# Forecasting Candy Production in the US

Rutwik Guntoorkar

12/07/2022

### Introduction

- Sweets, chocolates, and candy are universally enjoyed. In the US, there are holidays themed around giving candy! All this consumption first needs production. The dataset below shows the monthly production of candy in the US. The industrial production index measures the actual output of all relevant establishments in the United States, regardless of ownership, but not those in U.S. territories.

### Data Source

- Link: https://fred.stlouisfed.org/series/IPG3113N

### Data Dictionary

- Date: Year, Month, and Date during with the data was recorded
- IPG3113N: Production Index for Candy in the US

## Question and Hypothesis

### Question

- What will be the best method to forecast the given time series data?

### Hypothesis

- Expanding our knowledge from previous forecasting techniques, the modern ANOVA method might give us the best forecast for time series.
- We can check this hypothesis based on the accuracy of each model that we can check below.

```r
library(fpp)
```

```
## Loading required package: forecast

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

## Loading required package: fma

## Loading required package: expsmooth

## Loading required package: lmtest

## Loading required package: zoo

##
## Attaching package: 'zoo'
```

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

## Loading required package: tseries
```

```r
library(fpp2)
```

```
## -- Attaching packages ---------------------------------------------- fpp2 2.5 --

## v ggplot2 3.4.0

##

##
## Attaching package: 'fpp2'

## The following objects are masked from 'package:fpp':
##
##     ausair, ausbeer, austa, austourists, debitcards, departures,
##     elecequip, euretail, guinearice, oil, sunspotarea, usmelec
```

```r
library(TTR)
```

```
## Warning: package 'TTR' was built under R version 4.2.3
```

```r
library(ggplot2)
library(readr)
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
```

```r
library(forecast)
library(tidyverse)
```

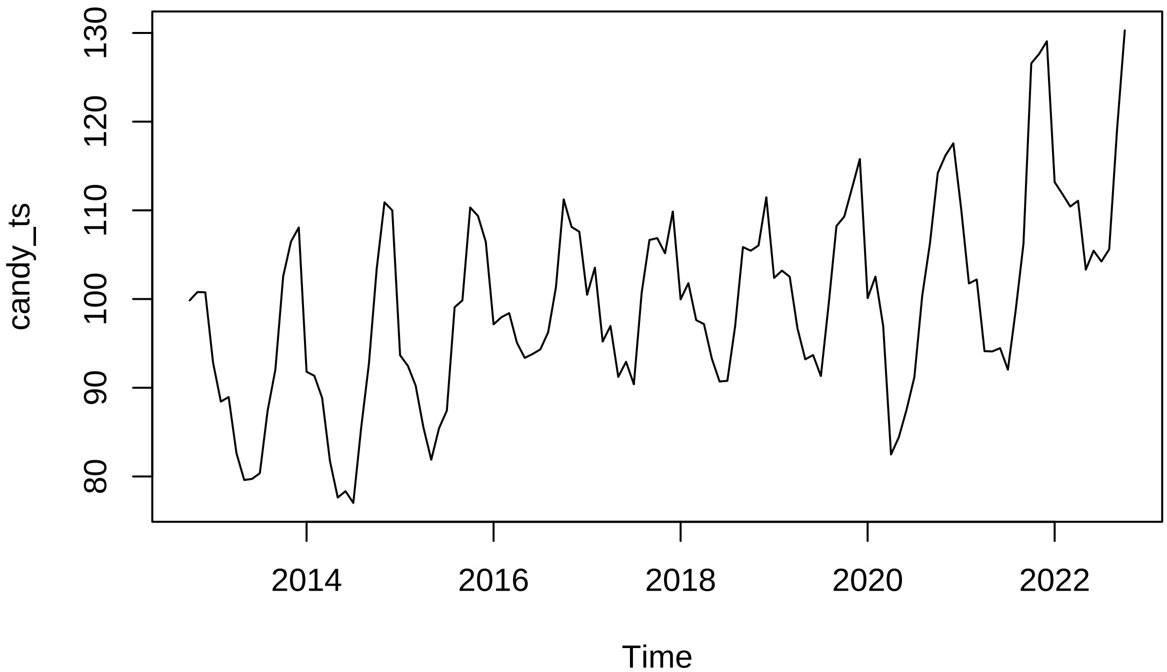
```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --

## v tibble  3.1.8     v stringr 1.5.0
## v tidyr   1.3.0     v forcats 1.0.0
## v purrr   1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(readr)
IPG3113N <- read_csv("/Users/rutwik/Desktop/Projects/US Candy Production/IPG3113N.csv")
```

```
## Rows: 121 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl  (1): IPG3113N
## date (1): DATE
##
```

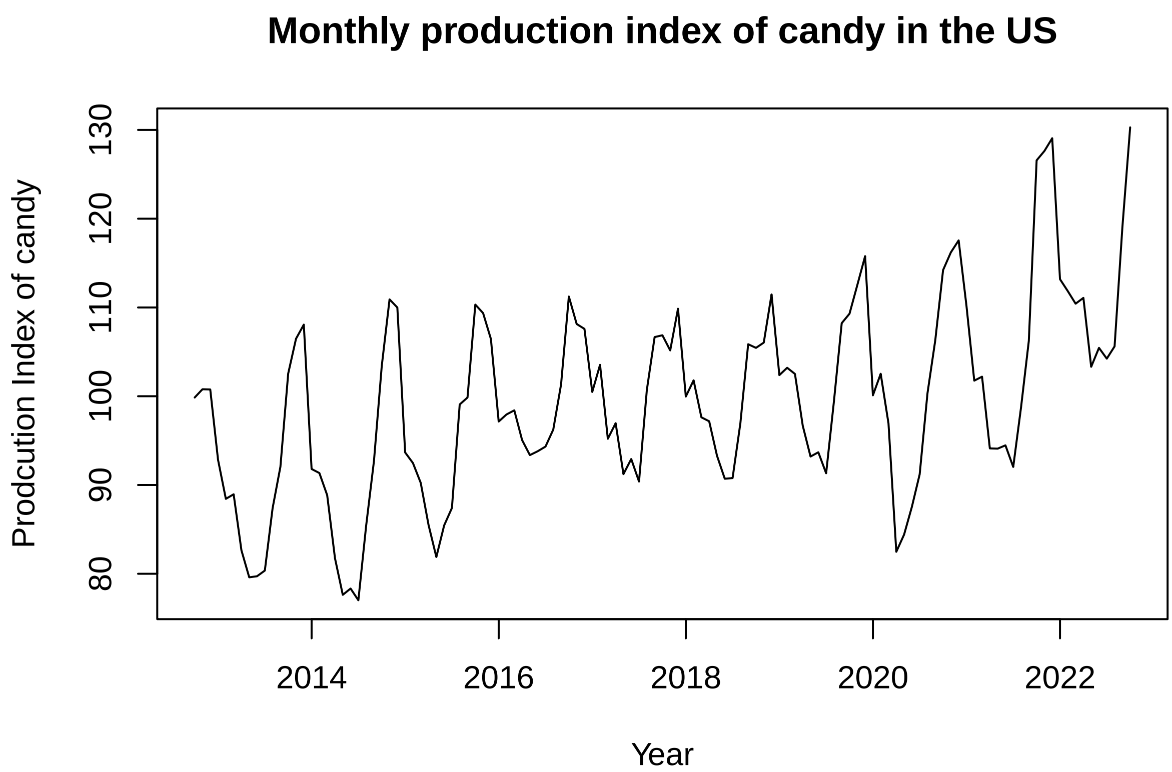
```
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
candy_ts <- ts(IPG3113N$IPG3113N,frequency = 12,start=c(2012,10))
candy_ts_original <- ts(IPG3113N$IPG3113N,frequency = 12,start=c(2012,10))
plot(candy_ts)
```




### Plot and Inference

```r
plot(candy_ts_original, main = 'Monthly production index of candy in the US', xlab = 'Year', ylab = 'Pr
```



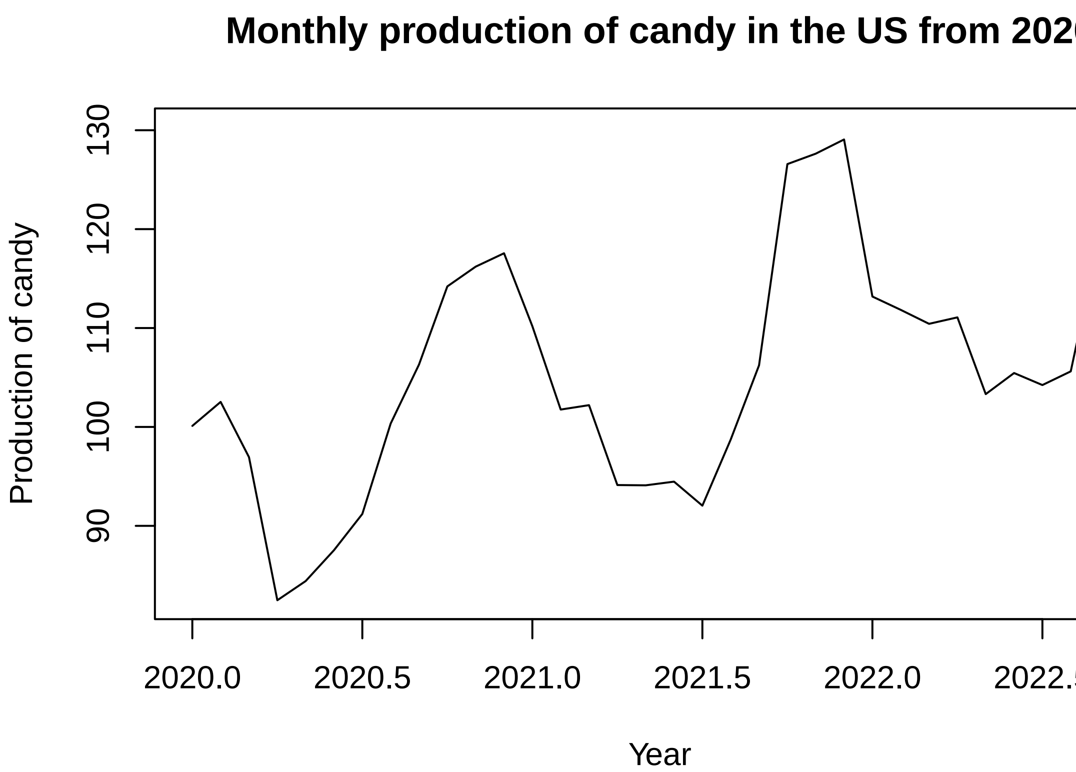

**Time Series Plot**

- We start with plotting the time series to visualise and understand the data.

**Initial Observations**

- The data from 2012 has seasonal variation and is peaking every November and December every year.
- This is because of the holiday season every year that has Thanksgiving and Christmas.
- However, from 2020, the data has an increasing trend and seasonal component.
- To explore this idea more, we consider a window starting from 2020 and considering two years of data will be good enough for a proper forecast.

```
candy_ts <- window(candy_ts_original, start = 2020)
plot(candy_ts, main = 'Monthly production of candy in the US from 2020', xlab = 'Year', ylab = 'Product
```




**Considering only a window**

- Considering the window function, the plot has both trend and seasonality.
- Forecasting this data will be more accurate as it is the recent data, and there is a high chance that the future data will have the same trend and seasonality.
- Further analysis of the data will be done considering this data set.

## Central Tendency

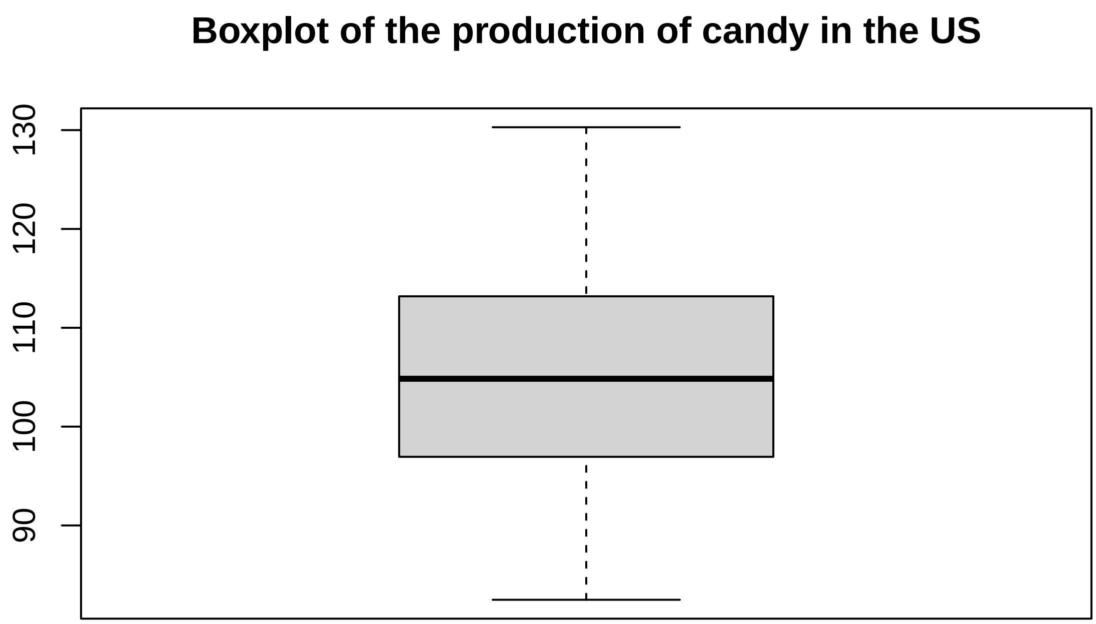
```
summary(candy_ts)
```

**Min, max, mean, median, 1st and 3rd Quartile values of the times series**

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   82.48   97.39  104.84  105.63  112.85  130.29
```

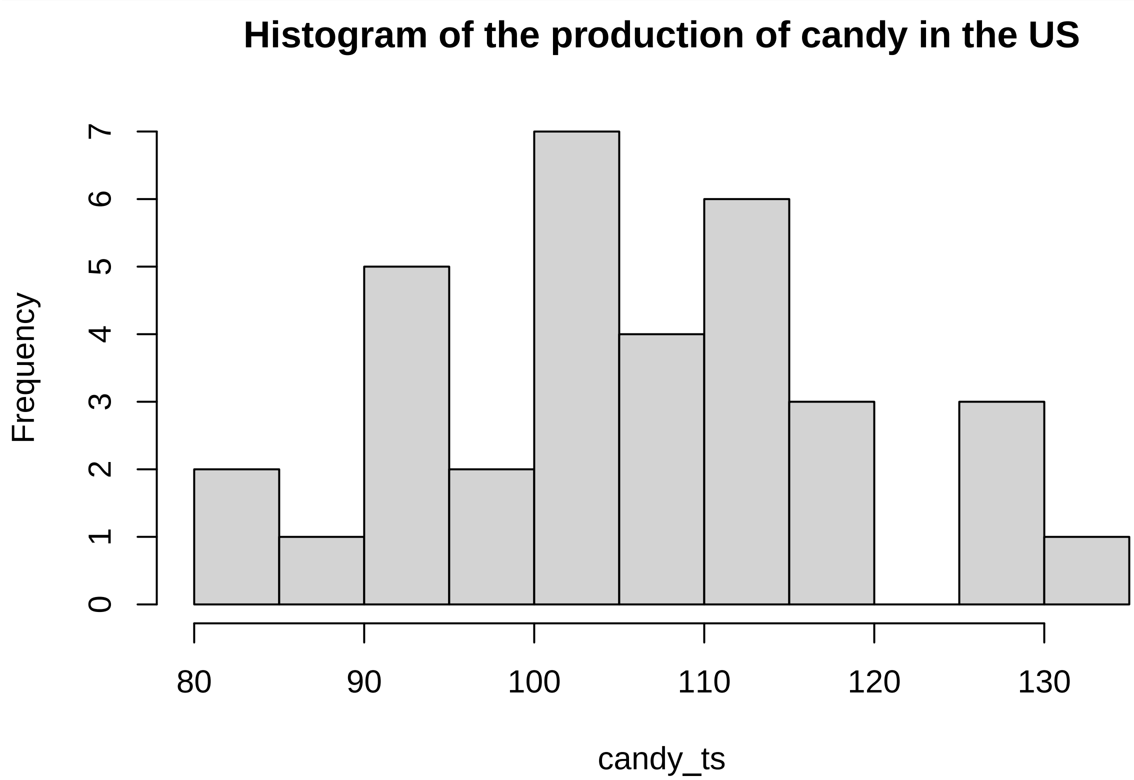
- The summary function above gives the min, max, mean, median, 1st and 3rd Quartile values of the times series.

```
boxplot(candy_ts, main = 'Boxplot of the production of candy in the US')
```

