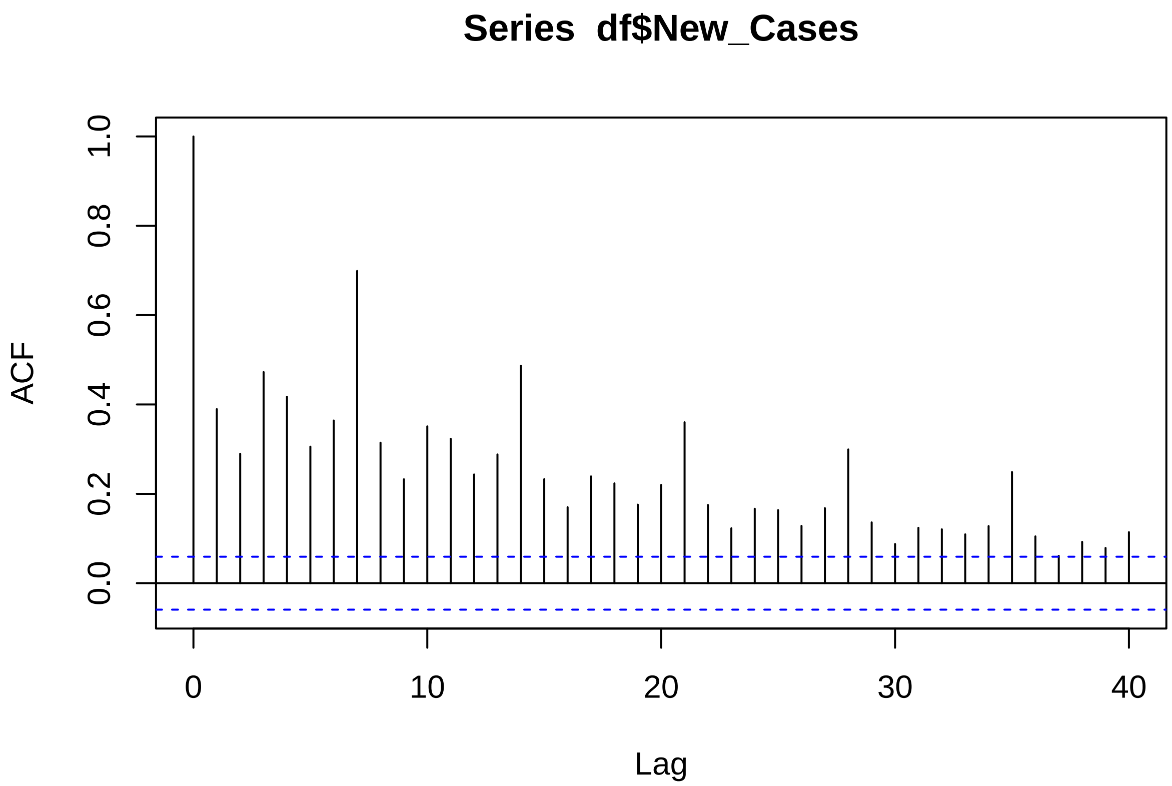
```
## 'summarise()' has grouped output by 'Year'. You can override using the
## '.groups' argument.
```

```
ggplot(grouped_death, aes(x = factor(Flu), y = New_Deaths, fill = factor(Year))) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(x = "Flu", y = "New Deaths", fill = "Year")
```

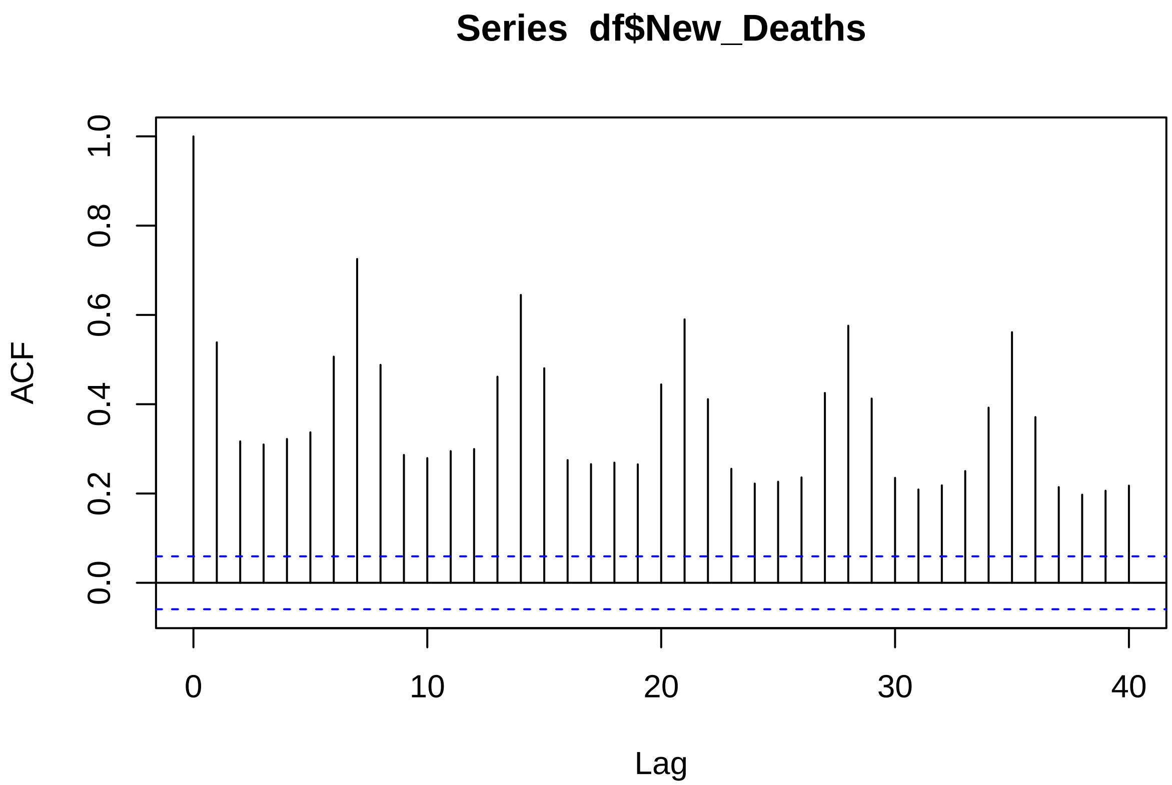


Plotting the autocorrelation to identify seasonality lag numbers. - There appears to be a strong weekly seasonal pattern with spiked at intervals of 7s.

```
acf(df$New_Cases, lag.max = 40)
```



