Forcasting

2023-04-11

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
#Load/install required package here
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

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

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

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

```
library(Kendall)
library(tseries)
library(outliers)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr   1.1.0     v stringr 1.5.0
## v forcats 1.0.0     v tibble  3.2.1
## v purrr   1.0.1     v tidyr   1.3.0
## v readr   2.1.4

## -- 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
```

```
library(smooth)
```

```
## Loading required package: greybox
## Package "greybox", v1.0.7 loaded.
##
##
## Attaching package: 'greybox'
##
## The following object is masked from 'package:tidyr':
```

```
##
##     spread
##
## The following object is masked from 'package:lubridate':
##
##     hm
##
## This is package "smooth", v3.2.0
```

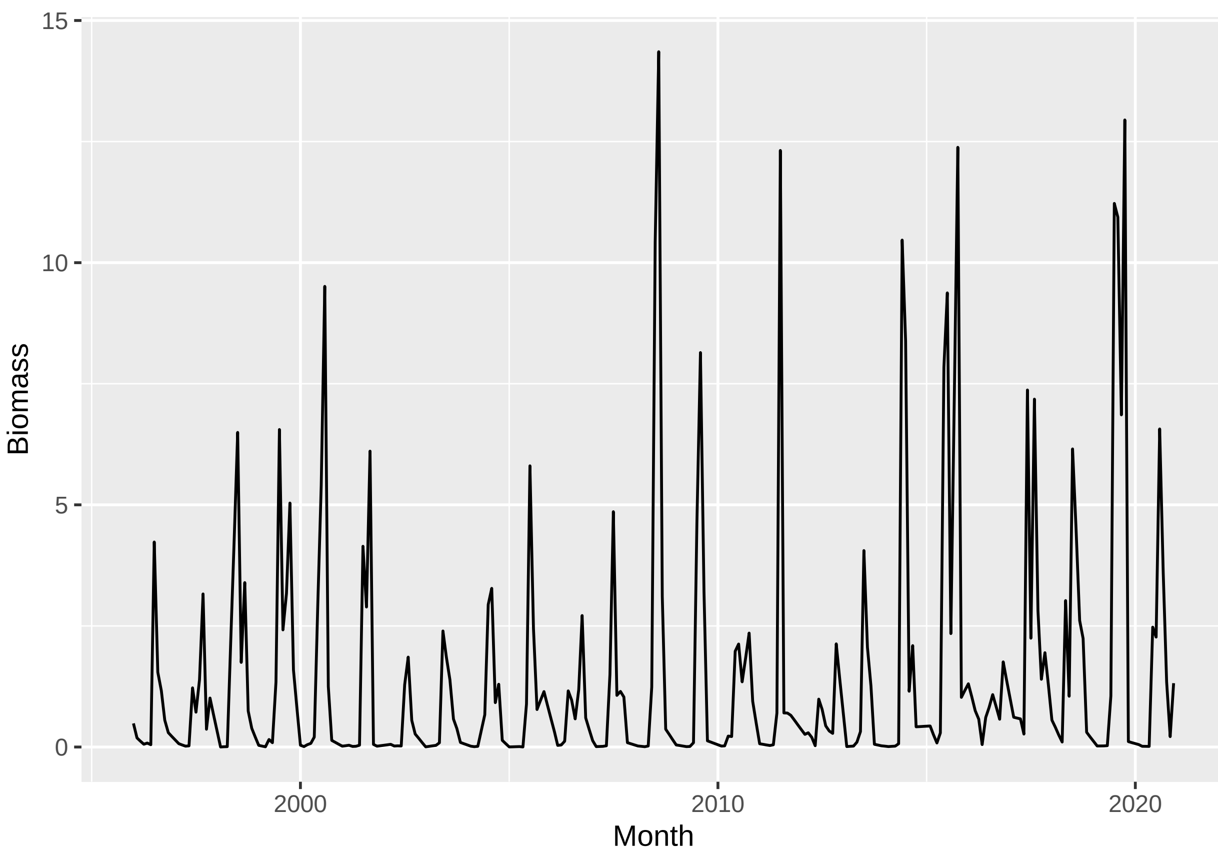
```
#New package for M9 to assist with tables
#install.packages("kableExtra")
library(kableExtra)
```