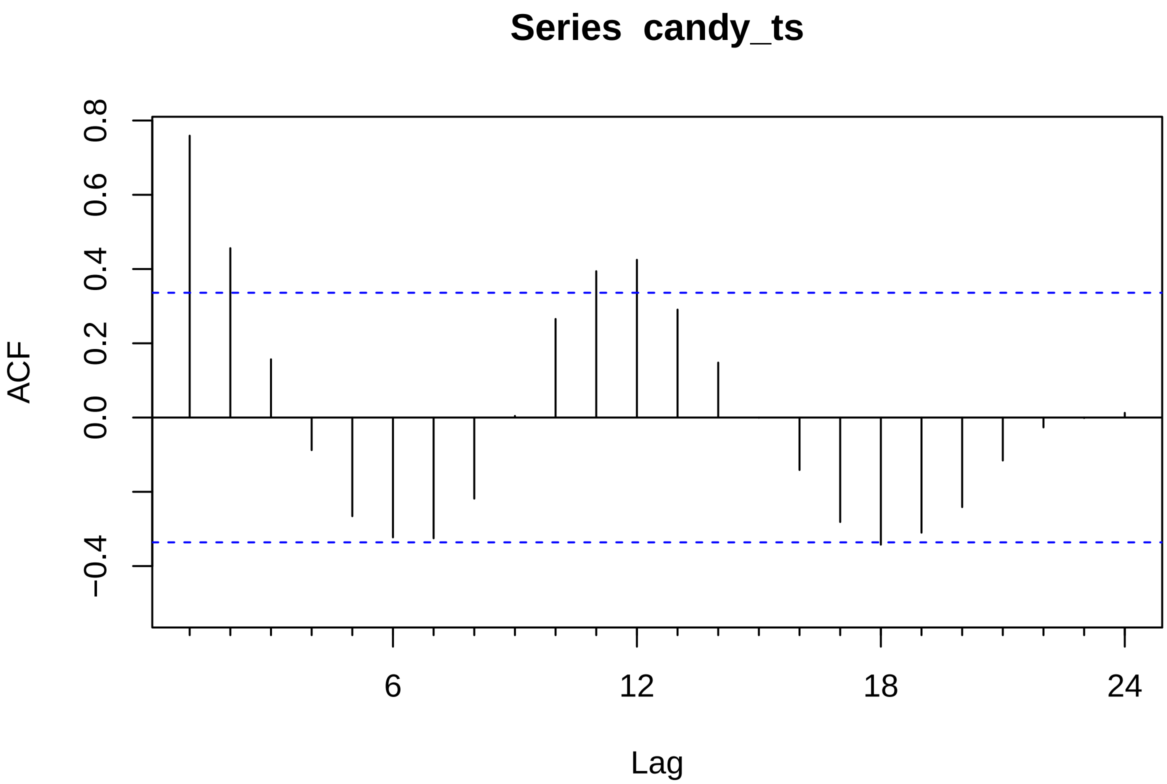



**Box Plot**

```
hist(candy_ts, main = 'Histogram of the production of candy in the US')
```




```
Acf(candy_ts)
```

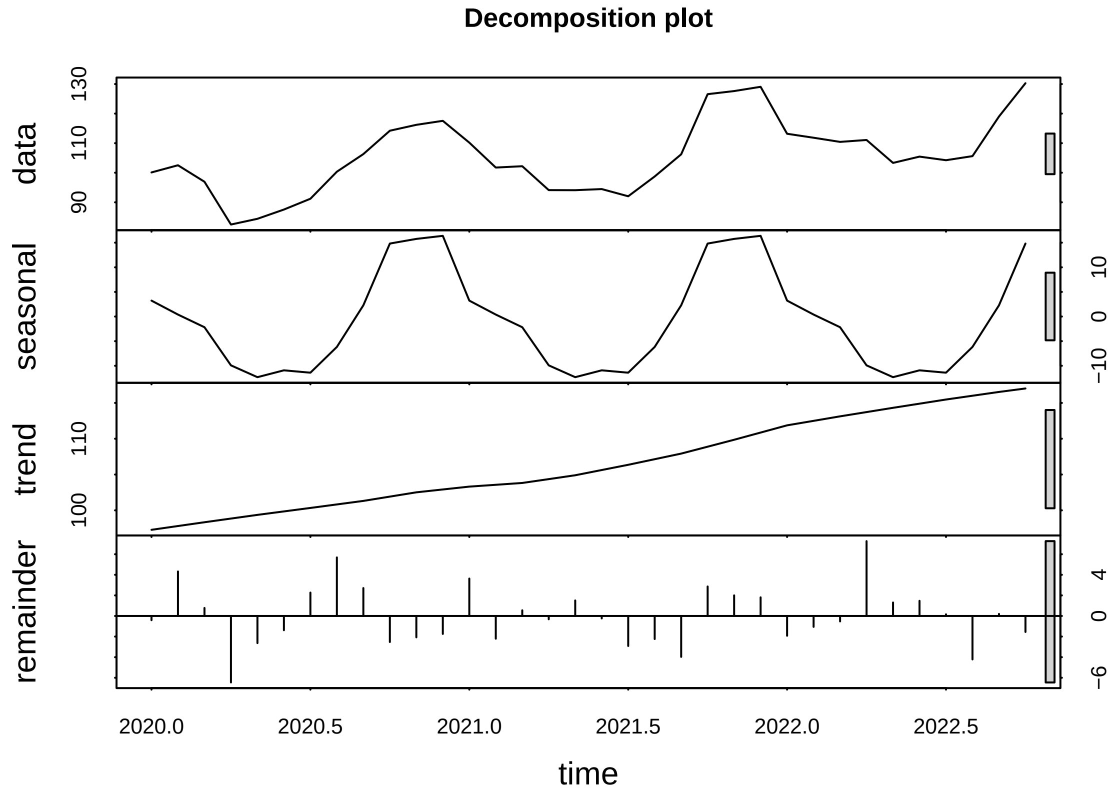



#### Observations and Inferences

- The boxplot shows that there are no outliers in the data.
- The data has a mean of 105.63 and doesn't look to have a proper normal distribution.
- The median is in between the 1st and 3rd quartile and is not specifically towards one of them.
- From the summary, we can also see that the median value is less than the mean for the time series.
- This means that the data is right-skewed. This can be justified by seeing the histogram above as well.
- The ACF plot shows a strong trend and seasonality in the data. The trend can be inferred based on the number of lines crossing the confidence interval.
- The strong seasonality can be inferred based on the wavy nature of the Acf plot, and the seasonality period is 12 months. We can see a peak and dip every six months simultaneously.
- We can observe the same thing in the plot as well.

### Decomposition

```
stl_dec <- stl(candy_ts,s.window ="periodic")
plot(stl_dec, main = 'Decomposition plot')
```




**Decomposition Plot**

```
dec <- decompose(candy_ts)
dec$type
```

**Decomposition characteristic**

```
## [1] "additive"
```

- The decomposition is additive.
- Because, with as trend increases, we do not see any increase in the seasonality. The seasonality appears to be the same throughout.

```
dec$figure
```

**Seasonal monthly indices**

```
##  [1]   3.626833  -1.640936  -2.498000  -6.810844 -11.252817 -11.836234
##  [7] -12.216967  -4.767840   1.490589  14.736393  15.270768  15.899054
```

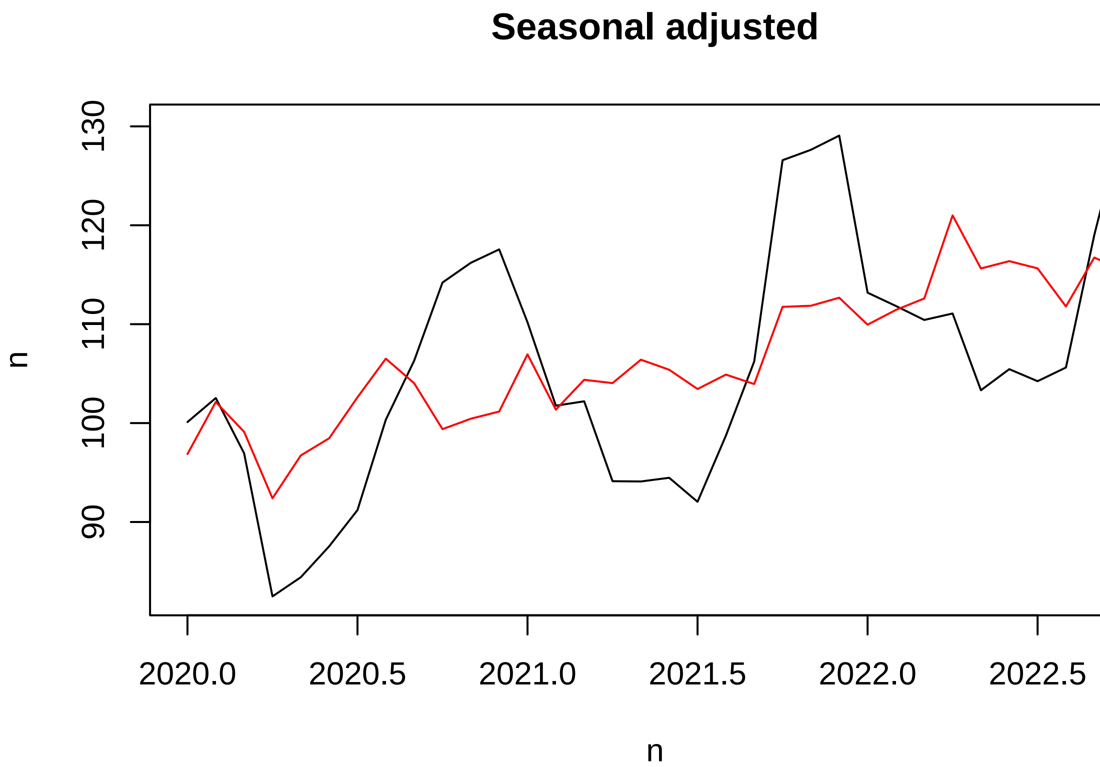
**Observations and Inferences**

- The time series is the highest for the month of December.
- The time series is the lowest for the month of July.

**Plausible reasons**

- The reason might be because of the winter holidays and Christmas season.
- Being a festival season, people purchase more candy during this season than the rest of the year.
- July, being the summer, may be the production going down and from July, the production restarts in numbers to cater for the demand of Thanksgiving and Christmas.

```r
plot(candy_ts, main='Seasonal adjusted', xlab='n', ylab='n')
lines(seasadj(stl_dec), col="Red")
```



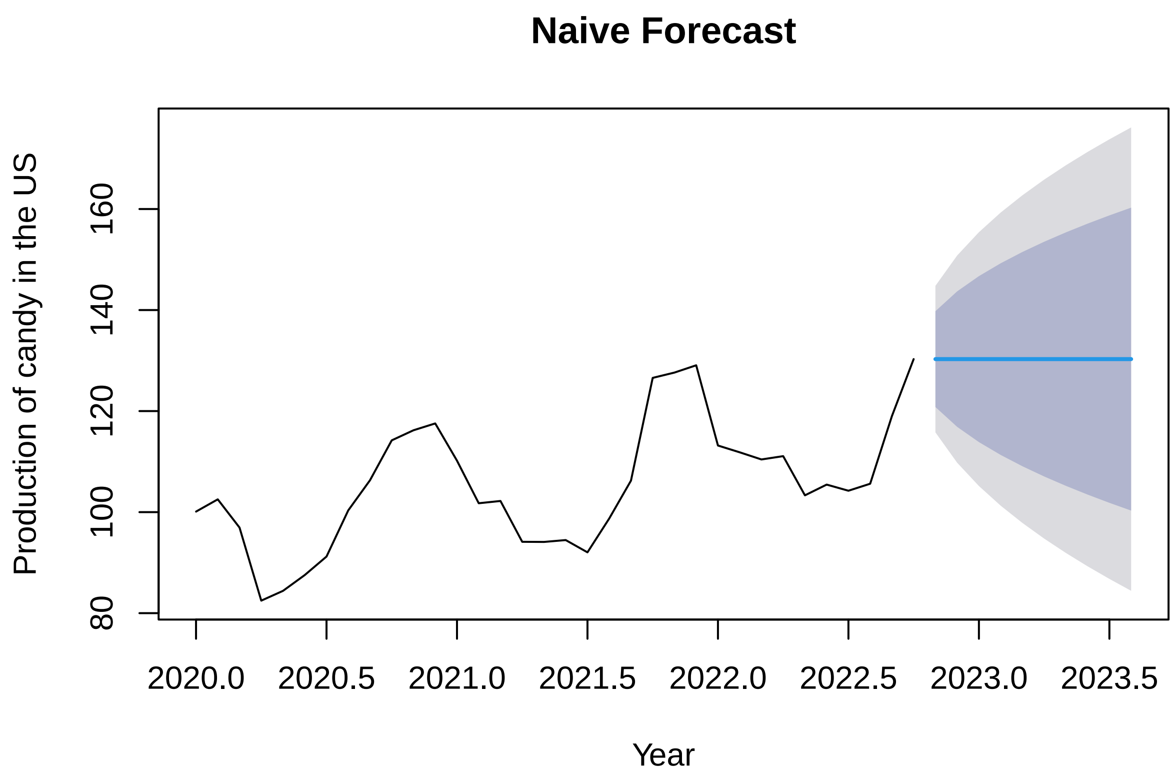

**Seasonality adjusted plot**

- The seasonality has significant fluctuations in the value of the time series.
- This is expected, as the data showed strong seasonality in the ACF plot.

## Testing various Forecasting methods for the given dataset

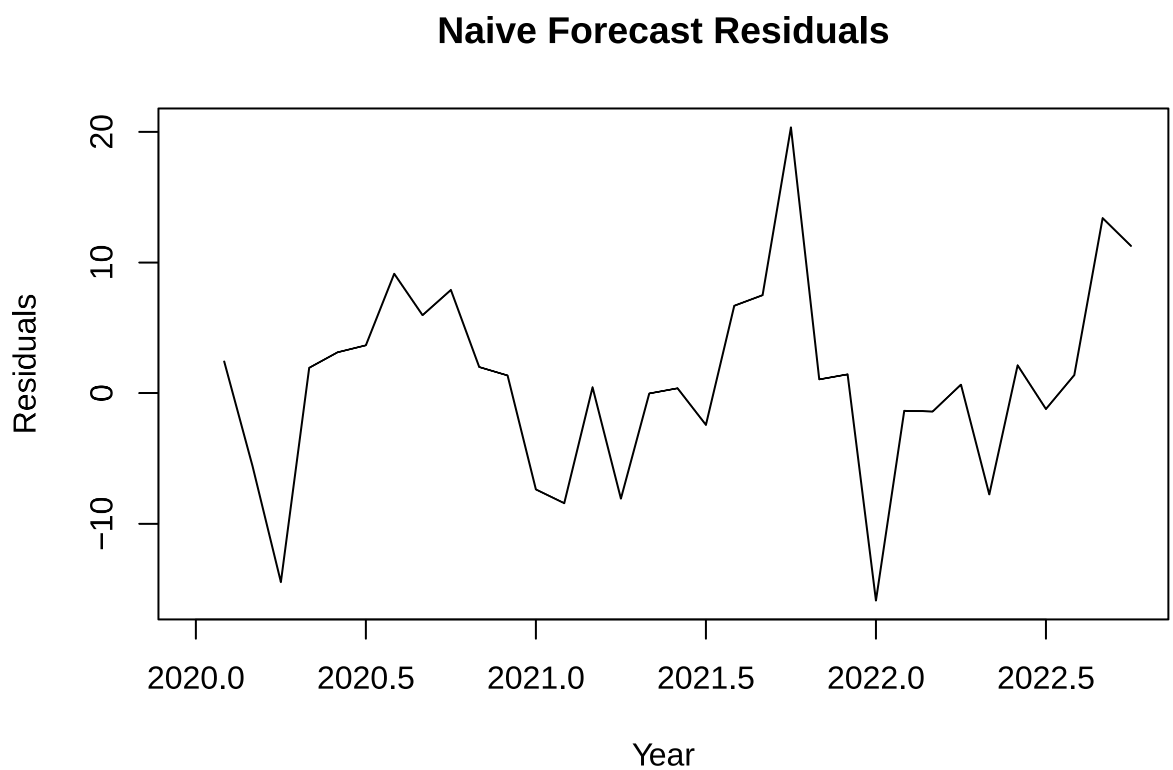
## Naïve Method

```r
naive_for = naive(candy_ts)
plot(naive_for, main = 'Naive Forecast', xlab='Year', ylab='Production of candy in the US')
```




**Q: Output**

```
plot(naive_for$residuals, main = 'Naive Forecast Residuals', xlab='Year', ylab='Residuals')
```




**Residual Analysis**

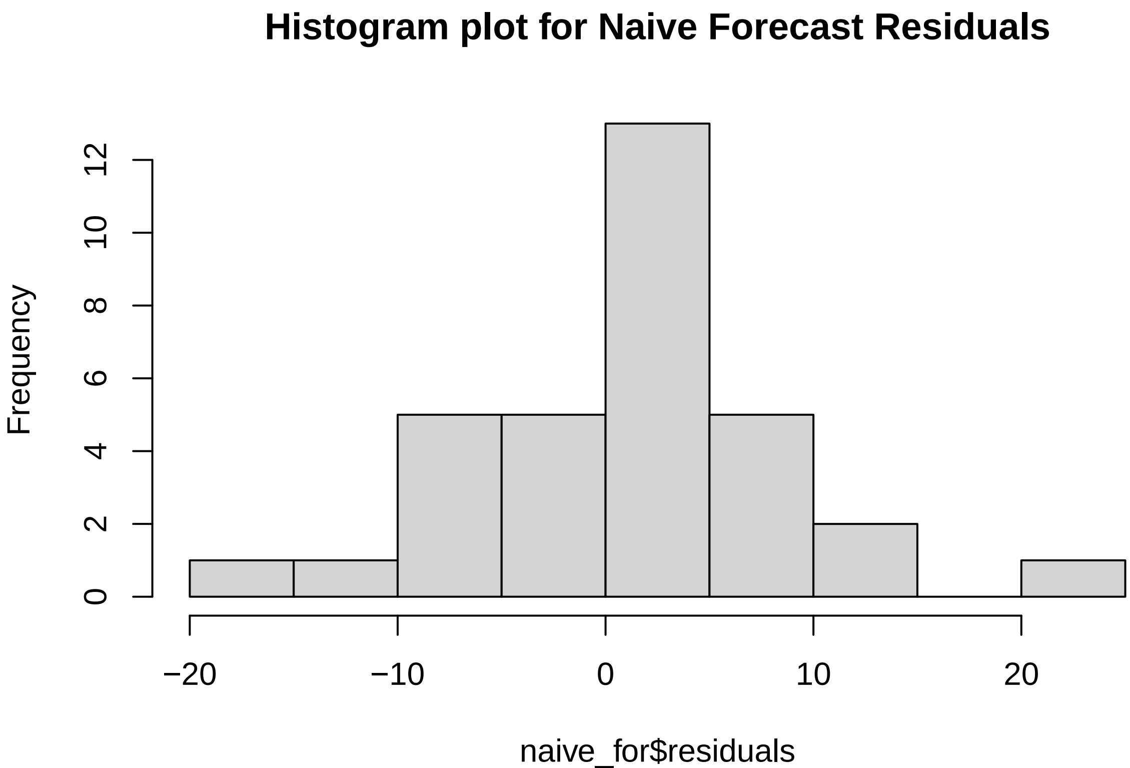
- The residuals have randomness until the year 2022.
- From 2022, the residuals have an increasing trend. This means we still need to include some factors to

be considered.

- The residuals have a mean of around zero. This can be checked in the histogram plot next.