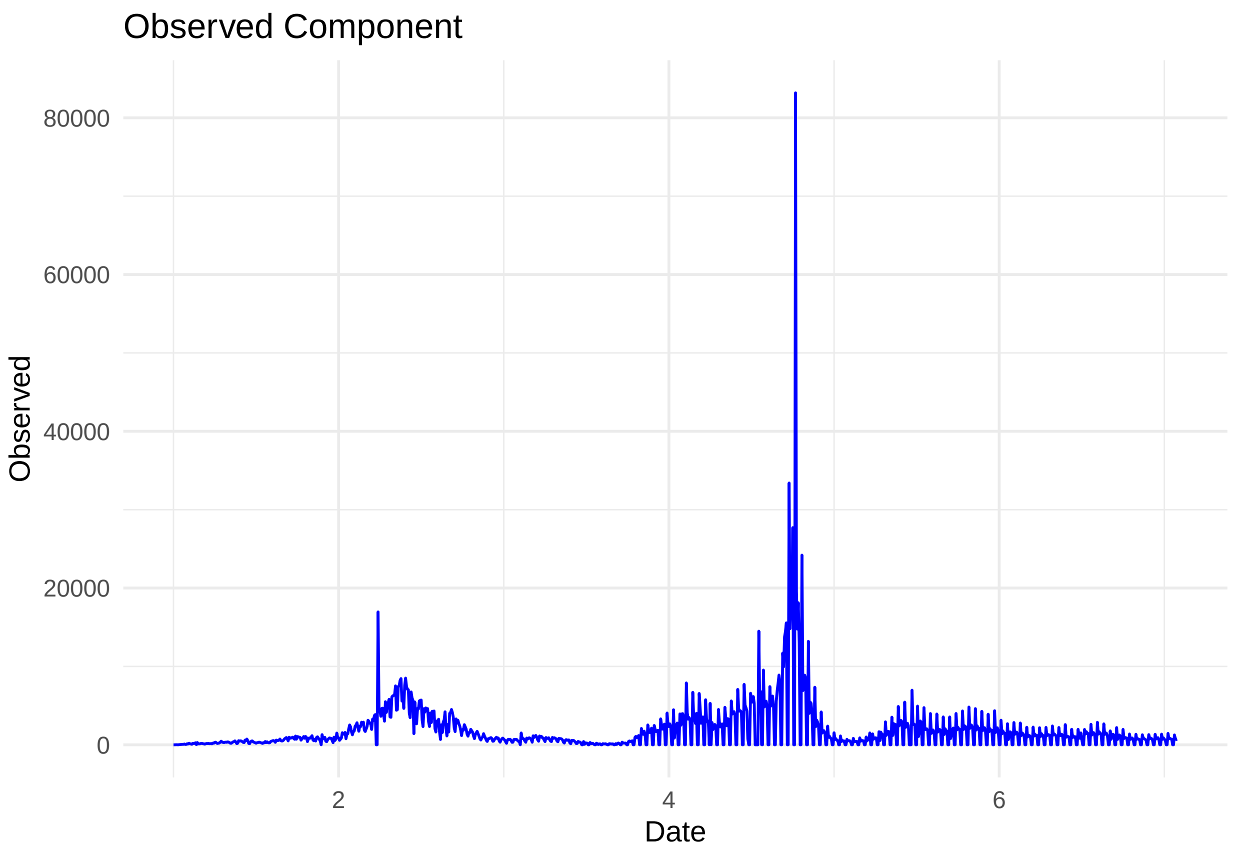
```
acf(df$New_Deaths, lag.max = 40)
```



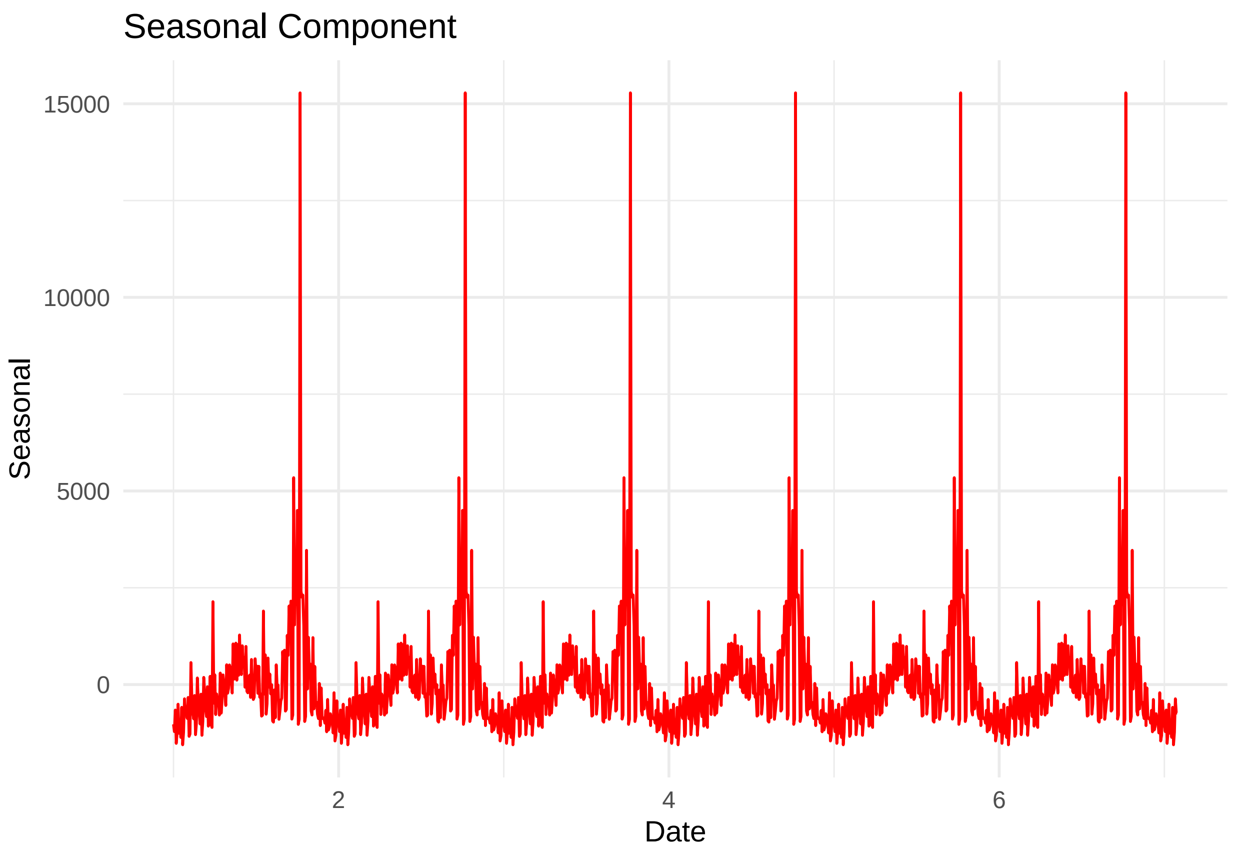
Decomposing the Season, Trend, and Residual + The variance ratio to see what % of the variances in the New Cases are explained by each component.

```r
decompose_plot <- function(df, col, season) {
  # Perform seasonal decomposition
  result <- decompose(ts(df[[col]], frequency = season), type = 'additive')

  # Create data frames for components
  df_observed <- data.frame(Date = time(result$x), Observed = as.numeric(result$x))
  df_seasonal <- data.frame(Date = time(result$seasonal), Seasonal = as.numeric(result$seasonal))
  df_trend <- data.frame(Date = time(result$trend), Trend = as.numeric(result$trend))
  df_random <- data.frame(Date = time(result$random), Random = as.numeric(result$random))

  # Plot each component
  plot_observed <- ggplot(df_observed, aes(x = Date, y = Observed)) +
    geom_line(color = "blue") +
    labs(title = "Observed Component") +
    theme_minimal()

  plot_seasonal <- ggplot(df_seasonal, aes(x = Date, y = Seasonal)) +
    geom_line(color = "red") +
    labs(title = "Seasonal Component") +
    theme_minimal()

  plot_trend <- ggplot(df_trend, aes(x = Date, y = Trend)) +
    geom_line(color = "green") +
    labs(title = "Trend Component") +
    theme_minimal()

  plot_random <- ggplot(df_random, aes(x = Date, y = Random)) +
    geom_line(color = "orange") +
    labs(title = "Random Component") +
    theme_minimal()

  # Print the plots
  print(plot_observed)
  print(plot_seasonal)
  print(plot_trend)
  print(plot_random)

  # Calculate variance ratios
  var_seasonal <- var(result$seasonal, na.rm = TRUE)
  var_trend <- var(result$trend, na.rm = TRUE)
  var_residual <- var(result$random, na.rm = TRUE)
  var_original <- var(result$x, na.rm = TRUE)

  var_ratio_seasonal <- var_seasonal / var_original
  var_ratio_trend <- var_trend / var_original
  var_ratio_residual <- var_residual / var_original

  cat("Variance ratio for seasonal component:", var_ratio_seasonal, "\n")
  cat("Variance ratio for trend component:", var_ratio_trend, "\n")
  cat("Variance ratio for residual component:", var_ratio_residual, "\n")
}

decompose_plot(df, "New_Cases", 180)
```