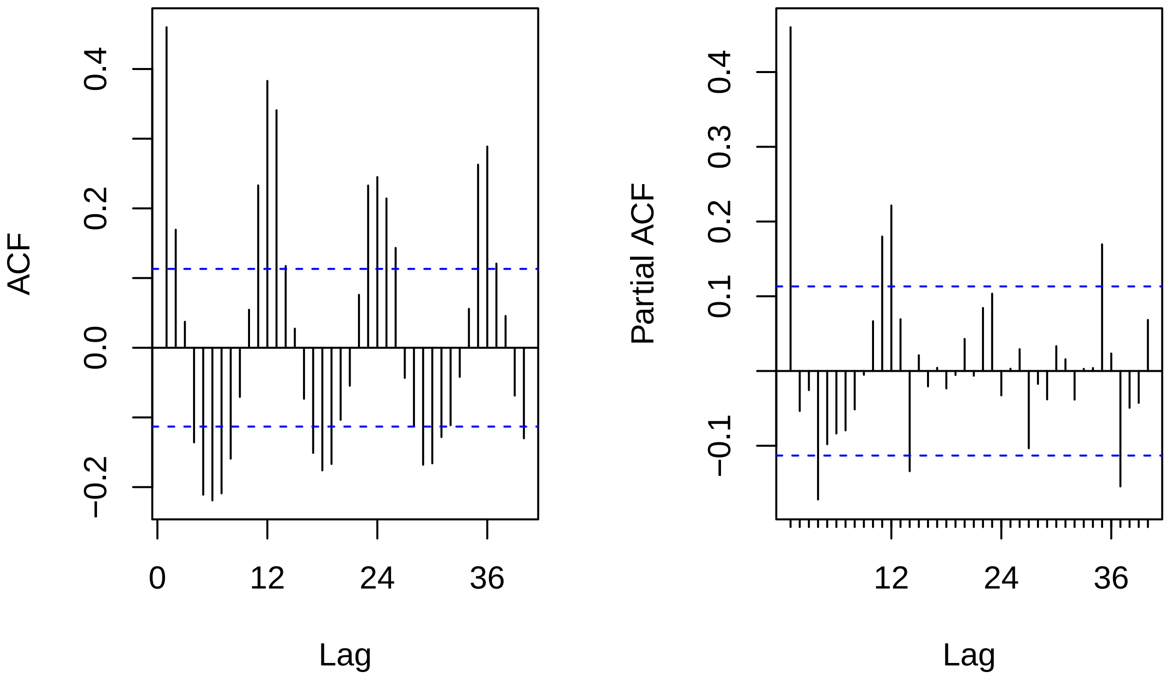
```
## Warning in !is.null(rmarkdown::metadata$output) && rmarkdown::metadata$output
## %in% : 'length(x) = 2 > 1' in coercion to 'logical(1)'
```

```
##
## Attaching package: 'kableExtra'
##
## The following object is masked from 'package:dplyr':
##
##     group_rows
```