Residuals Histogram

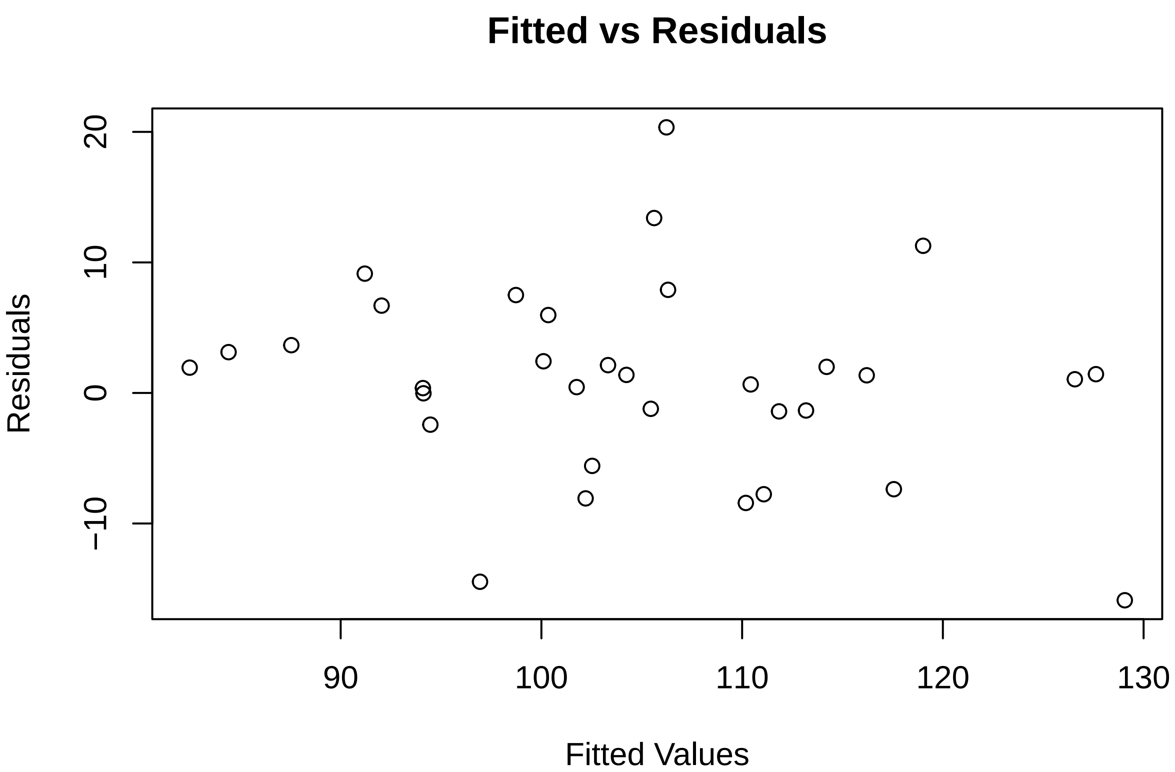
```
hist(naive_for$residuals, main = 'Histogram plot for Naive Forecast Residuals')
```




- The histogram appears to be normally distributed.
- But the values do not have a mean zero. The histogram appears to be skewed on one side.
- This means that the data is biased as the mean is not zero.

Fitted vs Residual Values

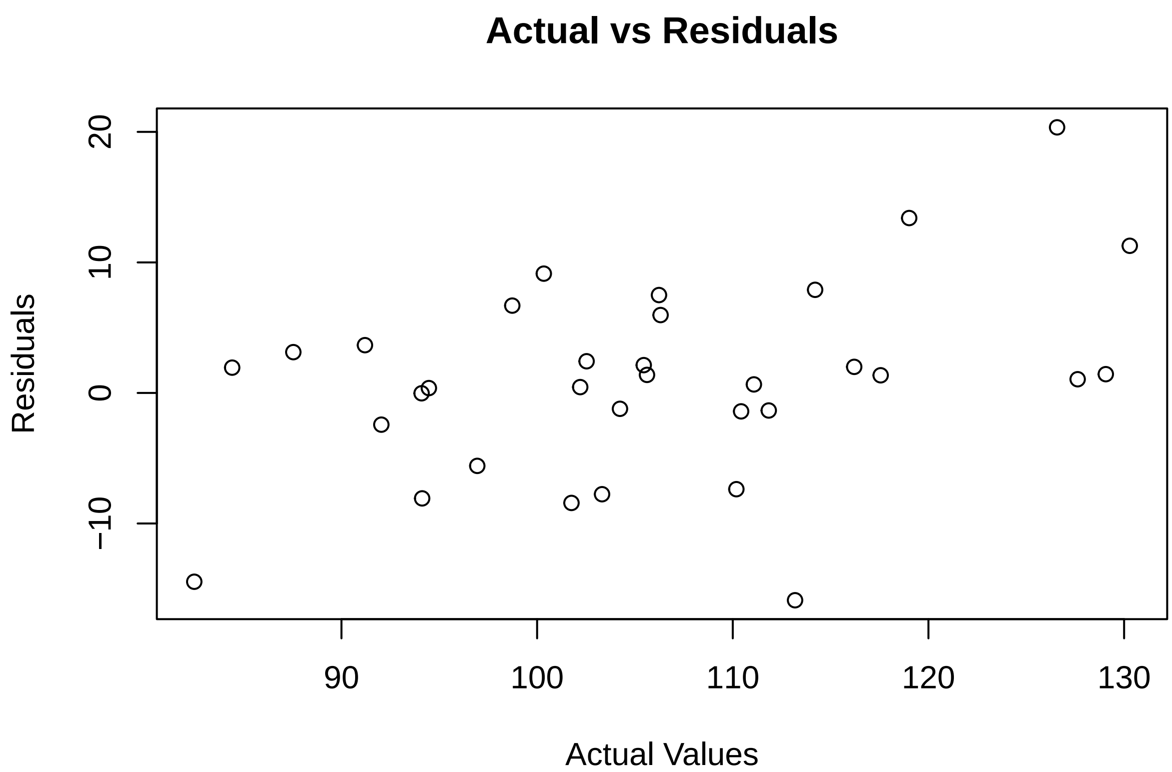
```
plot(as.numeric(fitted(naive_for)), residuals(naive_for), type='p', main = 'Fitted vs Residuals', ylab=
```




- The Fitted vs Residuals plot appears to be random and do not have any trend.
- The plot appears to have a mean around zero which is a good sign.
- However, there appear to be three outliers in the plot.

Actual vs Residual values

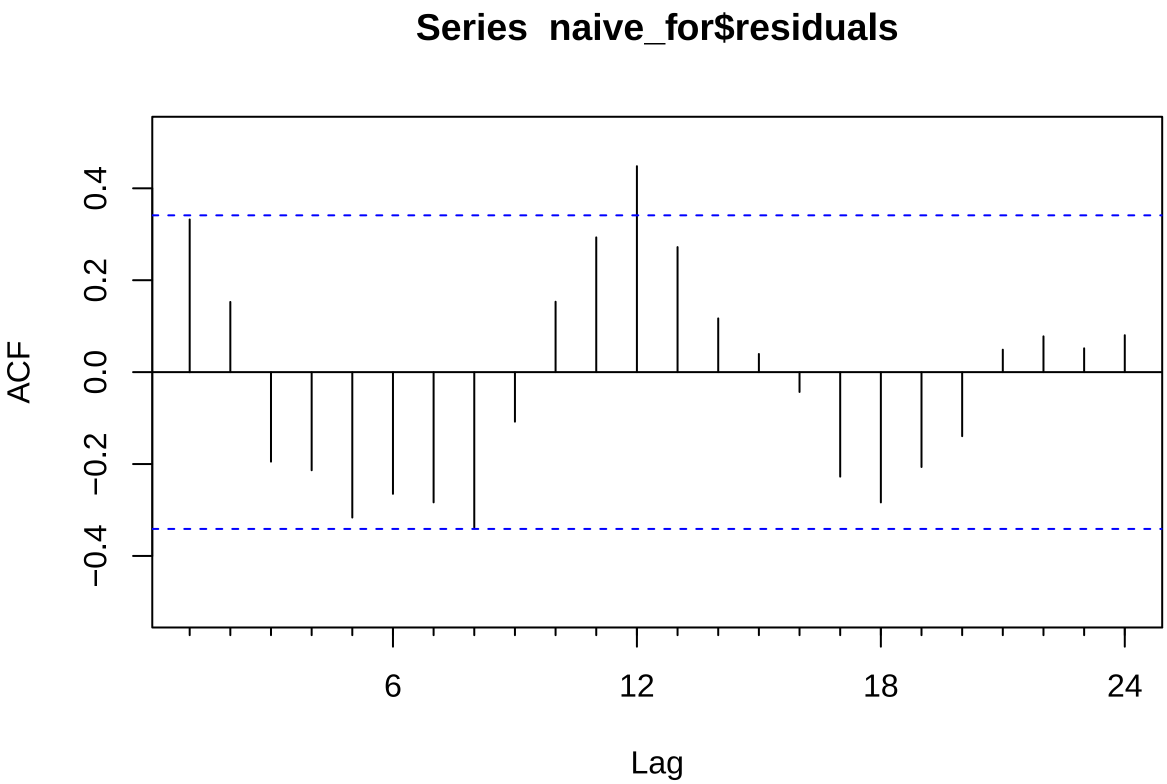
```
plot(as.numeric(candy_ts), residuals(naive_for), type='p', main = 'Actual vs Residuals', ylab='Residuals
```




- The Actual vs Residuals plot appears to have cone shape increasing residuals plot.
- This means the residuals are increasing with time which is a bad sign.
- Which means we are missing to consider some variable which is the reason for this abnormal residual plot.

ACF of residuals

```
Acf(naive_for$residuals)
```




- The Acf of residuals plot shows both trend and seasonality.
- Ideally the forecast is considered to be good if the Acf of residuals is white noice, meaning there is no trend or seasonality in the data and all the lines in the Acf plot are within the confidence interval.
- In this case, we missed some variable which is strongly affecting the residuals.

```
accuracy(naive_for)
```

**Accuracy**

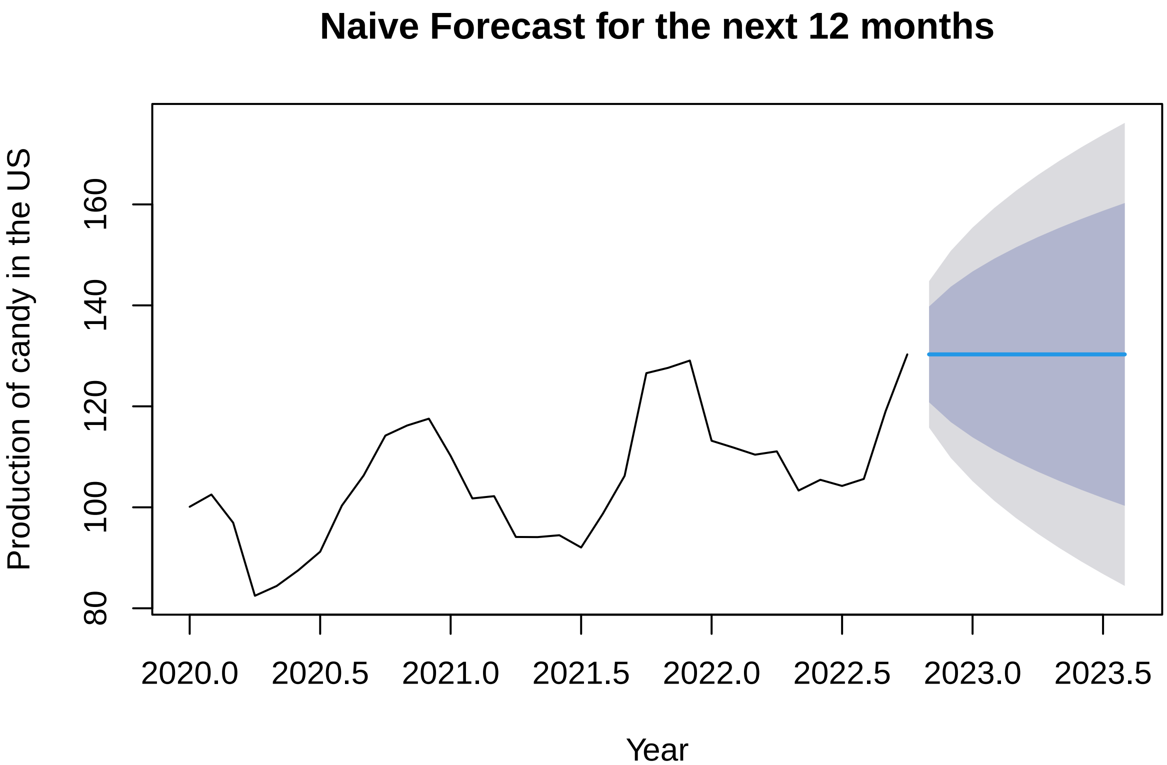
```
##                     ME     RMSE      MAE       MPE     MAPE      MASE      ACF1
## Training set 0.9147333 7.399605 5.399739 0.5619459 5.090241 0.6740498 0.3322229
```

```
forecast(naive_for)
```

**Forecast**

```
##          Point Forecast    Lo 80    Hi 80     Lo 95    Hi 95
## Nov 2022       130.2894 120.8064 139.7724 115.78644 144.7924
## Dec 2022       130.2894 116.8784 143.7004 109.77912 150.7997
## Jan 2023       130.2894 113.8644 146.7144 105.16954 155.4093
## Feb 2023       130.2894 111.3234 149.2554 101.28348 159.2953
## Mar 2023       130.2894 109.0848 151.4940  97.85980 162.7190
## Apr 2023       130.2894 107.0609 153.5179  94.76455 165.8142
## May 2023       130.2894 105.1998 155.3790  91.91818 168.6606
## Jun 2023       130.2894 103.4675 157.1113  89.26884 171.3100
## Jul 2023       130.2894 101.8405 158.7383  86.78052 173.7983
## Aug 2023       130.2894 100.3016 160.2772  84.42702 176.1518
```

```r
plot(forecast(naive_for), main = 'Naive Forecast for the next 12 months', xlab='Year', ylab='Production
```




**Naive Method Summary**

- The ME and RMSE values are very low, indicating that this method is suitable. But, it differs from what we can see as a trend and seasonality in the residuals.
- We can consider more forecasting techniques and check if the residuals are random.
- From 2020, there is an increasing trend in the residuals. We can try a naive method with a drift component, which may yield a better forecast.
- From the Acf of the residual plot, we can see that the residuals also have seasonality. So, we need to check other forecasting methods as well.

## Simple Moving Averages

```r
mavg3_forecast = ma(candy_ts,order=3)
mavg6_forecast = ma(candy_ts,order=6)
mavg9_forecast = ma(candy_ts,order=9)
plot(candy_ts, main = "Plot along with moving averages", xlab='Year', ylab='Production of candy in the U
lines(mavg3_forecast, col="Red")
lines(mavg6_forecast, col="Blue")
lines(mavg9_forecast, col="Green")
```

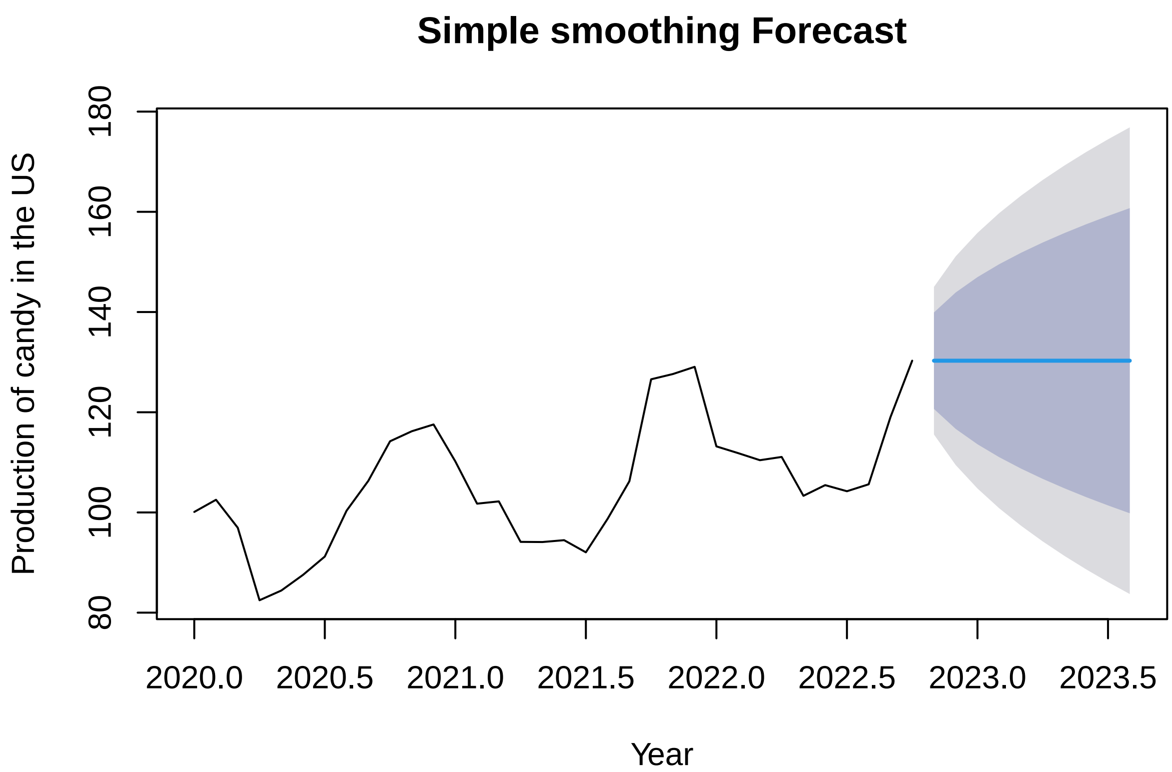

**Simple Moving average of order 3, 6, and 9**

**Observations**

- From the plots, it is observed that the higher the order we consider, the smoother the moving average curve in the plot.
- It can be seen that the Green line above is the smoothest compared to Blue or Red lines.
- The Red line (order 3) gives the most real data compared to the other two. The higher order averages smoother the plot and do not give the actual values.