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

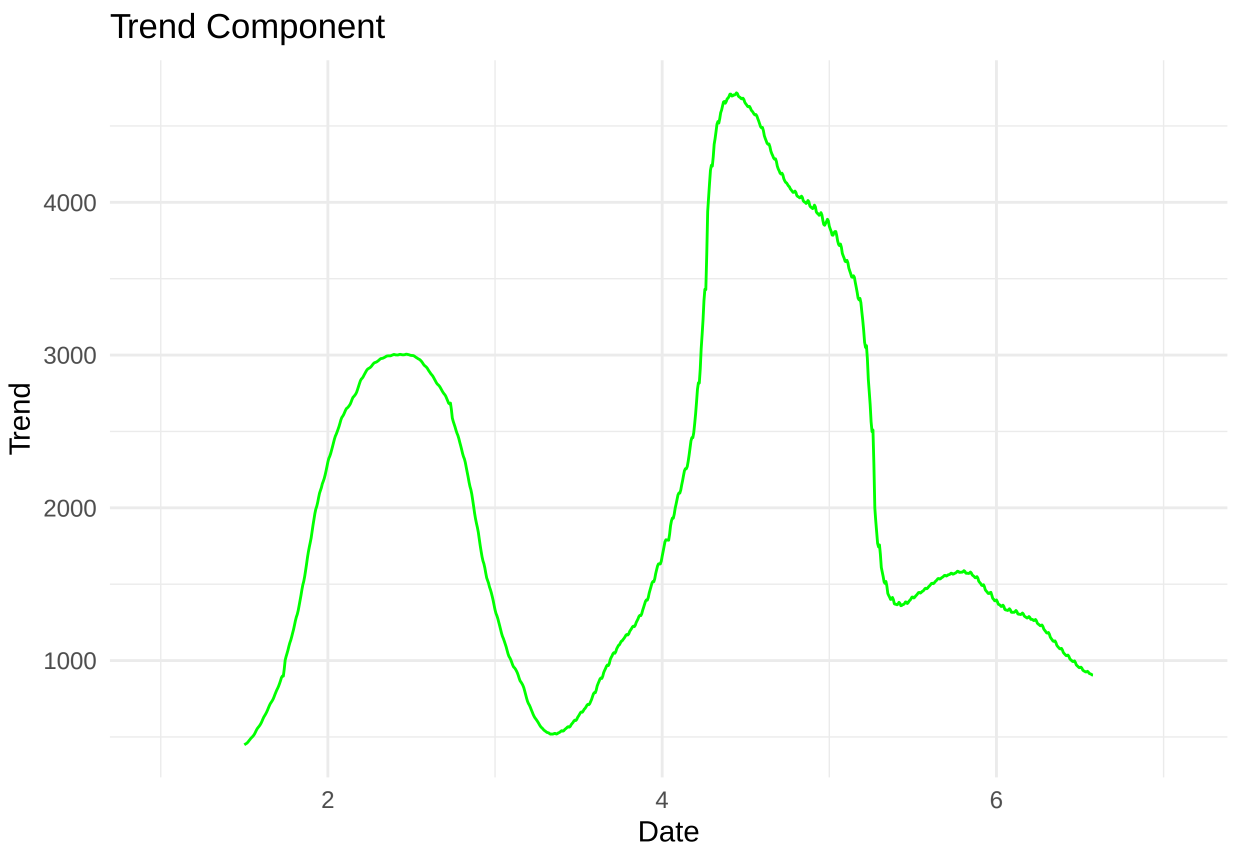


```
## Don’t know how to automatically pick scale for object of type <ts>. Defaulting
## to continuous.
```