(1) Auto-regression

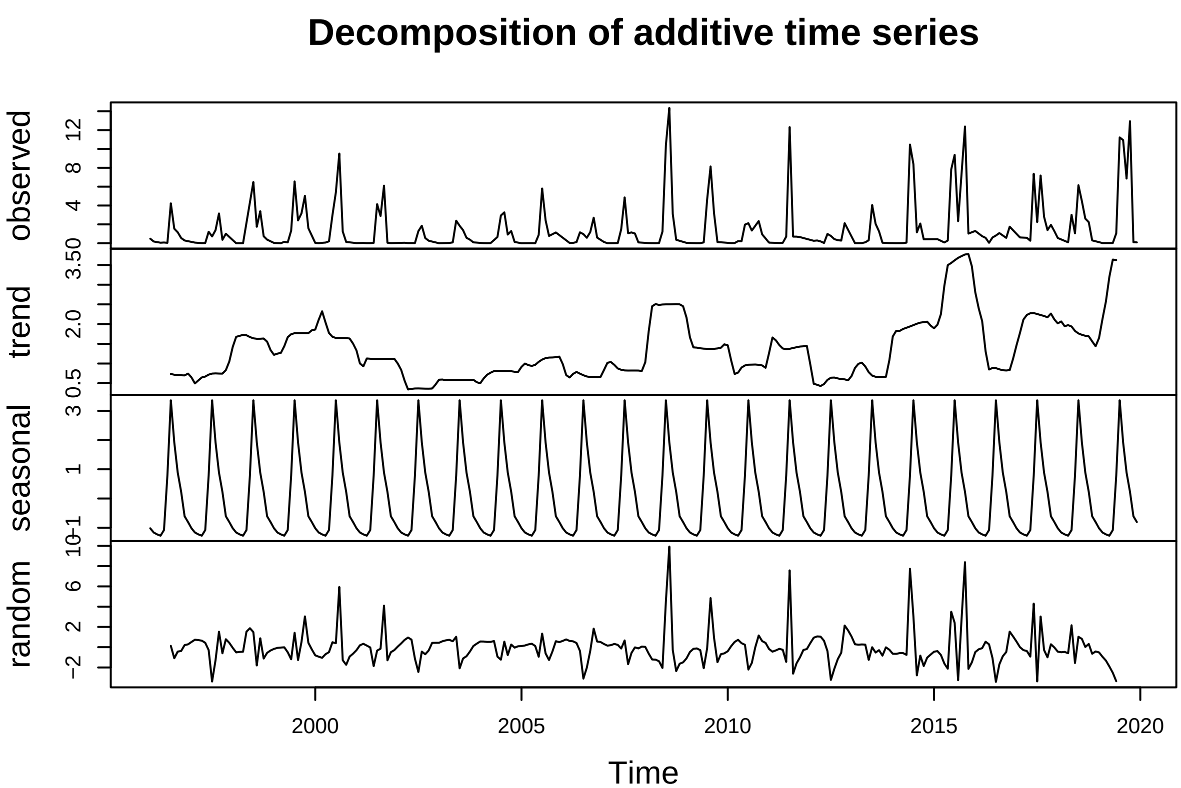
Using the full sample from 1996 to 2020, The original plot, Autocorrelation Function (ACF), and the partial autocorrelation function (PACF) plots are shown as follows:





It can be seen that there are both clear trend and seasonal components in this plot. The decomposition of the time series is shown as follows:

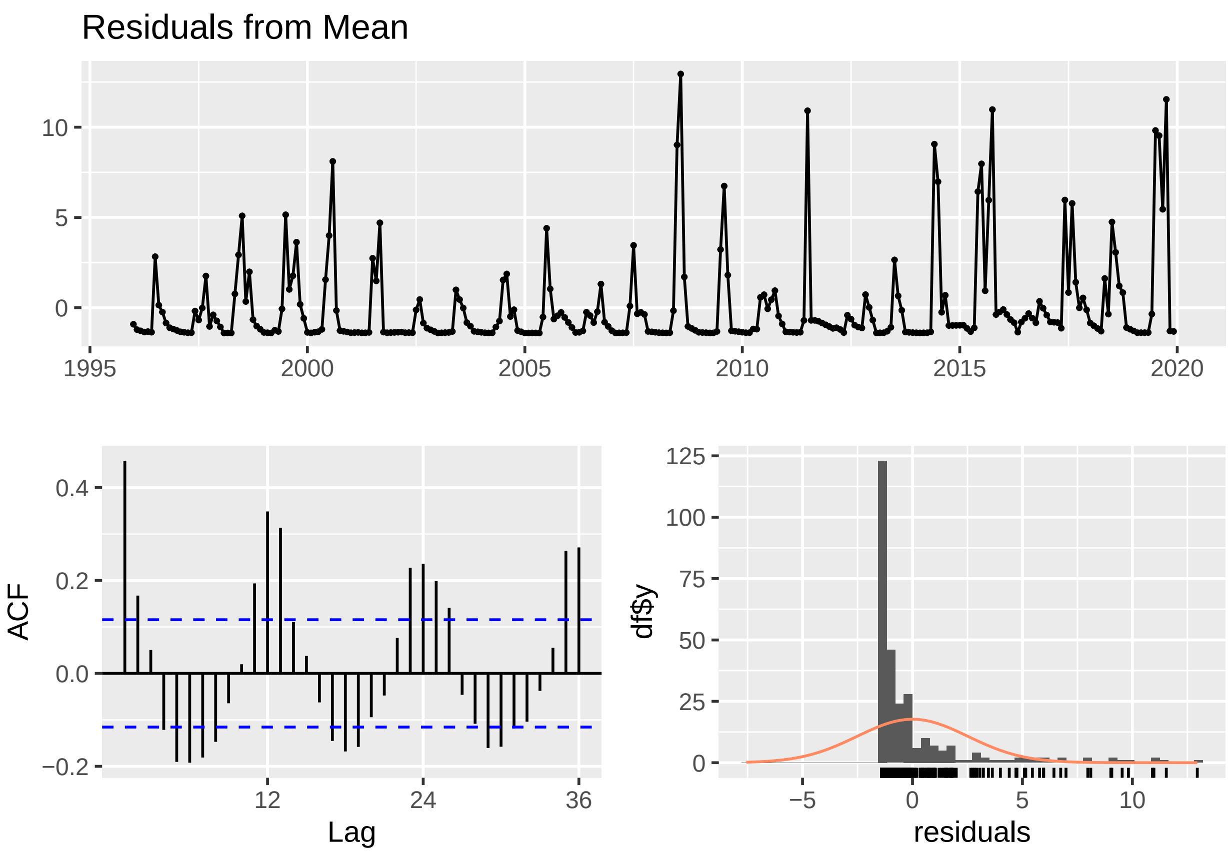
```
#Plot ts decompose
decompose_biomass_data <- decompose(ts_biomass,"additive")

plot(decompose_biomass_data)
```