## Simple Smoothing

```
ses_fit <- ses(candy_ts)
plot(ses_fit, main='Simple smoothing Forecast', xlab='Year', ylab='Production of candy in the US')
```




```r
attributes(ses_fit)
```

```
## $names
##  [1] "model"     "mean"      "level"     "x"         "upper"     "lower"
##  [7] "fitted"    "method"    "series"    "residuals"
##
## $class
## [1] "forecast"
```

Observations

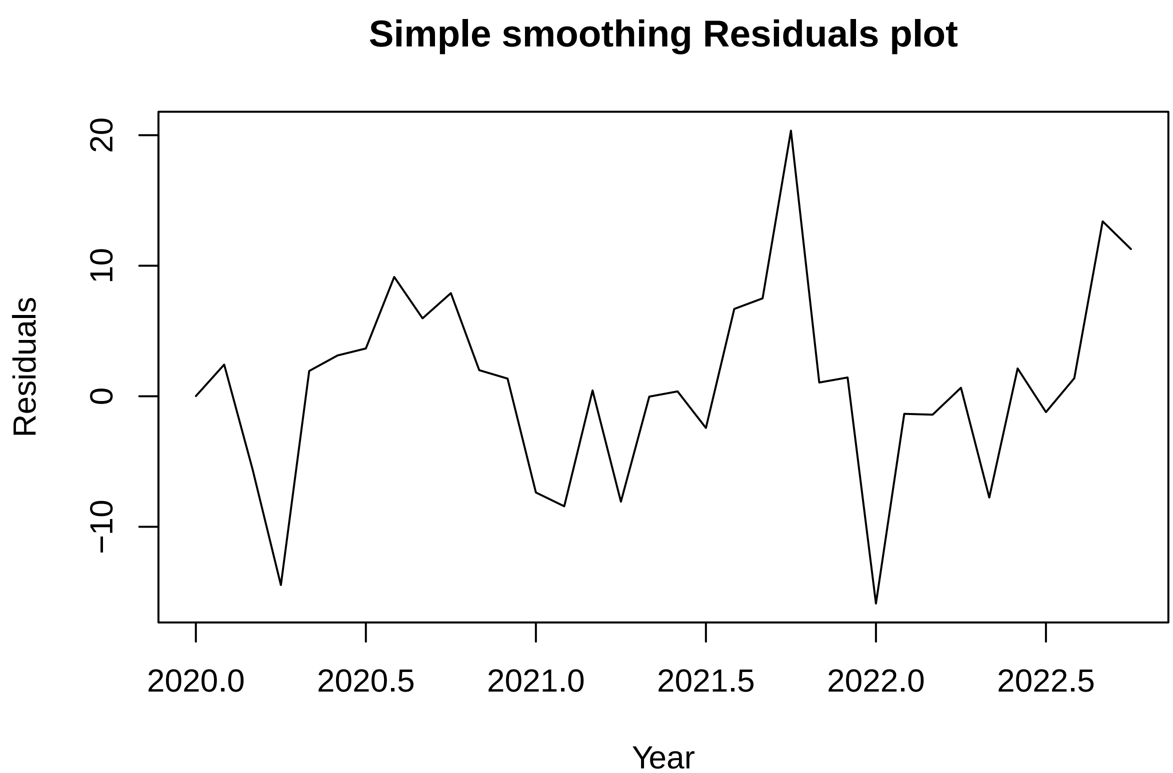
```r
summary(ses_fit)
```

```
##
## Forecast method: Simple exponential smoothing
##
## Model Information:
## Simple exponential smoothing
##
## Call:
##  ses(y = candy_ts)
##
##   Smoothing parameters:
##     alpha = 0.9999
##
##   Initial states:
##     l = 100.0911
##
##   sigma:  7.5146
##
##      AIC     AICc      BIC
```

```
## 260.9806 261.7806 265.5596
##
## Error measures:
##                     ME    RMSE      MAE       MPE     MAPE      MASE      ACF1
## Training set 0.8882407 7.29022 5.241508 0.5457879 4.941071 0.6542978 0.3312411
##
## Forecasts:
##          Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
## Nov 2022       130.2883 120.65794 139.9186 115.55995 145.0166
## Dec 2022       130.2883 116.66961 143.9069 109.46032 151.1162
## Jan 2023       130.2883 113.60916 146.9674 104.77977 155.7968
## Feb 2023       130.2883 111.02906 149.5475 100.83384 159.7427
## Mar 2023       130.2883 108.75592 151.8206  97.35738 163.2192
## Apr 2023       130.2883 106.70084 153.8757  94.21441 166.3621
## May 2023       130.2883 104.81100 155.7655  91.32414 169.2524
## Jun 2023       130.2883 103.05197 157.5246  88.63394 171.9426
## Jul 2023       130.2883 101.39985 159.1767  86.10724 174.4693
## Aug 2023       130.2883  99.83723 160.7393  83.71743 176.8591
```

- Alpha = 0.9999
- Alpha specifies the coefficient for the level smoothing.
- Values near 1.0 mean that the latest value has more weight.
- Initial state: l = 100.0911
- Sigma: 7.5146. Sigma defines the variance in the forecast predicted.

```r
plot(ses_fit$residuals, main='Simple smoothing Residuals plot', xlab='Year', ylab='Residuals')
```



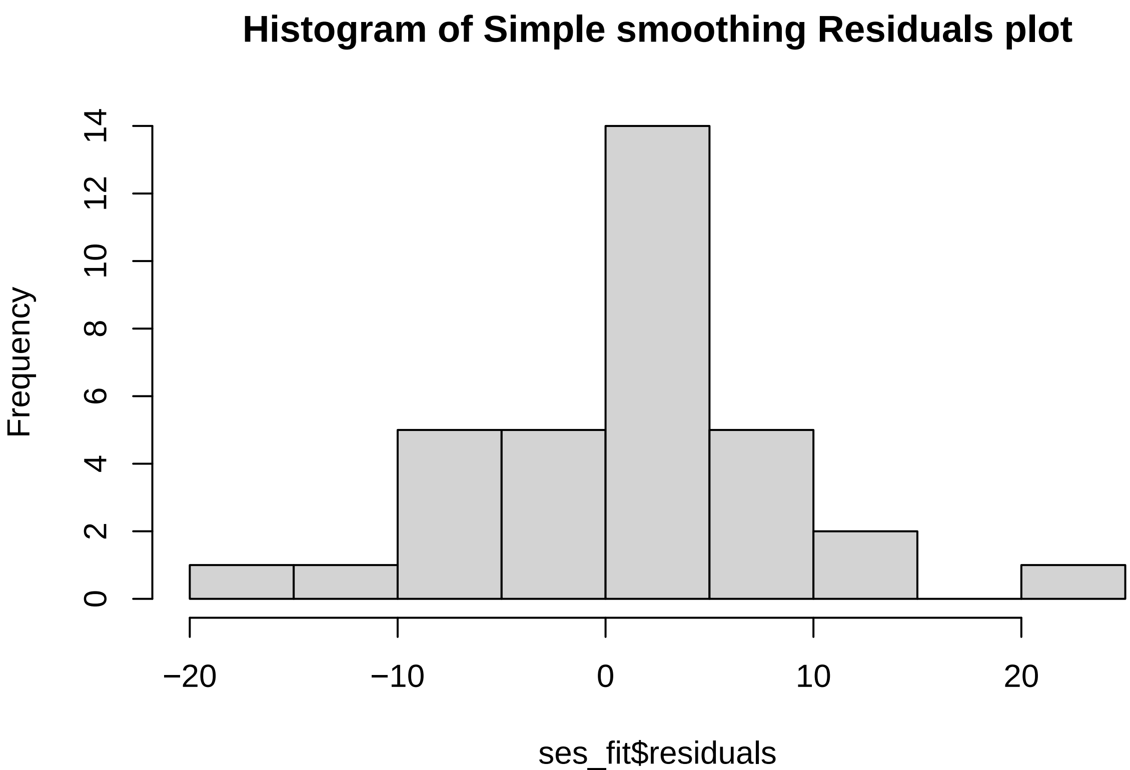

**Residual Analysis**

- The residuals seem to be have randomness til the year 2022.
- From 2022, the residuals seem to have an increasing trend. Which means we have missed some factor

to be considered.
- The residuals seem to have a mean around zero. This can be checked in the histogram plot next.

Histogram plot of residuals

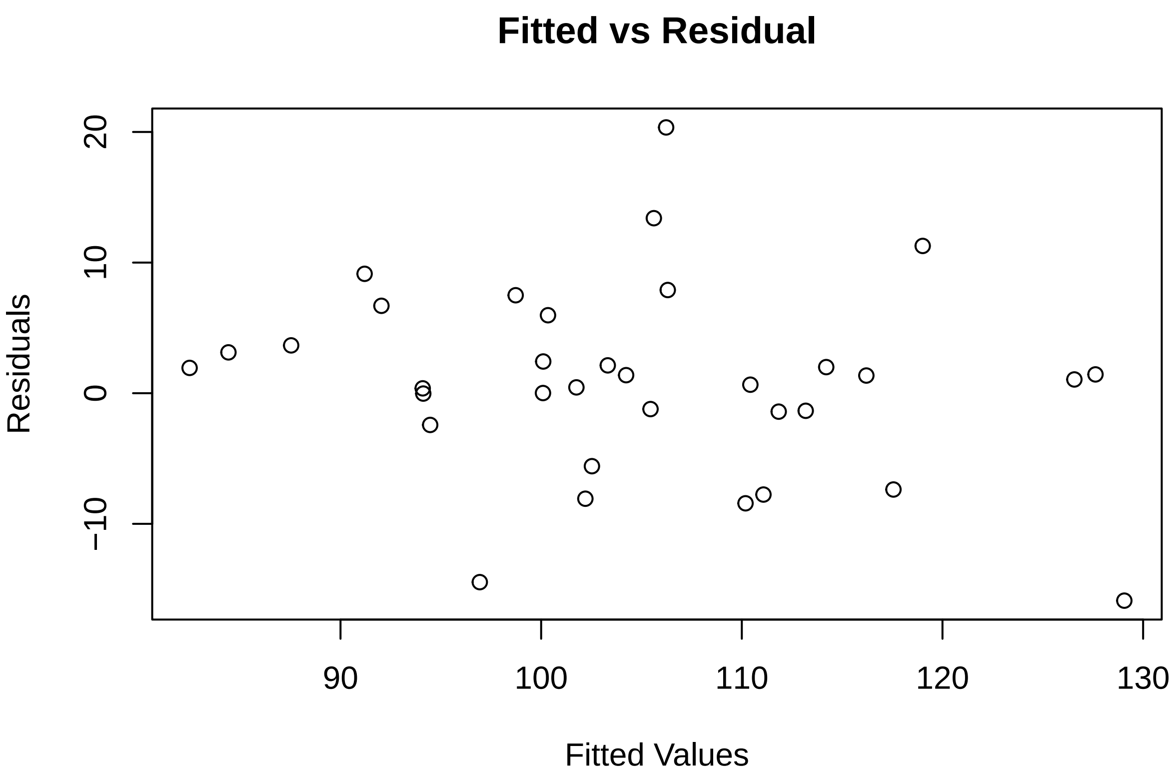
```
hist(ses_fit$residuals, main='Histogram of Simple smoothing Residuals plot')
```




- The histogram appears to be normally distributed.
- But the values do not have a mean zero. The histogram appears to be skewed on one side.
- This means that the data is biased as the mean is not zero.

Fitted values vs. residuals

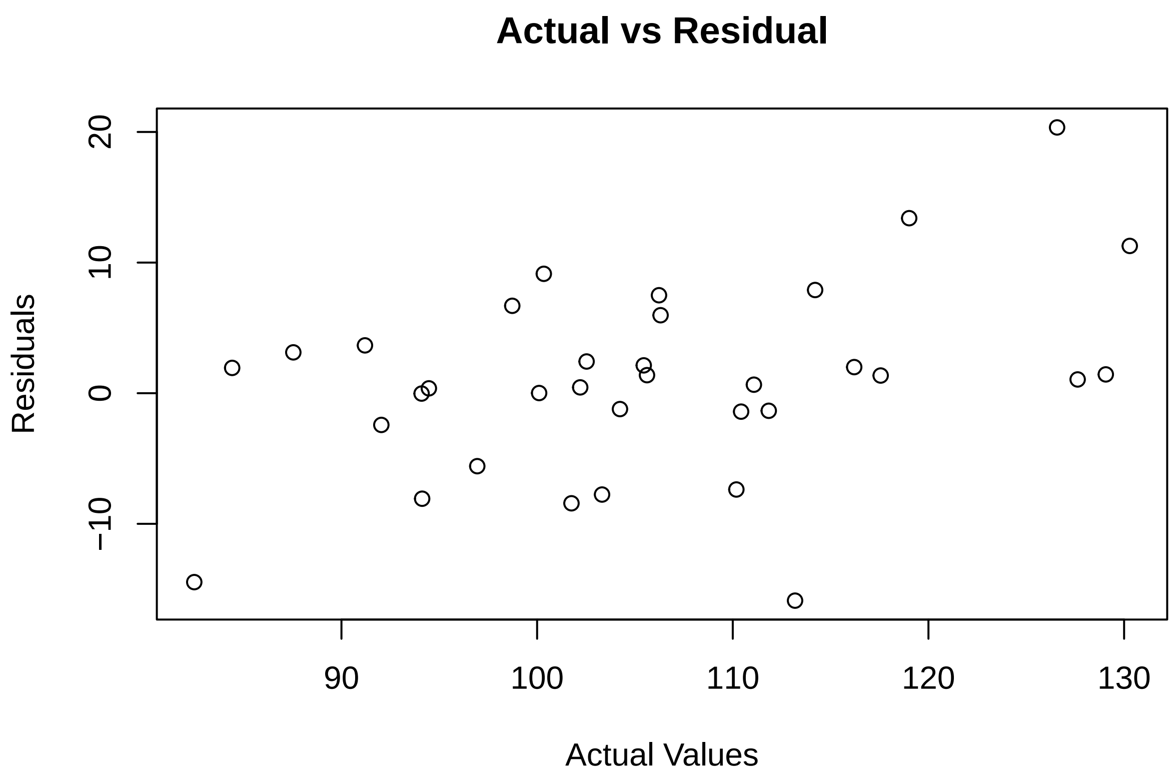
```
plot(as.numeric(fitted(ses_fit)), residuals(ses_fit), type='p', main = 'Fitted vs Residual', ylab='Resi
```




- The Fitted vs Residuals plot appears to be random and do not have any trend.
- The plot appears to have a mean around zero which is a good sign.
- The plot however seems to have 3 outliers.

Actual values vs. residuals

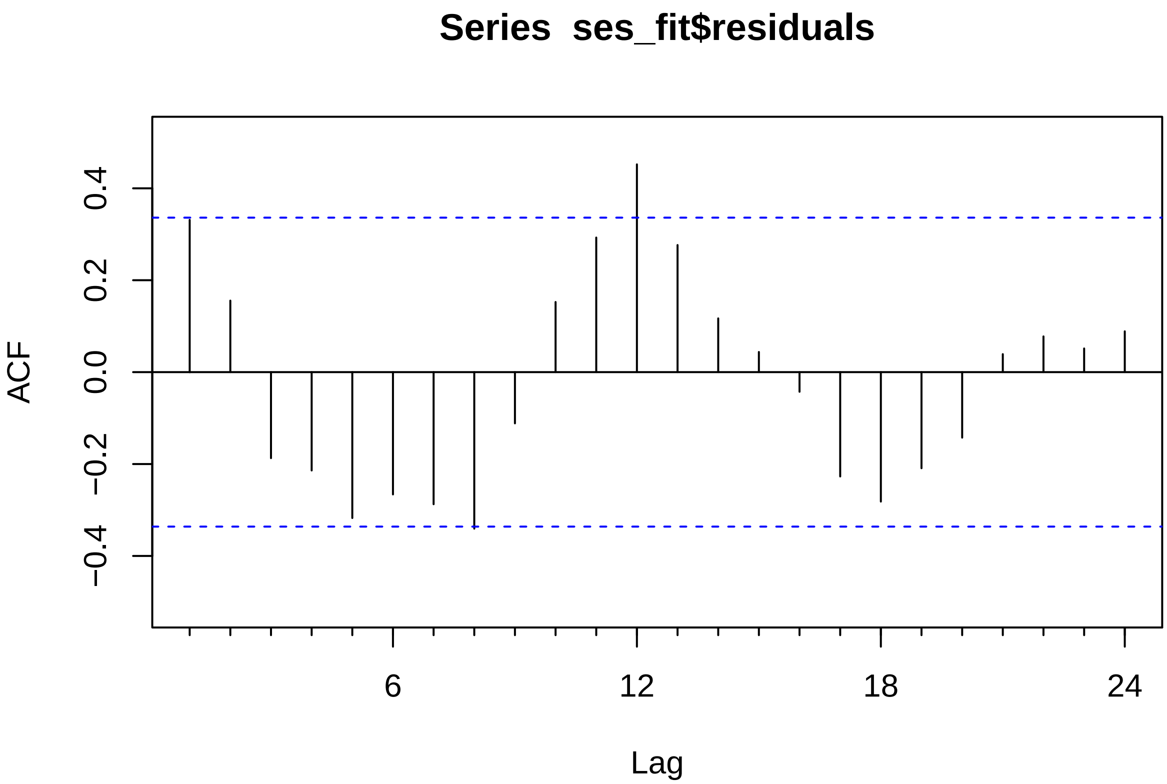
```
plot(as.numeric(candy_ts), residuals(ses_fit), type='p', main = 'Actual vs Residual', ylab='Residuals',
```




- The Actual vs Residuals plot appears to have cone shape increasing residuals plot.
- This means the residuals are increasing with time which is a bad sign.
- Which means we are missing to consider some variable which is the reason for this abnormal residual plot.