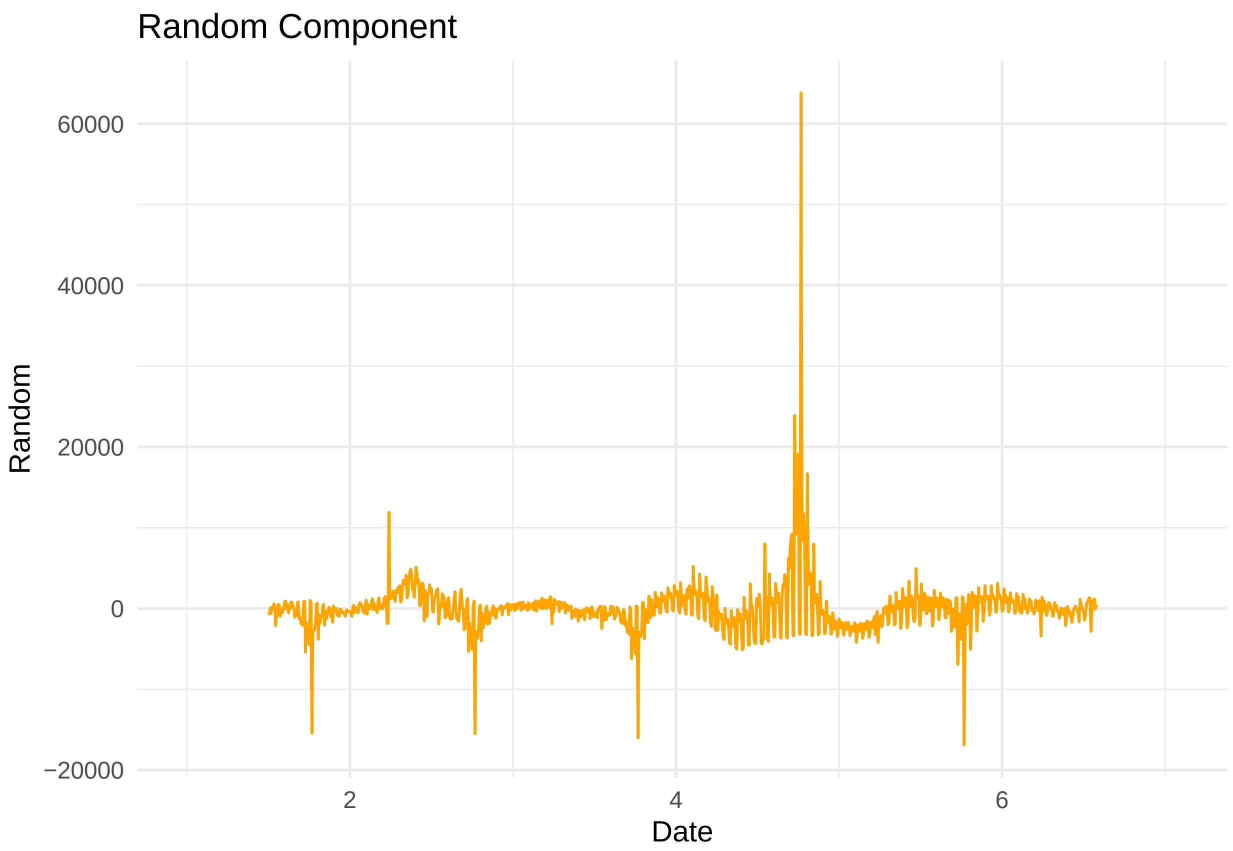
```
## Don't know how to automatically pick scale for object of type <ts>. Defaulting
## to continuous.
```

```
## Warning: Removed 180 rows containing missing values ('geom_line()').
```



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

```
## Warning: Removed 180 rows containing missing values ('geom_line()').
```

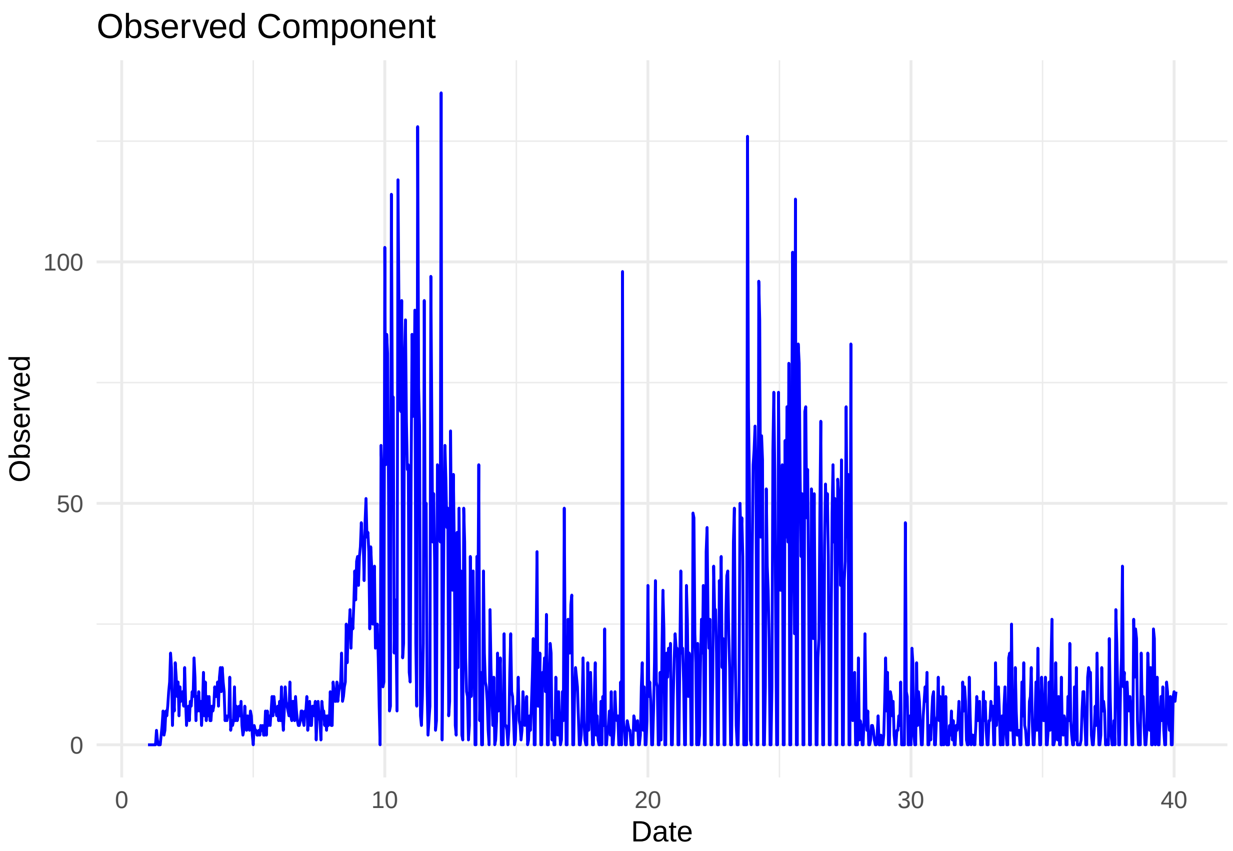


```
## Variance ratio for seasonal component: 0.1767593
## Variance ratio for trend component: 0.1083841
## Variance ratio for residual component: 0.8402315
```