Given the complexity of the series, we further try five models to fit the model and conduct forecasting.

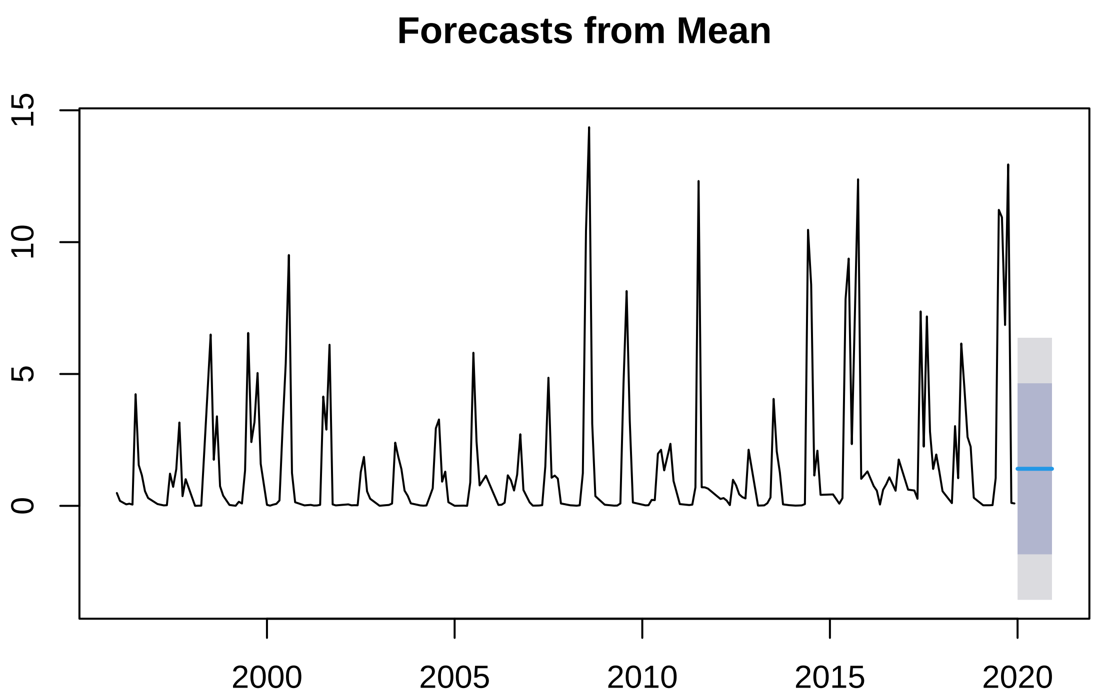
The method includes Arithmetic mean, Seasonal naive, SARIMA, SS Exponential smoothing, and BSM (SS with StructTS). We also present their forecast plots and residual analysis plots individually. We first leave the last year which is the year of 2020 out for the purpose of further comparison in the accuracy test.

```
# Model 1: Arithmetic mean
# The meanf() has no holdout option
MEAN_seas <- meanf(y = ts_biomass, h = 12)
checkresiduals(MEAN_seas)
```