ACF plot of the residuals

```
Acf(ses_fit$residuals)
```




- The Acf of residuals plot shows both trend and seasonality.
- Ideally the forecast is considered to be good if the Acf of residuals is white noice, meaning there is no trend or seasonality in the data and all the lines in the Acf plot are within the confidence interval.
- In this case, we missed some variable which is strongly affecting the residuals.

```
accuracy(ses_fit)
```

**Accuracy**

```
##                     ME    RMSE      MAE       MPE     MAPE      MASE      ACF1
## Training set 0.8882407 7.29022 5.241508 0.5457879 4.941071 0.6542978 0.3312411
```

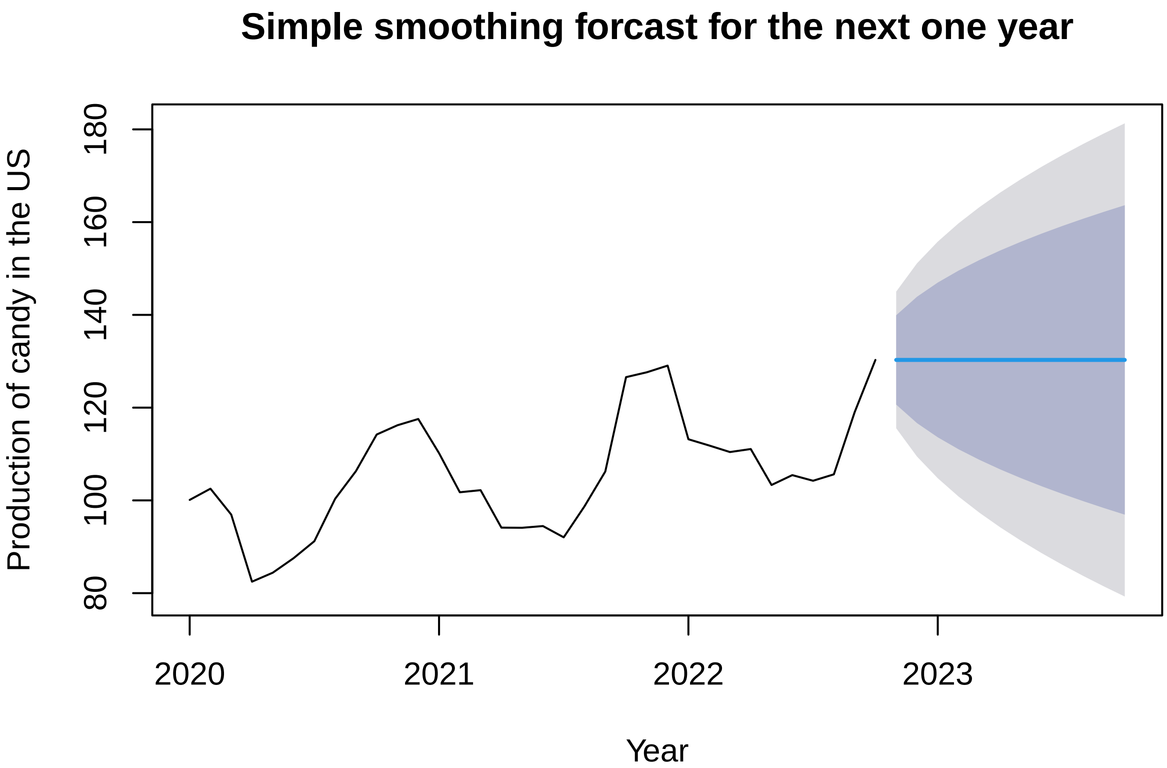
```
ses(candy_ts, h=12)
```

**Q: Forecast**

```
##          Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
## Nov 2022       130.2883 120.65794 139.9186 115.55995 145.0166
## Dec 2022       130.2883 116.66961 143.9069 109.46032 151.1162
## Jan 2023       130.2883 113.60916 146.9674 104.77977 155.7968
## Feb 2023       130.2883 111.02906 149.5475 100.83384 159.7427
## Mar 2023       130.2883 108.75592 151.8206  97.35738 163.2192
## Apr 2023       130.2883 106.70084 153.8757  94.21441 166.3621
## May 2023       130.2883 104.81100 155.7655  91.32414 169.2524
## Jun 2023       130.2883 103.05197 157.5246  88.63394 171.9426
## Jul 2023       130.2883 101.39985 159.1767  86.10724 174.4693
## Aug 2023       130.2883  99.83723 160.7393  83.71743 176.8591
## Sep 2023       130.2883  98.35098 162.2256  81.44440 179.1321
```

```
## Oct 2023       130.2883  96.93089 163.6457  79.27255 181.3040
```

```
plot(ses(candy_ts, h=12), main = 'Simple smoothing forcast for the next one year', xlab='Year', ylab='P
```

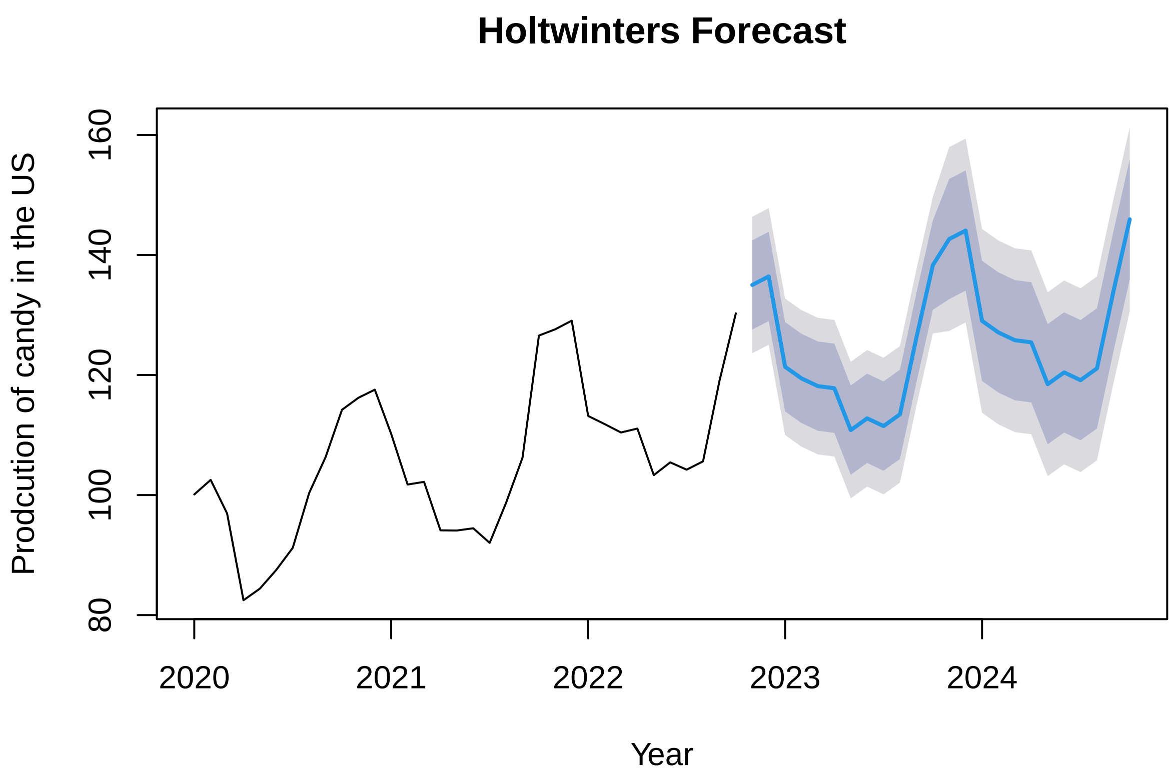



**Simple Smoothing Summary**

- The ME and RMSE values are very low, indicating that this method is suitable. But, it differs from what we can see as a trend and seasonality in the residuals.
- We can consider more forecasting techniques and check if the residuals are random.
- From 2020, there is an increasing trend in the residuals. This means we still need some variable that needs to be considered.
- From the Acf of the residual plot, we can see that the residuals also have seasonality. So, we need to check other forecasting methods as well. Next, we check the HoltWinters forecasting method.

## Holt-Winters

```
HW_forecast <- hw(candy_ts, seasonal = "additive")
plot(forecast(HW_forecast), main='Holtwinters Forecast', xlab='Year', ylab='Prodcution of candy in the U
```




```r
attributes(HW_forecast)
```

```
## $names
##  [1] "model"     "mean"      "level"     "x"         "upper"     "lower"
##  [7] "fitted"    "method"    "series"    "residuals"
##
## $class
## [1] "forecast"
```

```r
hw_add <- forecast(HW_forecast)
```

- Here, additive Holtwinters method is considered.
- This is because the seasonality isn't increasing with trend. This is an additive time series.

Observations

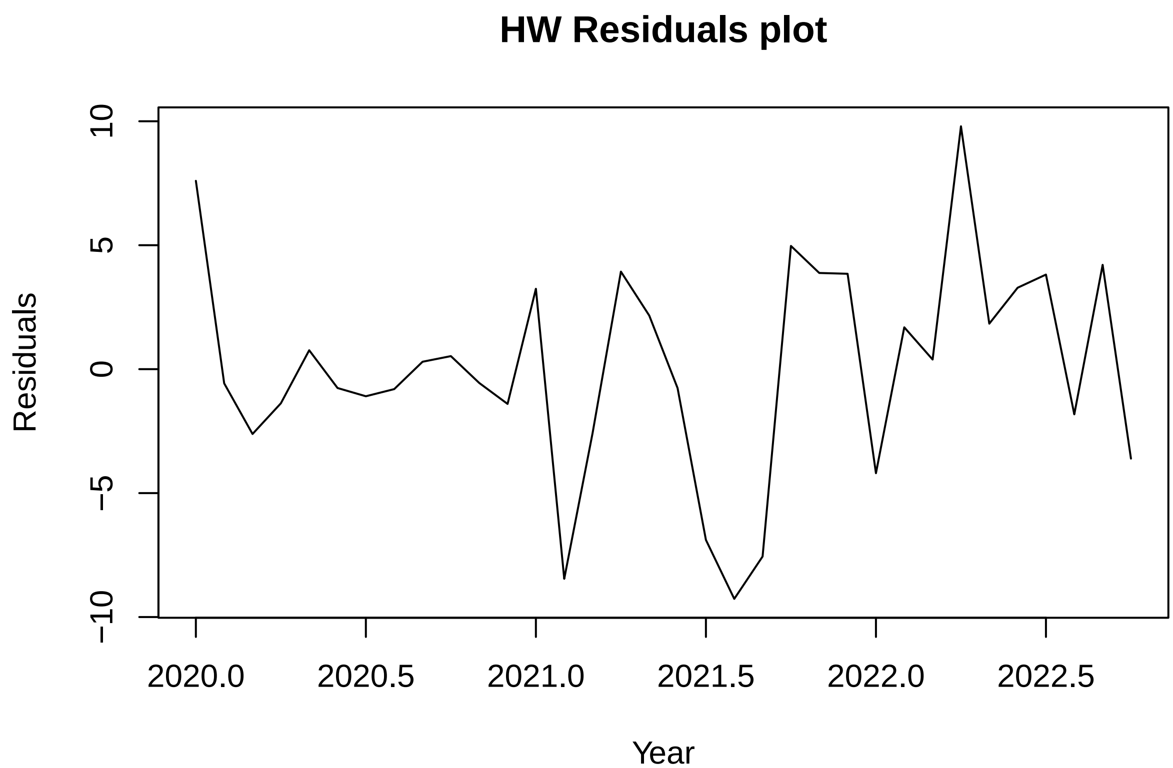
```r
hw_add$model
```

```
## Holt-Winters' additive method
##
## Call:
##  hw(y = candy_ts, seasonal = "additive")
##
##   Smoothing parameters:
##     alpha = 0.0068
##     beta  = 1e-04
##     gamma = 0.8931
##
##   Initial states:
##     l = 95.8204
##     b = 0.6373
##     s = 15.479 13.9148 11.4759 4.4435 0.2118 -8.01
```

```
##            -11.3705 -15.3768 -14.5466 1.7751 5.9553 -3.9515
##
##   sigma:  5.8034
##
##      AIC     AICc      BIC
## 251.8472 290.0972 277.7954
```

- Alpha = 0.0068. Alpha specifies the coefficient for the level smoothing in Holtwinters.
- Beta = 0.00001. Beta specifies the coefficient for the trend smoothing in Holtwinters.
- Gamma = 0.8931. Gamma specifies the coefficient for the seasonal smoothing in Holtwinters.
- Values 1.0 means that the latest value has highest weight.
- Initial states: l = 95.8204 b = 0.6373 s = 15.479 13.9148 11.4759 4.4435 0.2118 -8.01 -11.3705 -15.3768 -14.5466 1.7751 5.9553 -3.9515
- Sigma = 5.8034. Sigma defines the variance of the forecast values.

```r
plot(hw_add$residuals, main='HW Residuals plot', xlab='Year', ylab='Residuals')
```

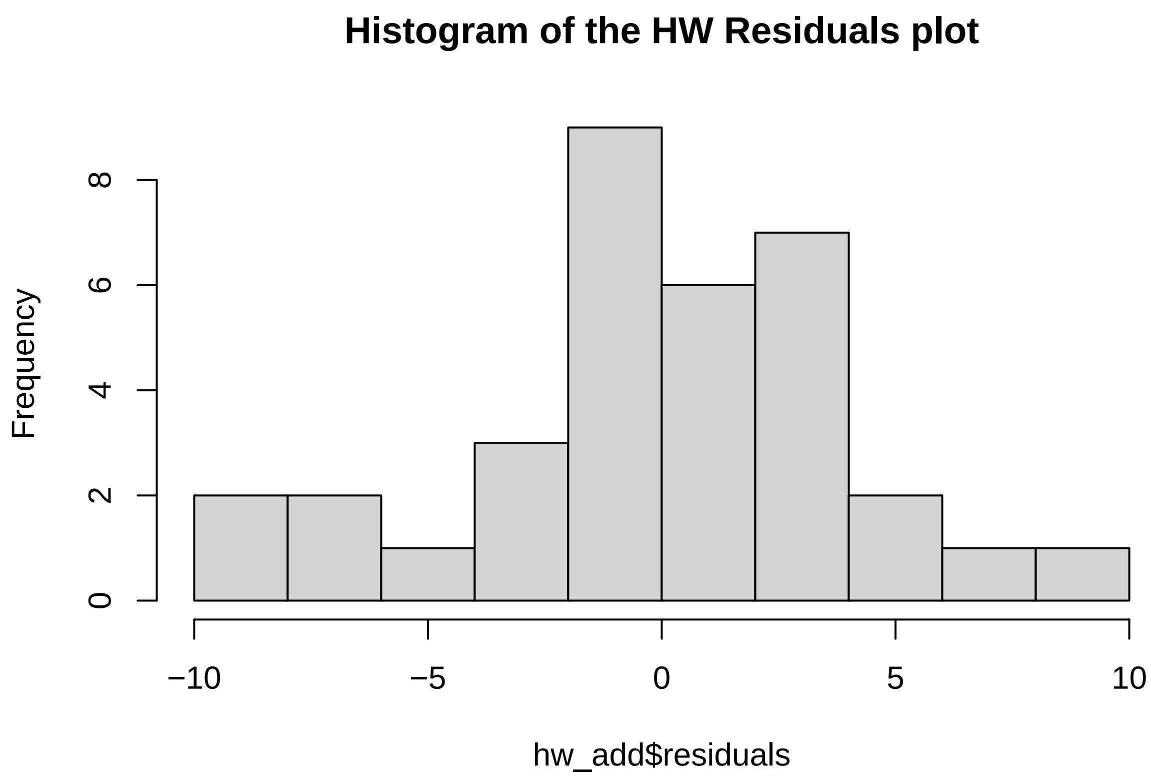



**Residual Analysis**

- The residuals appear to be random and also the mean looks to be near zero. We can check this with histogram.
- We can observe a couple of up and downs throughout. But even they did not show and growing residual pattern.

Histogram plot of residuals

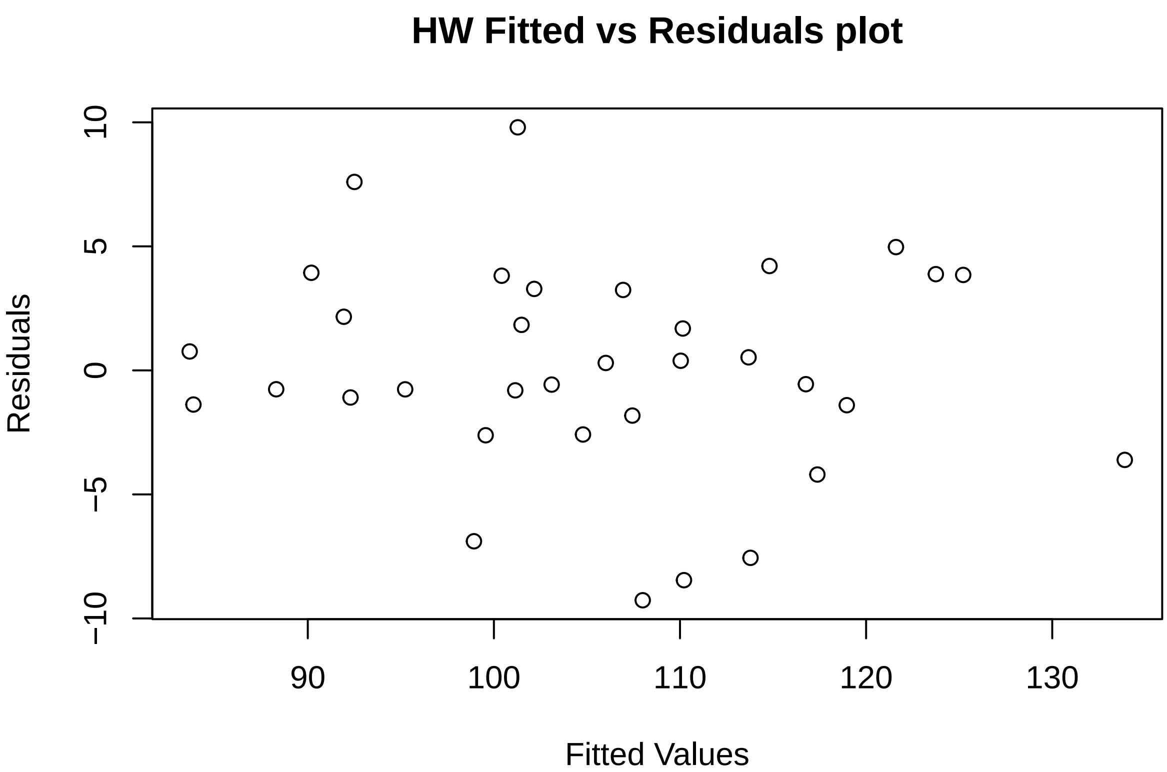
```r
hist(hw_add$residuals, main='Histogram of the HW Residuals plot')
```




- The histogram appears to be normally distributed.
- And the mean does not appear to be at zero. This means the data is biased and we might have missed some variable.