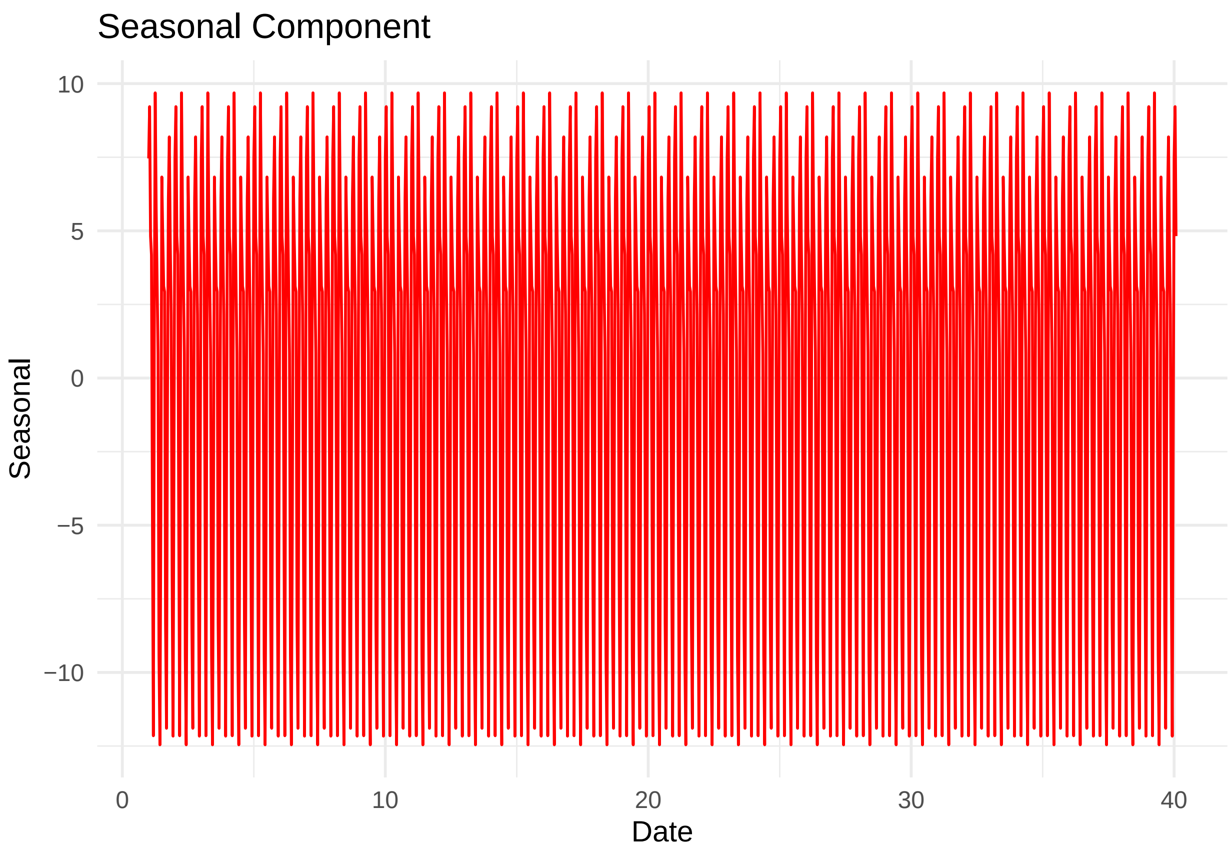
Decomposing the Season, Trend, and Residual + The variance ratio to see what % of the variances in the New Deaths are explained by each component.

```
decompose_plot(df, "New_Deaths", 28)
```

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

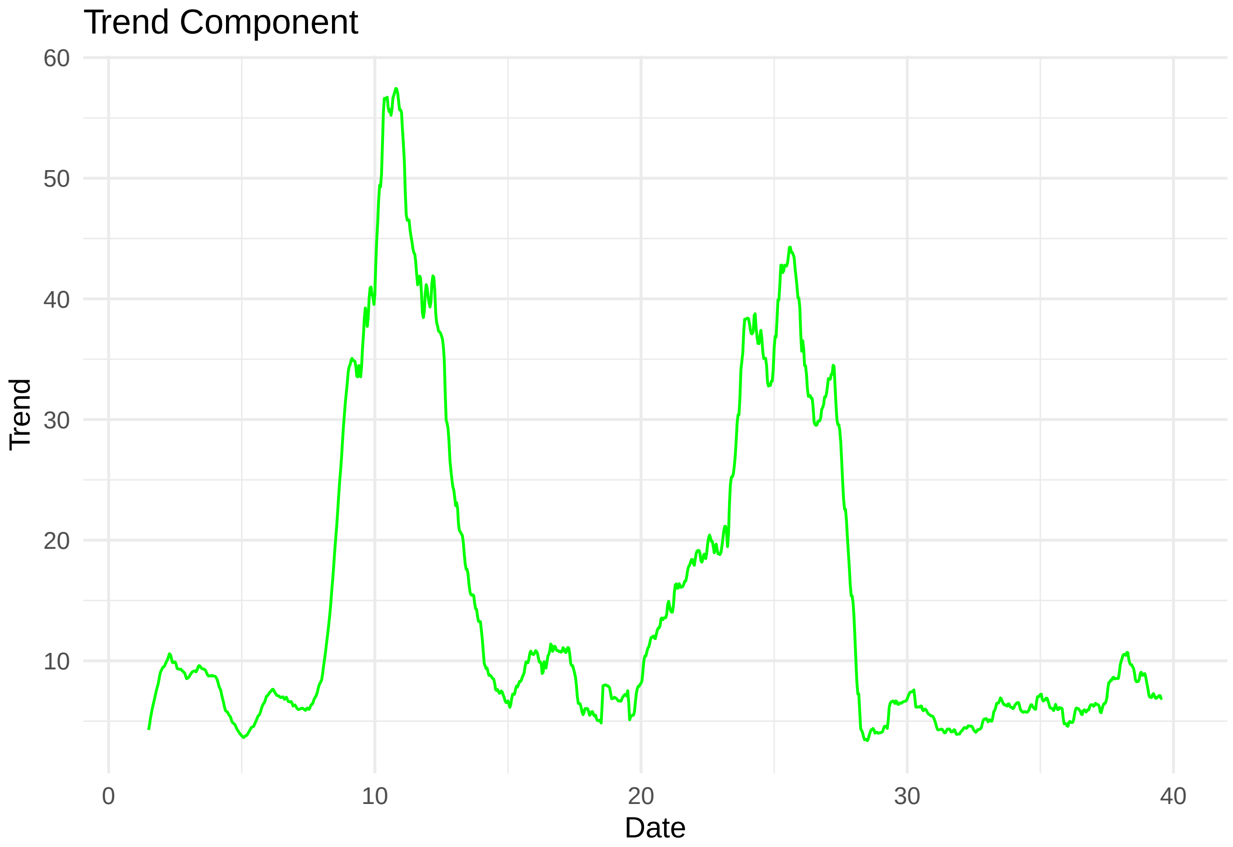


```
## Don’t know how to automatically pick scale for object of type <ts>. Defaulting
## to continuous.
```



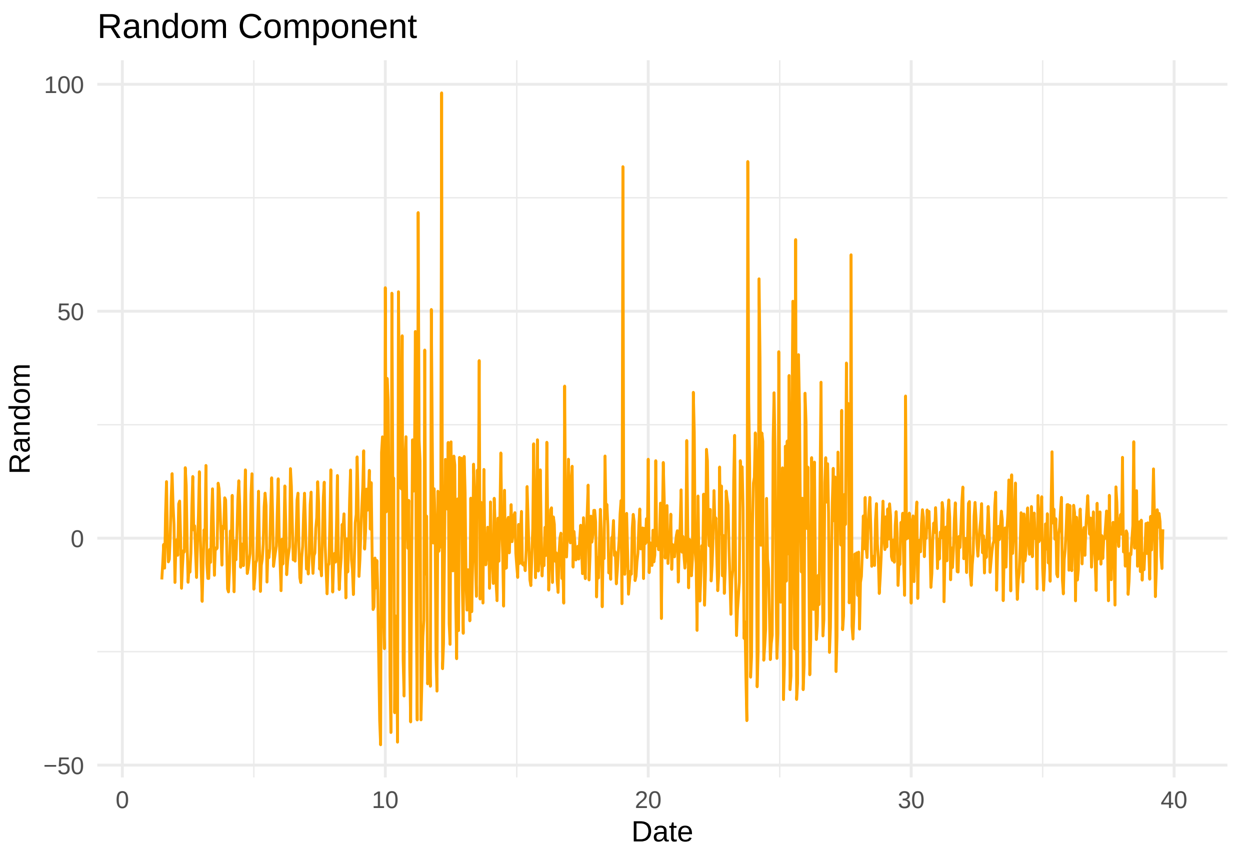
```
## Don’t know how to automatically pick scale for object of type <ts>. Defaulting
## to continuous.
```

```
## Warning: Removed 28 rows containing missing values (‘geom_line()‘).
```



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

```
## Warning: Removed 28 rows containing missing values (‘geom_line()‘).
```



```
## Variance ratio for seasonal component: 0.1157766
## Variance ratio for trend component: 0.423583
## Variance ratio for residual component: 0.4587932
```

Modeling

Hypothesis testing 1: is there a difference in the mean daily cases during Flu season vs not during flu season? Result: Reject the null hypothesis, there is a difference in mean daily cases. (alpha .05)

```
library(BSDA)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'BSDA'
```

```
## The following object is masked from 'package:datasets':
##
##     Orange
```

```
perform_z_test <- function(data, variable) {
  sd_flu <- sd(data[[variable]][data$Flu == 1])
  sd_non_flu <- sd(data[[variable]][data$Flu == 0])
```

```r
  z_test <- z.test(x = data[[variable]][data$Flu == 1],
                   y = data[[variable]][data$Flu == 0],
                   sigma.x = sd_flu, sigma.y = sd_non_flu)

  cat("Mean Daily Flu Season COVID:", mean(data[[variable]][data$Flu == 1]), "\n")
  cat("Mean Daily Non-Flu Season COVID:", mean(data[[variable]][data$Flu == 0]), "\n")
  cat("Z-statistic:", z_test$statistic, "\n")
  cat("P-value:", z_test$p.value, "\n")

  # Return the z-test result
  return(z_test)
}

z_test_result <- perform_z_test(df, "New_Cases")
```

```
## Mean Daily Flu Season COVID: 2619.456
## Mean Daily Non-Flu Season COVID: 1049.834
## Z-statistic: 7.152869
## P-value: 8.498279e-13
```