Fitted values vs. residuals

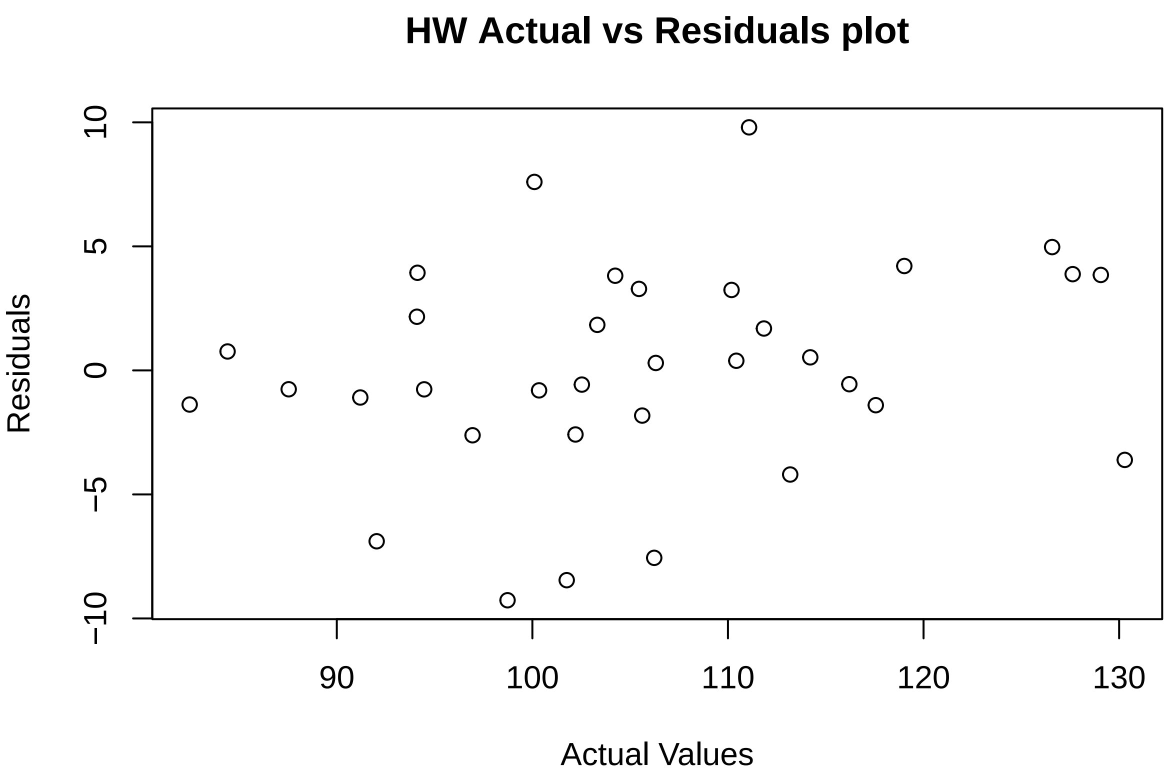
```
plot(as.numeric(fitted(hw_add)), residuals(hw_add), type='p', main='HW Fitted vs Residuals plot',  ylab=
```




- The Fitted vs Residuals plot appears to be random and do not have any trend.
- The plot appears to have a mean around zero which is a good sign.
- The plot however seems to have 2 outliers.

Actual values vs. residuals

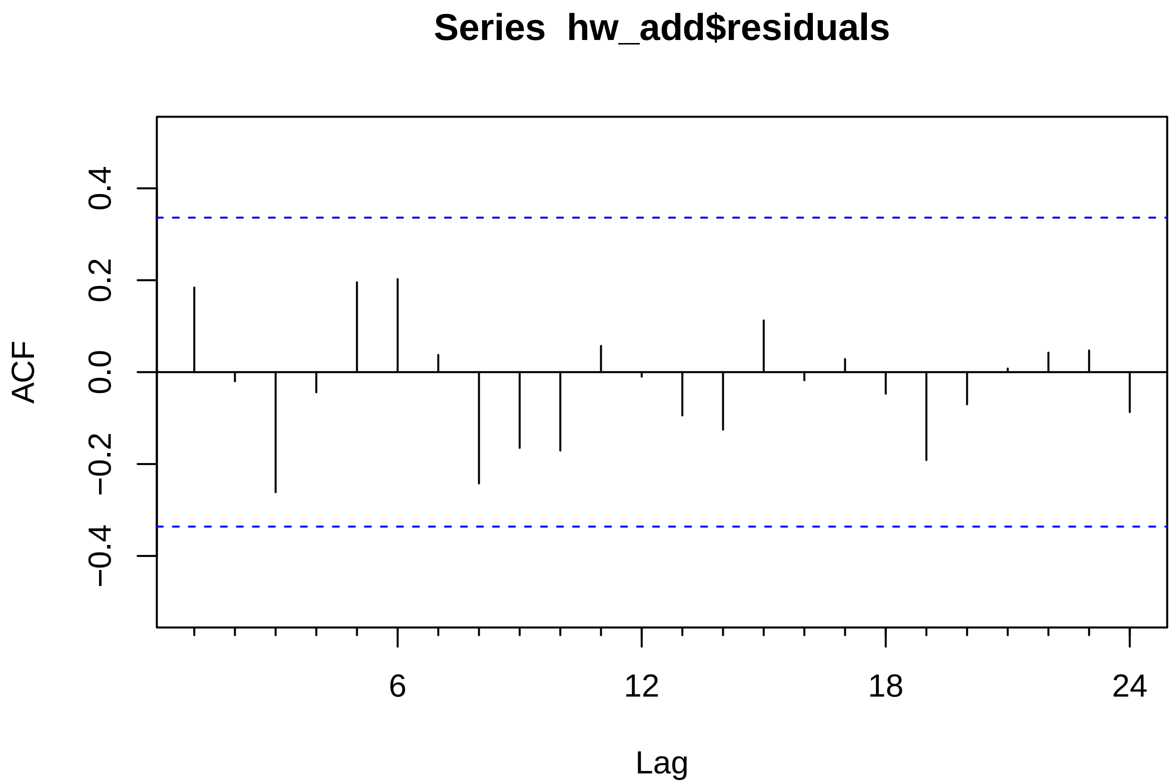
```
plot(as.numeric(candy_ts), residuals(hw_add), type='p', main='HW Actual vs Residuals plot',  ylab='Resi
```




- The Actual vs Residuals plot appears to be random and do not have any trend.
- The plot appears to have a mean around zero which is a good sign.
- The plot however seems to have 4 outliers.

ACF plot of the residuals

```
Acf(hw_add$residuals)
```




- In the Acf plot, none of the values crossed the confidence levels. It appears to be white noice.
- This signifies that the forecast is a good forecast.
- This proves to be the best forecast comparing all the previous ones tested.

```r
accuracy(hw_add)
```

**Accuracy**

```
##                      ME     RMSE      MAE         MPE     MAPE      MASE
## Training set 0.05597211 4.222618 3.252036 -0.05703846 3.078744 0.4059518
##                   ACF1
## Training set 0.1842016
```

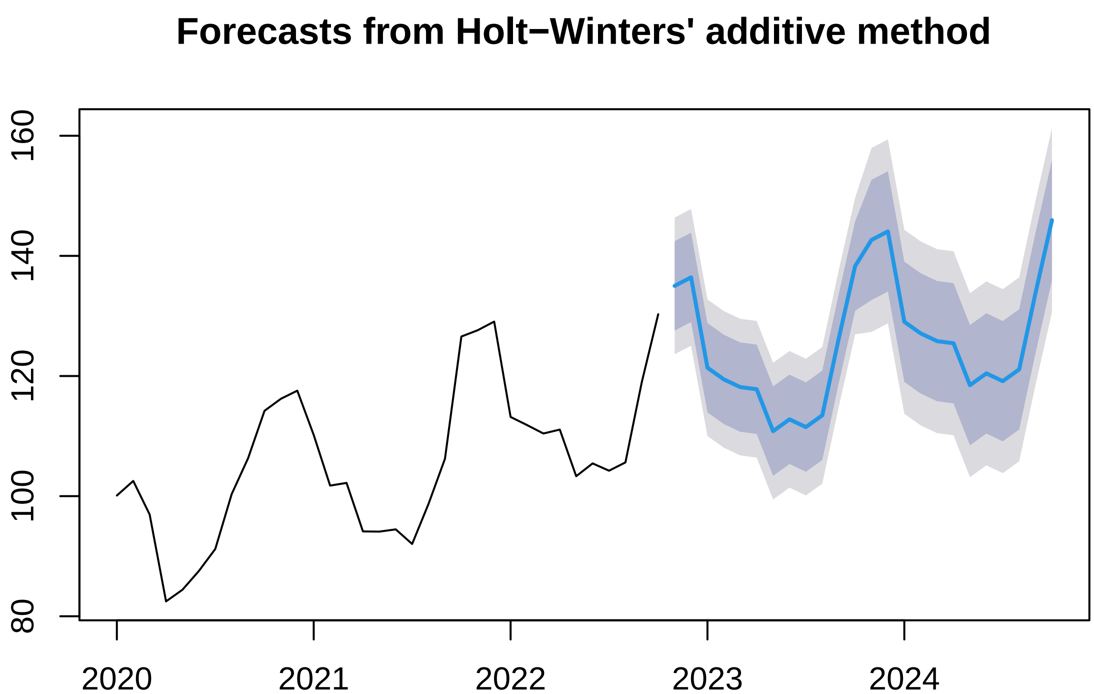
```r
forecast(HW_forecast)
```

**Q: Forecast**

```
##          Point Forecast    Lo 80    Hi 80     Lo 95    Hi 95
## Nov 2022       135.0090 127.5716 142.4464 123.63445 146.3835
## Dec 2022       136.4263 128.9887 143.8639 125.05150 147.8011
## Jan 2023       121.3807 113.9429 128.8184 110.00558 132.7557
## Feb 2023       119.4375 111.9996 126.8755 108.06216 130.8129
## Mar 2023       118.1547 110.7165 125.5928 106.77900 129.5303
## Apr 2023       117.8001 110.3617 125.2384 106.42411 129.1761
## May 2023       110.8252 103.3867 118.2638  99.44892 122.2015
## Jun 2023       112.7892 105.3504 120.2280 101.41259 124.1658
## Jul 2023       111.4952 104.0562 118.9342 100.11828 122.8722
## Aug 2023       113.4537 106.0145 120.8929 102.07642 124.8310
```

```
## Sep 2023       126.2208 118.7814 133.6603 114.84321 137.5985
## Oct 2023       138.3009 130.8613 145.7406 126.92293 149.6789
## Nov 2023       142.6593 132.6460 152.6725 127.34530 157.9732
## Dec 2023       144.0766 134.0631 154.0900 128.76235 159.3908
## Jan 2024       129.0309 119.0173 139.0446 113.71642 144.3455
## Feb 2024       127.0878 117.0740 137.1016 111.77300 142.4026
## Mar 2024       125.8050 115.7909 135.8190 110.48983 141.1201
## Apr 2024       125.4504 115.4362 135.4646 110.13495 140.7658
## May 2024       118.4755 108.4611 128.4899 103.15975 133.7913
## Jun 2024       120.4395 110.4249 130.4541 105.12342 135.7556
## Jul 2024       119.1455 109.1307 129.1604 103.82912 134.4619
## Aug 2024       121.1040 111.0889 131.1191 105.78726 136.4208
## Sep 2024       133.8711 123.8558 143.8865 118.55405 149.1882
## Oct 2024       145.9512 135.9357 155.9668 130.63377 161.2687
```

```
plot(forecast(HW_forecast))
```




**Holtwinters Summary**

- The ME, RMSE values are quite low compared to any of our previous forecasts.
- HolWinters is a better forecast compared to naive and simple smoothing.
- Holtwinters appears to be the best forecast considering all the previous forecast methods.
- However, this forecast can still be improved as we can try forecasting using ARIMA models.

## ARIMA

### Is Time Series data Stationary?

- The Time Series data is not stationary.
- A time series is considered stationary if there is no trend and seasonality in the time series.
- The time series that we considered has both trend and seasonality. So, it is not stationary.

```
nsdiffs(candy_ts)
```

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

```r
ndiffs(candy_ts)
```

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

- A seasonality component is needed in this case.
- First, we do the seasonal differencing.
- This is because once the seasonal differencing is done, in most cases, it will take care of trend differencing itself.
- We see that the trend differencing is one, but let us check for the trend differencing in the following case after seasonal differencing.

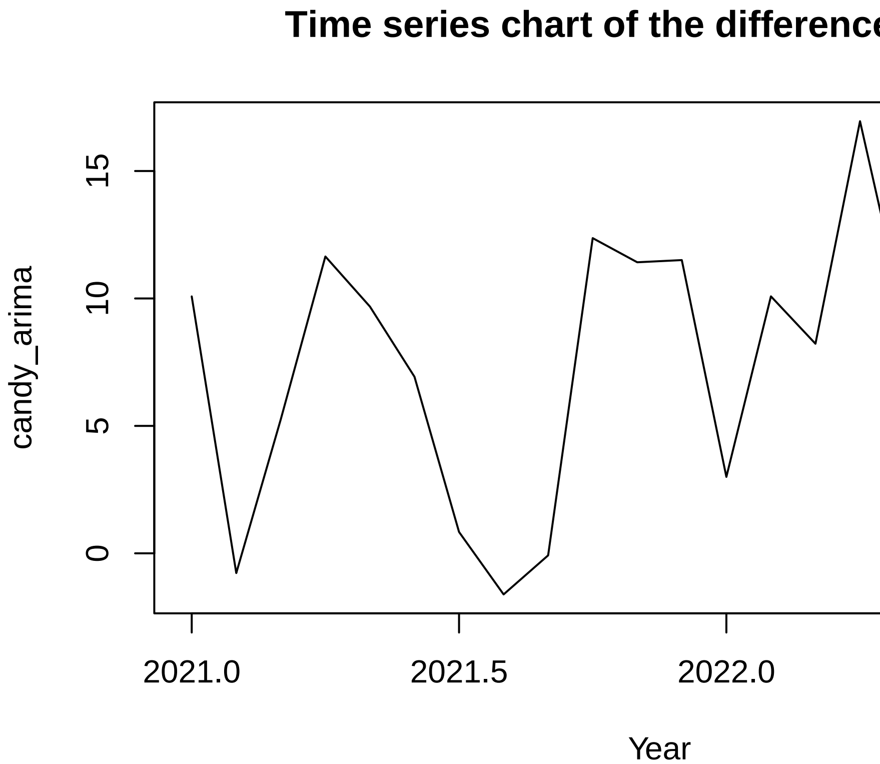
```r
ndiffs((diff(candy_ts,12)))
```

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

- As discussed earlier, the ndiffs value is zero now after performing the seasonal differencing.
- The seasonal differencing took care of trend differencing itself.

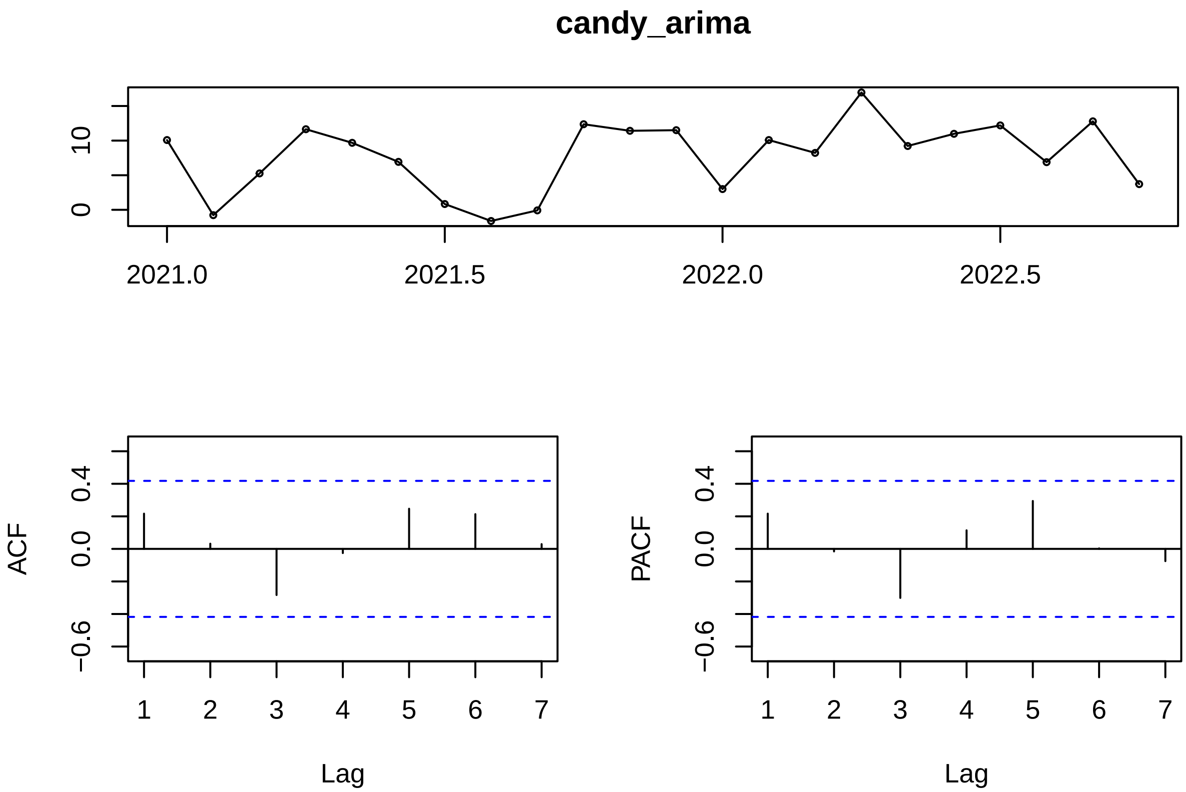
```r
candy_arima <- diff(candy_ts,12)
```

```r
plot(candy_arima, main='Time series chart of the differenced series', xlab='Year')
```



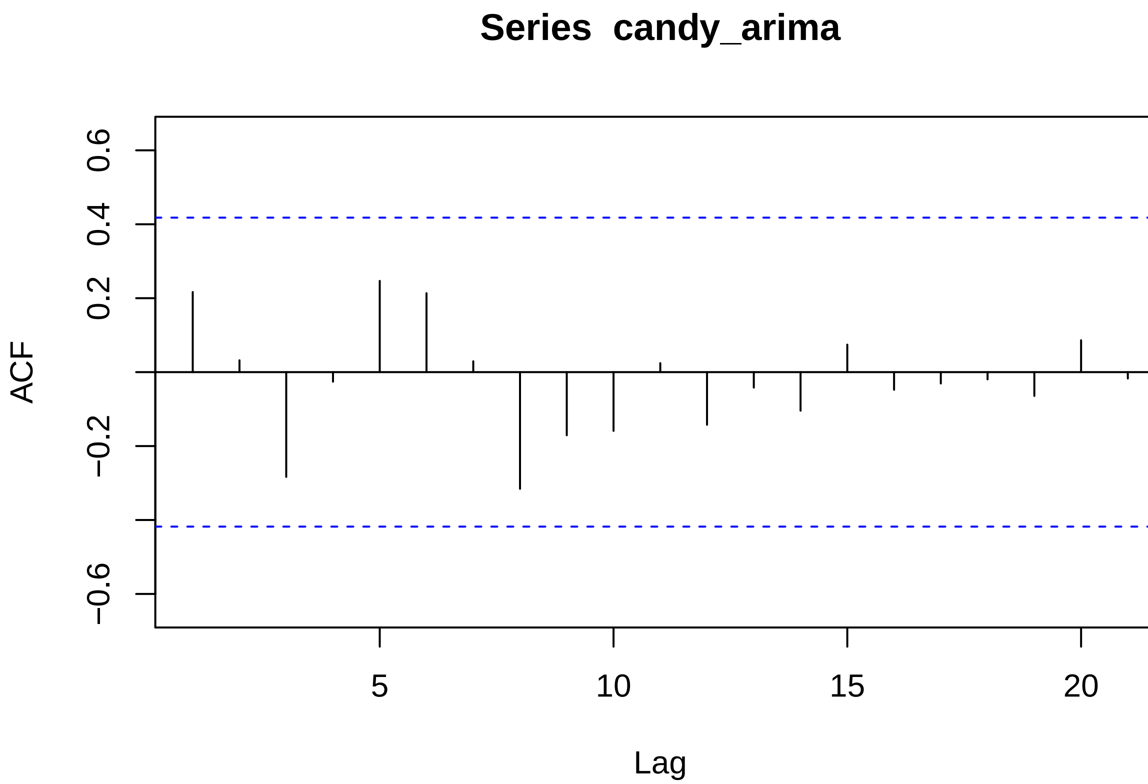

**Time Series chart of the differenced series.**

```r
tsdisplay(candy_arima)
```



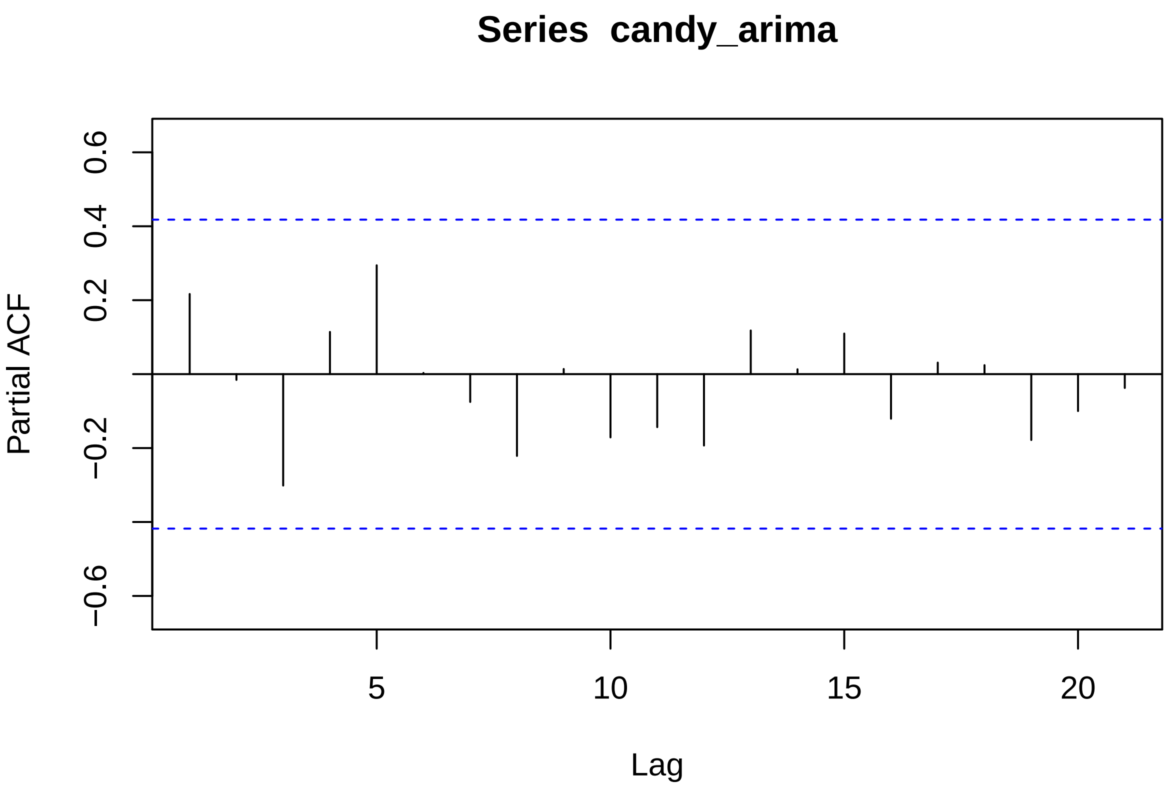

- The time series plot of the differenced series is ploted.
- Also, the tsdiagram of the differenced series is shown.

```
Acf(candy_arima)
```




**ACF and PACF plots**

```
Pacf(candy_arima)
```




**Observations**

- None of the lines in ACF and PACF are crossing the confidence interval.
- This means the p, q, P, Q have a maximum value of zero.
- The possible ARIMA models can be of the format ARIMA(0,1,0)(0,1,0) or ARIMA(0,0,0)(0,1,0) or ARIMA(0,1,0)(0,0,0) or ARIMA(0,0,0)(0,0,0).
- However, the system takes in values of p, q, P, Q values other than 0 as well to cross check if there is any other model that has even lower AIC and BIC.

```r
fit_arima_mod <- auto.arima(candy_ts,trace=TRUE, stepwise = FALSE )
```

**AIC, BIC and Sigma^2 for the possible models**

```
##
##  ARIMA(0,0,0)(0,1,0)[12]                    : 162.3849
##  ARIMA(0,0,0)(0,1,0)[12] with drift         : 137.3626
##  ARIMA(0,0,1)(0,1,0)[12]                    : 155.1324
##  ARIMA(0,0,1)(0,1,0)[12] with drift         : 139.098
##  ARIMA(0,0,2)(0,1,0)[12]                    : 149.3208
##  ARIMA(0,0,2)(0,1,0)[12] with drift         : 140.6968
##  ARIMA(0,0,3)(0,1,0)[12]                    : 150.6092
##  ARIMA(0,0,3)(0,1,0)[12] with drift         : 142.3787
##  ARIMA(0,0,4)(0,1,0)[12]                    : 153.8033
##  ARIMA(0,0,4)(0,1,0)[12] with drift         : 145.7527
##  ARIMA(0,0,5)(0,1,0)[12]                    : Inf
##  ARIMA(0,0,5)(0,1,0)[12] with drift         : 150.1826
##  ARIMA(1,0,0)(0,1,0)[12]                    : 145.6165
##  ARIMA(1,0,0)(0,1,0)[12] with drift         : 139.0092
##  ARIMA(1,0,1)(0,1,0)[12]                    : Inf
##  ARIMA(1,0,1)(0,1,0)[12] with drift         : 142.0215
##  ARIMA(1,0,2)(0,1,0)[12]                    : 150.1784
##  ARIMA(1,0,2)(0,1,0)[12] with drift         : 143.2609
##  ARIMA(1,0,3)(0,1,0)[12]                    : Inf
##  ARIMA(1,0,3)(0,1,0)[12] with drift         : Inf
##  ARIMA(1,0,4)(0,1,0)[12]                    : 157.5498
##  ARIMA(1,0,4)(0,1,0)[12] with drift         : Inf
##  ARIMA(2,0,0)(0,1,0)[12]                    : 146.6259
##  ARIMA(2,0,0)(0,1,0)[12] with drift         : 141.9967
##  ARIMA(2,0,1)(0,1,0)[12]                    : 149.4575
##  ARIMA(2,0,1)(0,1,0)[12] with drift         : 145.1134
##  ARIMA(2,0,2)(0,1,0)[12]                    : Inf
##  ARIMA(2,0,2)(0,1,0)[12] with drift         : Inf
##  ARIMA(2,0,3)(0,1,0)[12]                    : Inf
##  ARIMA(2,0,3)(0,1,0)[12] with drift         : Inf
##  ARIMA(3,0,0)(0,1,0)[12]                    : 149.6439
##  ARIMA(3,0,0)(0,1,0)[12] with drift         : 142.9658
##  ARIMA(3,0,1)(0,1,0)[12]                    : 152.8453
##  ARIMA(3,0,1)(0,1,0)[12] with drift         : 146.5868
##  ARIMA(3,0,2)(0,1,0)[12]                    : Inf
##  ARIMA(3,0,2)(0,1,0)[12] with drift         : Inf
##  ARIMA(4,0,0)(0,1,0)[12]                    : 148.9181
##  ARIMA(4,0,0)(0,1,0)[12] with drift         : 146.0576
##  ARIMA(4,0,1)(0,1,0)[12]                    : 150.0605
##  ARIMA(4,0,1)(0,1,0)[12] with drift         : 149.668
##  ARIMA(5,0,0)(0,1,0)[12]                    : 147.72
##  ARIMA(5,0,0)(0,1,0)[12] with drift         : 147.4963
```

```
##
##
##
##  Best model: ARIMA(0,0,0)(0,1,0)[12] with drift
```

```r
fit_arima_mod
```

```
## Series: candy_ts
## ARIMA(0,0,0)(0,1,0)[12] with drift
##
## Coefficients:
##        drift
##       0.6489
## s.e.  0.0878
##
## sigma^2 = 25.59:  log likelihood = -66.37
## AIC=136.73   AICc=137.36   BIC=138.91
```

- ARIMA model is run automatically and the system selects the model with the last AIC and BIC values.

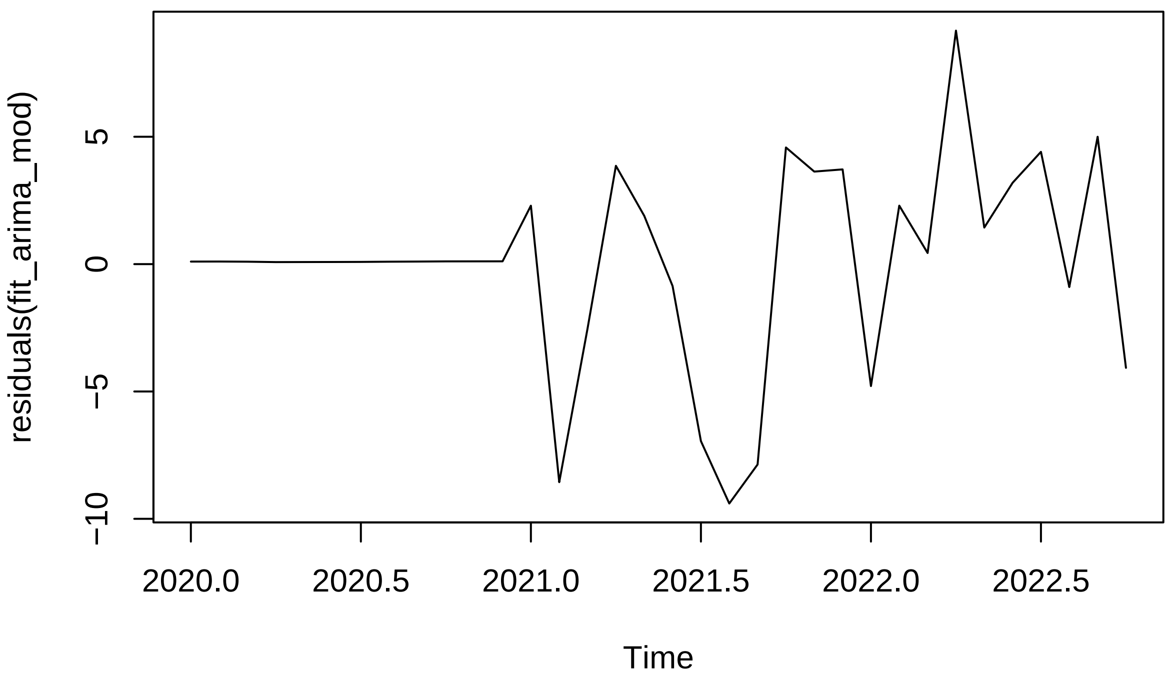
### Best model?

- The model with least AIC and BIC values shall be selected.
- The sigma^2 value should be the highest.

### Final formula for ARIMA with the coefficients

- Final ARIMA formula: ARIMA(0,0,0)(0,1,0)[12] with drift

```r
plot(residuals(fit_arima_mod))
```



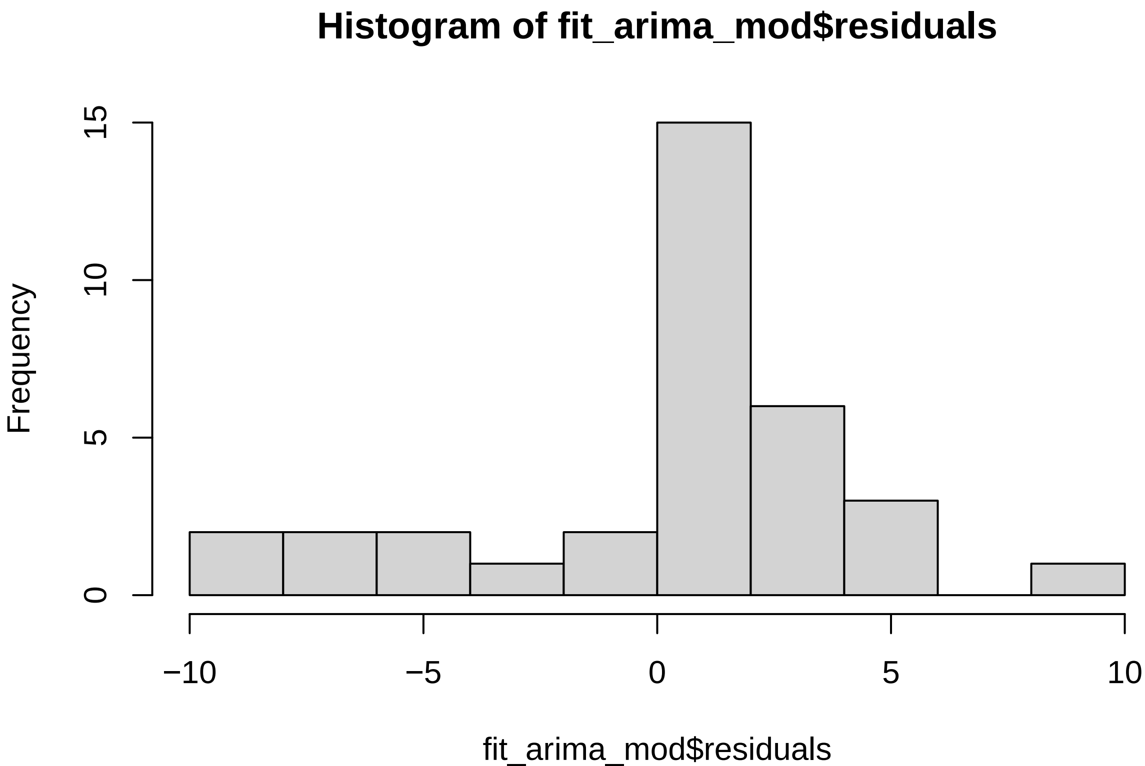

### Residual Analysis

- The residuals appear to be random and also the mean looks to be near zero. We can check this with histogram.

- We can observe a couple of up and downs throughout. But even they did not show and growing residual pattern.

Histogram plot of Residuals

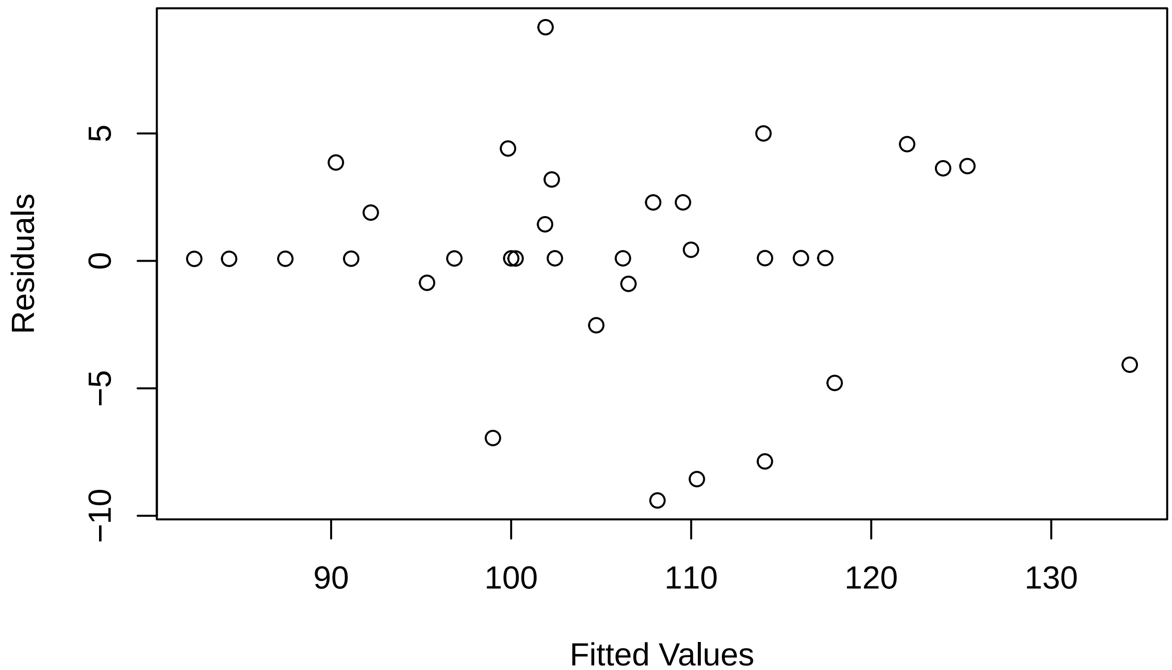
```
hist(fit_arima_mod$residuals)
```




- The histogram appears to be normally distributed.
- But the values do not have a mean zero. The histogram appears to be skewed on one side.
- This means that the data is biased as the mean is not zero.