Hypothesis testing 1.2: is there a difference in the mean daily deaths during Flu season vs not during flu season? Result: Reject the null hypothesis, there is a difference in mean daily deaths (alpha .05)

```r
z_test_result <- perform_z_test(df, "New_Deaths")
```

```
## Mean Daily Flu Season COVID: 22.87363
## Mean Daily Non-Flu Season COVID: 7.245902
## Z-statistic: 13.47543
## P-value: 2.182115e-41
```

Hypothesis testing 2: is there a difference in the mean monthly cases during Flu season vs not during flu season? Result: Fail to reject the null hypothesis, there is no difference in mean monthly cases. (alpha .05)

```r
perform_t_test <- function(data, variable) {
  sd_flu <- sd(data[[variable]][data$Flu == 1])
  sd_non_flu <- sd(data[[variable]][data$Flu == 0])

  t_test <- t.test(x = data[[variable]][data$Flu == 1],
                   y = data[[variable]][data$Flu == 0],
                   sigma.x = sd_flu, sigma.y = sd_non_flu)

  cat("Mean Daily Flu Season COVID:", mean(data[[variable]][data$Flu == 1]), "\n")
  cat("Mean Daily Non-Flu Season COVID:", mean(data[[variable]][data$Flu == 0]), "\n")
  cat("Z-statistic:", t_test$statistic, "\n")
  cat("P-value:", t_test$p.value, "\n")

  return(t_test)
}

t_test_result <- perform_t_test(df_month, "Total_Cases")
```

```
## Mean Daily Flu Season COVID: 75274.89
## Mean Daily Non-Flu Season COVID: 32019.94
## Z-statistic: 1.991234
## P-value: 0.06017881
```

Hypothesis testing 2.1: is there a difference in the mean monthly deaths during Flu season vs not during flu season? Result: Fail to reject the null hypothesis, there is no difference in mean monthly deaths (alpha .05)

```
t_test_result <- perform_t_test(df_month, "Total_Deaths")
```

```
## Mean Daily Flu Season COVID: 657.3158
## Mean Daily Non-Flu Season COVID: 221
## Z-statistic: 3.77807
## P-value: 0.001278093
```

Hypothesis testing 3: is there a difference in the mean daily cases during peak Flu season vs flu season? Result: Fail to reject the null hypothesis, there is no difference in mean daily cases. (alpha .05)

```
perform_z_test <- function(data, variable) {
  sd_flu <- sd(data[[variable]][data$PeakFlu == 1])
  sd_non_flu <- sd(data[[variable]][data$Flu == 1])

  z_test <- z.test(x = data[[variable]][data$PeakFlu == 1],
                   y = data[[variable]][data$Flu == 1],
                   sigma.x = sd_flu, sigma.y = sd_non_flu)

  cat("Mean Daily Peak Flu Season COVID:", mean(data[[variable]][data$PeakFlu == 1]), "\n")
  cat("Mean Daily Flu Season COVID:", mean(data[[variable]][data$Flu == 1]), "\n")
  cat("Z-statistic:", z_test$statistic, "\n")
  cat("P-value:", z_test$p.value, "\n")

  return(z_test)
}

z_test_result <- perform_z_test(df, "New_Cases")
```

```
## Mean Daily Peak Flu Season COVID: 3226.97
## Mean Daily Flu Season COVID: 2619.456
## Z-statistic: 1.329193
## P-value: 0.1837842
```

Hypothesis testing 3.2: is there a difference in the mean daily deaths during peak Flu season vs flu season? Result: Reject the null hypothesis, there is a difference in mean daily deaths (alpha .05)

```
z_test_result <- perform_z_test(df, "New_Deaths")
```

```
## Mean Daily Peak Flu Season COVID: 27.85185
## Mean Daily Flu Season COVID: 22.87363
## Z-statistic: 2.33861
## P-value: 0.0193556
```

Hypothesis testing 4: is there a difference in the mean monthly cases during peak Flu season vs flu season? Result: Fail to reject the null hypothesis, there is no difference in mean monthly cases. (alpha .05)

```r
perform_t_test <- function(data, variable) {
  sd_flu <- sd(data[[variable]][data$PeakFlu == 1])
  sd_non_flu <- sd(data[[variable]][data$Flu == 1])

  t_test <- t.test(x = data[[variable]][data$PeakFlu == 1],
                   y = data[[variable]][data$Flu == 1],
                   sigma.x = sd_flu, sigma.y = sd_non_flu)

  cat("Mean Daily Flu Season COVID:", mean(data[[variable]][data$PeakFlu == 1]), "\n")
  cat("Mean Daily Non-Flu Season COVID:", mean(data[[variable]][data$Flu == 1]), "\n")
  cat("T-statistic:", t_test$statistic, "\n")
  cat("P-value:", t_test$p.value, "\n")

  return(t_test)
}

t_test_result <- perform_t_test(df_month, "Total_Cases")
```

```
## Mean Daily Flu Season COVID: 96809.11
## Mean Daily Non-Flu Season COVID: 75274.89
## T-statistic: 0.4770576
## P-value: 0.6414533
```

Hypothesis testing 4.1: is there a difference in the mean monthly deaths during peak Flu season vs flu season? Result: Fail to reject the null hypothesis, there is no difference in mean monthly deaths (alpha .05)

```r
t_test_result <- perform_t_test(df_month, "Total_Deaths")
```

```
## Mean Daily Flu Season COVID: 835.5556
## Mean Daily Non-Flu Season COVID: 657.3158
## T-statistic: 0.8325477
## P-value: 0.4185356
```

Linear Model 1: is the slope between flu season = 1 and the daily cases 0? I.e., Is there no linear relationship between flu season and daily cases. Result: Reject the null hypothesis, the slope is not 0. (alpha .05)

```r
X <- df$Flu
Y <- df$New_Cases
model <- lm(Y ~ X)

summary(model)
```

```
##
## Call:
## lm(formula = Y ~ X)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -2619  -1376   -656    623  80568
```

```
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1049.8      154.6   6.792 1.82e-11 ***
## X             1569.6      218.9   7.170 1.37e-12 ***
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## Residual standard error: 3622 on 1093 degrees of freedom
## Multiple R-squared:  0.04493,    Adjusted R-squared:  0.04405
## F-statistic: 51.42 on 1 and 1093 DF,  p-value: 1.374e-12
```