Fitted values vs. residuals

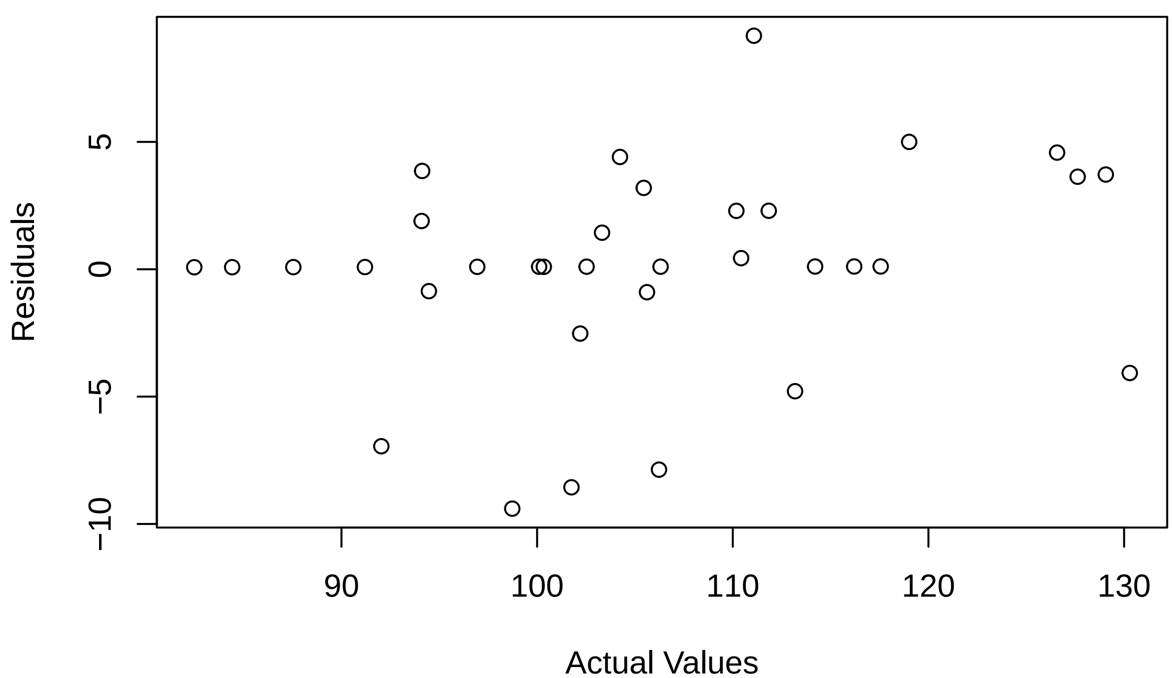
```
plot(as.numeric(fitted(fit_arima_mod)), residuals(fit_arima_mod), type='p', ylab='Residuals', xlab='Fit
```




- The Fitted vs Residuals plot appears to be random and do not have any trend.
- The plot appears to have a mean around zero which is a good sign.
- The plot however seems to have a few outliers.

Actual values vs. residuals

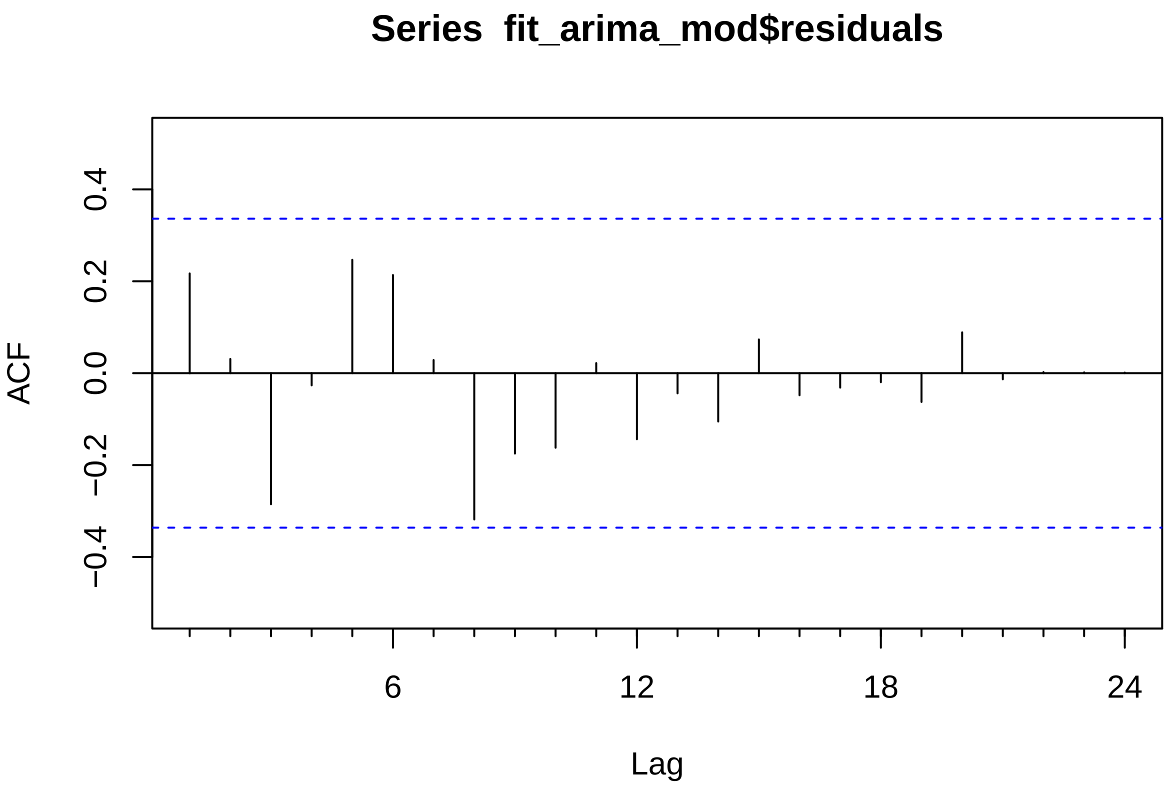
```
plot(as.numeric(candy_ts), residuals(fit_arima_mod), type='p', ylab='Residuals', xlab='Actual Values')
```




- The Actual vs Residuals plot appears to be random and do not have any trend.
- The plot appears to have a mean around zero which is a good sign.
- The plot however seems to have a few outliers.

ACF plot of the residuals

```r
Acf(fit_arima_mod$residuals)
```




- In the Acf plot, none of the values crossed the confidence levels. It appears to be white noice.
- This signifies that the forecast is a good forecast.
- This proves to be the best forecast comparing all the previous ones tested.

```r
accuracy(fit_arima_mod)
```

**Accuracy**

```
##                      ME     RMSE      MAE         MPE     MAPE     MASE
## Training set 0.03380058 3.975377 2.735006 -0.06563588 2.550658 0.341411
##                   ACF1
## Training set 0.2170688
```

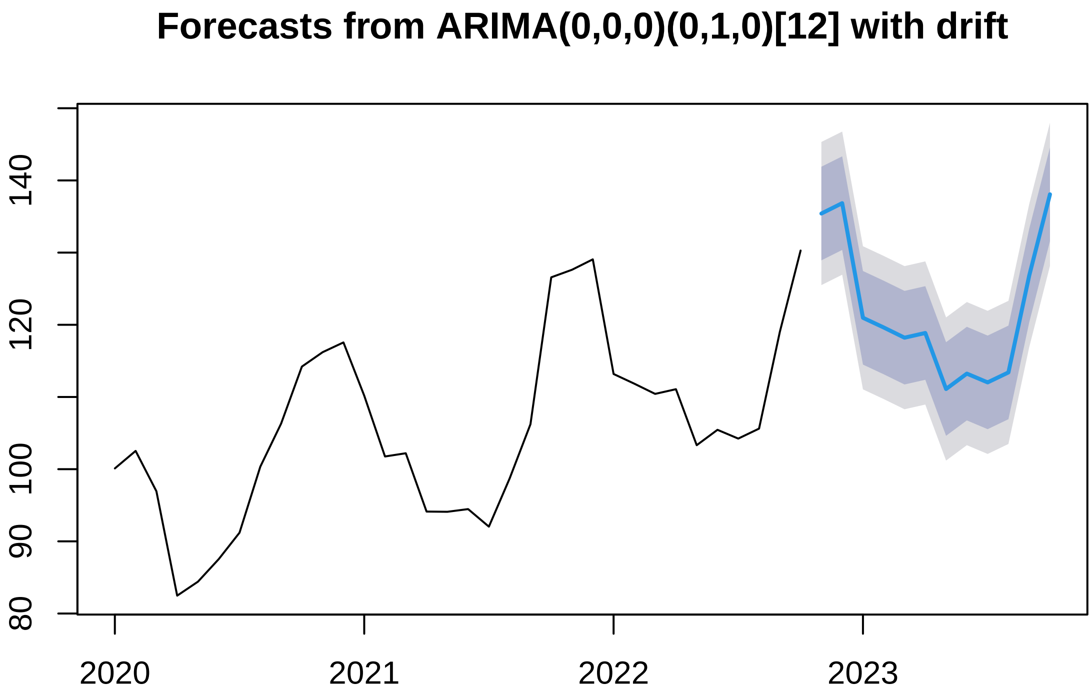
```r
forecast(fit_arima_mod, h=12)
```

**Forecast**

```
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Nov 2022       135.4122 128.9297 141.8947 125.4980 145.3264
## Dec 2022       136.8507 130.3682 143.3332 126.9365 146.7649
## Jan 2023       120.9695 114.4870 127.4520 111.0553 130.8837
## Feb 2023       119.6251 113.1426 126.1076 109.7109 129.5393
## Mar 2023       118.2130 111.7305 124.6955 108.2988 128.1272
## Apr 2023       118.8655 112.3830 125.3480 108.9513 128.7797
## May 2023       111.1055 104.6230 117.5880 101.1913 121.0197
## Jun 2023       113.2364 106.7539 119.7189 103.3222 123.1506
```

```
## Jul 2023       112.0215 105.5390 118.5040 102.1073 121.9357
## Aug 2023       113.4033 106.9208 119.8858 103.4891 123.3175
## Sep 2023       126.8028 120.3203 133.2853 116.8886 136.7170
## Oct 2023       138.0761 131.5936 144.5586 128.1619 147.9903
```

```
plot(forecast(fit_arima_mod, h=12))
```



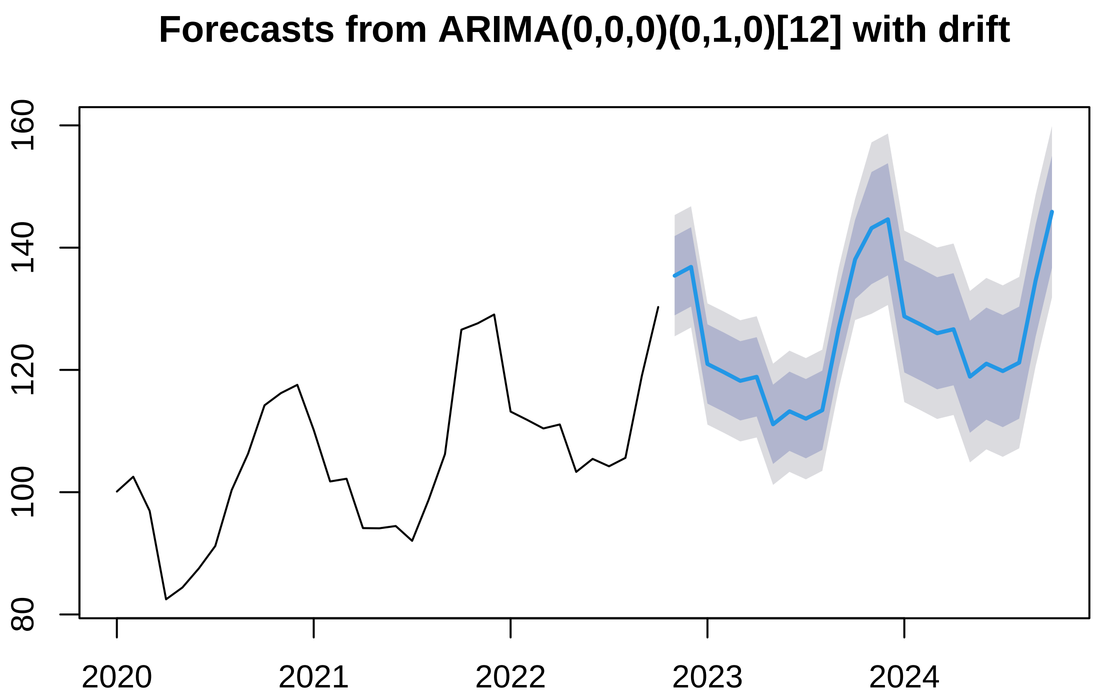

Next two years. Show table and plot

```
forecast(fit_arima_mod, h=24)
```

```
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Nov 2022       135.4122 128.9297 141.8947 125.4980 145.3264
## Dec 2022       136.8507 130.3682 143.3332 126.9365 146.7649
## Jan 2023       120.9695 114.4870 127.4520 111.0553 130.8837
## Feb 2023       119.6251 113.1426 126.1076 109.7109 129.5393
## Mar 2023       118.2130 111.7305 124.6955 108.2988 128.1272
## Apr 2023       118.8655 112.3830 125.3480 108.9513 128.7797
## May 2023       111.1055 104.6230 117.5880 101.1913 121.0197
## Jun 2023       113.2364 106.7539 119.7189 103.3222 123.1506
## Jul 2023       112.0215 105.5390 118.5040 102.1073 121.9357
## Aug 2023       113.4033 106.9208 119.8858 103.4891 123.3175
## Sep 2023       126.8028 120.3203 133.2853 116.8886 136.7170
## Oct 2023       138.0761 131.5936 144.5586 128.1619 147.9903
## Nov 2023       143.1989 134.0312 152.3666 129.1782 157.2196
## Dec 2023       144.6374 135.4697 153.8051 130.6167 158.6581
## Jan 2024       128.7562 119.5885 137.9239 114.7355 142.7769
## Feb 2024       127.4118 118.2441 136.5795 113.3911 141.4325
## Mar 2024       125.9997 116.8320 135.1674 111.9790 140.0204
## Apr 2024       126.6522 117.4845 135.8199 112.6315 140.6729
## May 2024       118.8922 109.7245 128.0599 104.8715 132.9129
## Jun 2024       121.0231 111.8554 130.1908 107.0024 135.0438
## Jul 2024       119.8082 110.6405 128.9759 105.7875 133.8289
## Aug 2024       121.1900 112.0223 130.3577 107.1693 135.2107
```

```
## Sep 2024       134.5895 125.4218 143.7572 120.5688 148.6102
## Oct 2024       145.8628 136.6951 155.0305 131.8421 159.8835
```

```r
plot(forecast(fit_arima_mod, h=24))
```




### ARIMA Summary

- The ME and RMSE values are quite low compared to our previous forecasts.
- And all the residual plots also seem random.
- Considering all these, the ARIMA model seems to be the best forecasting model compared to all the other models that were done above.
- ARIMA models appear to be the best forecast considering all the previous forecast methods.
- Considering both accuracy numbers and the residual analysis, ARIMA proves to be the best forecasting model.

## Accuracy Summary

```r
accuracy(naive_for)
```

```
##                     ME     RMSE      MAE       MPE     MAPE      MASE      ACF1
## Training set 0.9147333 7.399605 5.399739 0.5619459 5.090241 0.6740498 0.3322229
```

```r
accuracy(ses_fit)
```

```
##                     ME    RMSE      MAE       MPE     MAPE      MASE      ACF1
## Training set 0.8882407 7.29022 5.241508 0.5457879 4.941071 0.6542978 0.3312411
```

```r
accuracy(hw_add)
```

```
##                      ME     RMSE      MAE         MPE     MAPE      MASE
## Training set 0.05597211 4.222618 3.252036 -0.05703846 3.078744 0.4059518
##                   ACF1
## Training set 0.1842016
```

```r
accuracy(fit_arima_mod)
```

```
##                      ME     RMSE      MAE         MPE     MAPE     MASE
## Training set 0.03380058 3.975377 2.735006 -0.06563588 2.550658 0.341411
##                   ACF1
## Training set 0.2170688
```

### Best & Worst Forecasts

- To start with, there is nothing like best or worst forecast.
- Considering the accuracy data above, ARIMA forecast seems to fit the time series the best as it has the least error values (ME, RMSE).
- And naive forecast seems to be the worst as it has the largest ME and RMSE values.

## Conclusion

- The data seemed to have seasonality initially.
- Later, we can consider a window function of the data, which has trend and seasonality from 2020.
- Based on the four forecasting methods, naive, simple smoothing, HoltWinters, and ARIMA, we can see that ARIMA forecast is the better method.
- This is because the forecast fits perfectly, and the error values are pretty low for the ARIMA forecast.
- HoltWinters and ARIMA models have fewer error values than naive and straightforward smoothing. However, HoltWinters has deviations in its residual plots compared to the ARIMA.
- In conclusion, the ARIMA forecast is the best forecasting model considering both error numbers (accuracy) and the residual analysis.
- Based on the analysis and forecast, the time series will increase over the next year and two years.