Linear Model 2: is the slope between flu season = 1 and the daily deaths 0? I.e., Is there no linear relationship between flu season and daily deaths Result: Reject the null hypothesis, the slope is not 0. (alpha .05)

```r
X <- df$Flu
Y <- df$New_Deaths
model <- lm(Y ~ X)

summary(model)
```

```
##
## Call:
## lm(formula = Y ~ X)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -22.874  -8.874  -3.246   4.754 112.126
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   7.2459     0.8171   8.868   <2e-16 ***
## X            15.6277     1.1571  13.506   <2e-16 ***
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## Residual standard error: 19.14 on 1093 degrees of freedom
## Multiple R-squared:  0.143,  Adjusted R-squared:  0.1422
## F-statistic: 182.4 on 1 and 1093 DF,  p-value: < 2.2e-16
```

Linear Model 3: is the slope between flu season = 1 and the monthly cases 0? I.e., Is there no linear relationship between flu season and monthly cases. Result: Fail to reject the null hypothesis, the slope is 0. (alpha .05)

```r
X <- df_month$Flu
Y <- df_month$Total_Cases
model <- lm(Y ~ X)

summary(model)
```

```
##
## Call:
```

```
## lm(formula = Y ~ X)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -73863 -40834  -7739  19047 320690
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    32020      15957   2.007   0.0526 .
## X              43255      22268   1.943   0.0602 .
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## Residual standard error: 67700 on 35 degrees of freedom
## Multiple R-squared:  0.09732,    Adjusted R-squared:  0.07153
## F-statistic: 3.773 on 1 and 35 DF,  p-value: 0.06016
```

Linear Model 4: is the slope between flu season = 1 and the monthly deaths 0? I.e., Is there no linear relationship between flu season and monthly deaths Result: Reject the null hypothesis, the slope is not 0. (alpha .05)

```r
X <- df_month$Flu
Y <- df_month$Total_Deaths
model <- lm(Y ~ X)

summary(model)
```

```
##
## Call:
## lm(formula = Y ~ X)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -614.3 -109.0  -30.0   93.0 1090.7
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   221.00      84.98   2.601  0.01354 *
## X             436.32     118.58   3.679  0.00078 ***
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## Residual standard error: 360.5 on 35 degrees of freedom
## Multiple R-squared:  0.2789, Adjusted R-squared:  0.2583
## F-statistic: 13.54 on 1 and 35 DF,  p-value: 0.0007803
```

Conclusion & Bias

Across all tests, the monthly average cases and death models using T-tests had a lower test statistic than the daily average case and death models using Z-tests. Due to the decreased sample size when testing the difference in means between Monthly data, the statistical power was weaker. I hypothesize that due to the smaller monthly sample size, the T-test was less likely to reject the null hypothesis even if the null hypothesis is false. The wider variability in monthly data compared to daily data could also contribute to the differences in statistical significance.

The one hypothesis test where there was a statistically significant difference in monthly averages was comparing the mean deaths in flu season months to non-flu season months. The monthly average cases during flu season compared to non-flu season had a p-value of .06, due to our given alpha of .05, we failed to reject the null hypothesis. However, if alpha was .1, then the differences in monthly average cases would have also been statistically significant.

Upon exploring the four linear regression models, with alpha of .05, all slopes are positive and are statistically significant except for flu season and total monthly cases. The highest R-Squared value is 27% looking at monthly deaths and flu seasons.

Viewing the results from the hypothesis testing and the linear regression models, it is clear that the flu season has a significant impact on the number of cases and deaths. The peak flu seasons have a statistically significant higher average deaths than the flu season alone, however, no other tests comparing peak flu season with flu season resulted in statistical significance.

As mentioned in the abstract, to counteract my personal biases regarding the impact of the flu seasons, I scrutinized the seasonality patterns using both time series decomposition and testing of means across flu seasons. The test to compare peak flu season and all flu season was an additional effort to analyze the relationships between COVID cases and deaths with the flu season. Additionally, it is important to acknowledge the bias in choosing Wisconsin as the proxy state for a geographic area with four seasons. While WI has the temperature patterns I wanted to study, it may not have had reporting accuracy or standards to ensure complete case or death information. At the beginning of COVID-19, WI may not have had the necessary systems or technology to accurately identify and report COVID-19.

Across any US states, there may have also been incentives to manipulate reporting of COVID diagnoses. An interesting expansion to this study could analyze the shifts in seasonality across various years.

```r
sessionInfo()
```

```
## R version 4.3.1 (2023-06-16)
## Platform: x86_64-apple-darwin20 (64-bit)
## Running under: macOS Ventura 13.6.1
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/lib/libRlapack.dylib;  LAPACK
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## time zone: America/Chicago
## tzcode source: internal
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] BSDA_1.2.2      lattice_0.21-8  lubridate_1.9.3 forcats_1.0.0
##  [5] stringr_1.5.1   dplyr_1.1.3     purrr_1.0.2     readr_2.1.5
##  [9] tidyr_1.3.1     tibble_3.2.1    ggplot2_3.4.3   tidyverse_2.0.0
##
## loaded via a namespace (and not attached):
##  [1] utf8_1.2.3        generics_0.1.3    class_7.3-22      stringi_1.8.3
##  [5] hms_1.1.3         digest_0.6.35     magrittr_2.0.3    evaluate_0.23
##  [9] grid_4.3.1        timechange_0.3.0  fastmap_1.1.1     e1071_1.7-14
## [13] fansi_1.0.4       scales_1.2.1      cli_3.6.1         rlang_1.1.1
## [17] crayon_1.5.2      bit64_4.0.5       munsell_0.5.0     withr_2.5.0
## [21] yaml_2.3.8        tools_4.3.1       parallel_4.3.1    tzdb_0.4.0
## [25] colorspace_2.1-0  curl_5.2.1        vctrs_0.6.3       R6_2.5.1
## [29] proxy_0.4-27      lifecycle_1.0.3   bit_4.0.5         vroom_1.6.5
## [33] pkgconfig_2.0.3   pillar_1.9.0      gtable_0.3.4      glue_1.6.2
## [37] xfun_0.42         tidyselect_1.2.0  highr_0.10        rstudioapi_0.15.0
## [41] knitr_1.45        farver_2.1.1      htmltools_0.5.7   rmarkdown_2.26
## [45] labeling_0.4.3    compiler_4.3.1
```