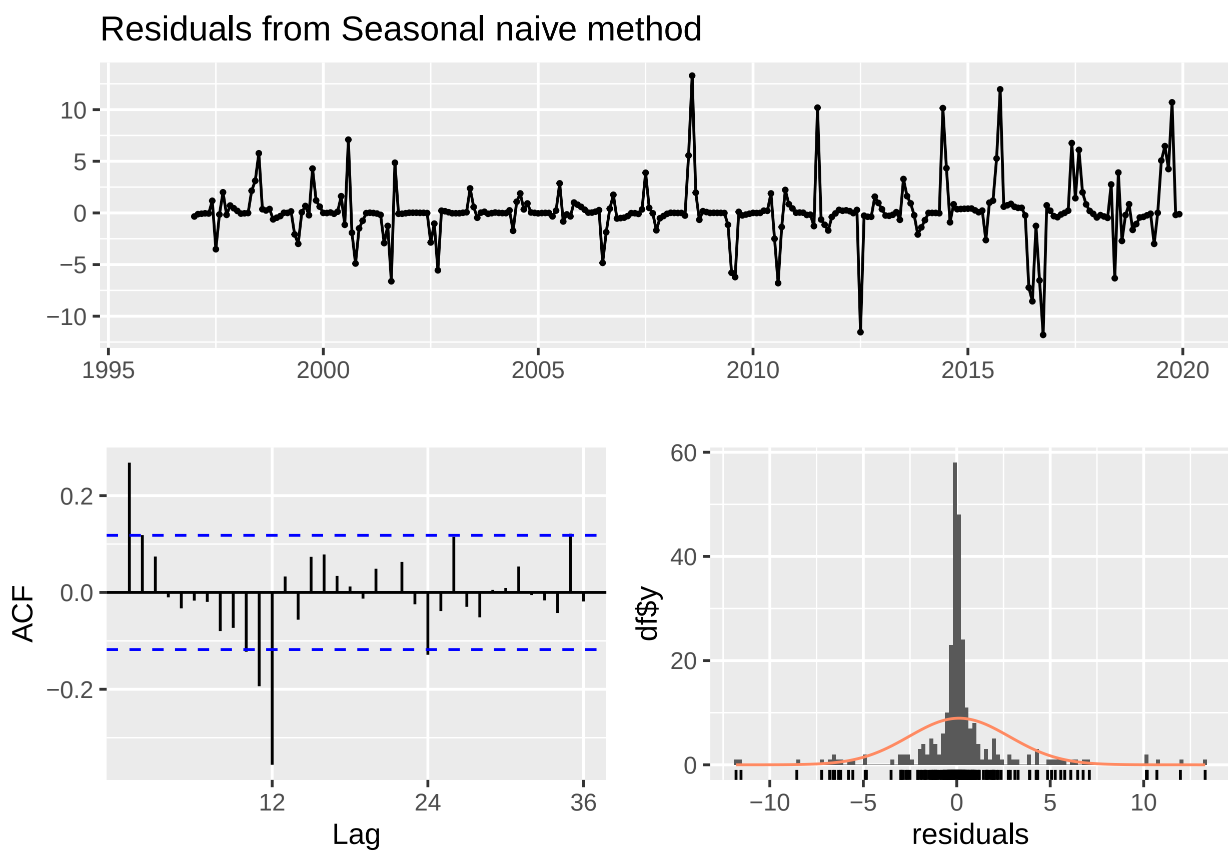



```
##
##  Ljung-Box test
##
## data:  Residuals from Mean
## Q* = 258.95, df = 23, p-value < 2.2e-16
##
## Model df: 1.   Total lags used: 24
```

```
plot(MEAN_seas)
```



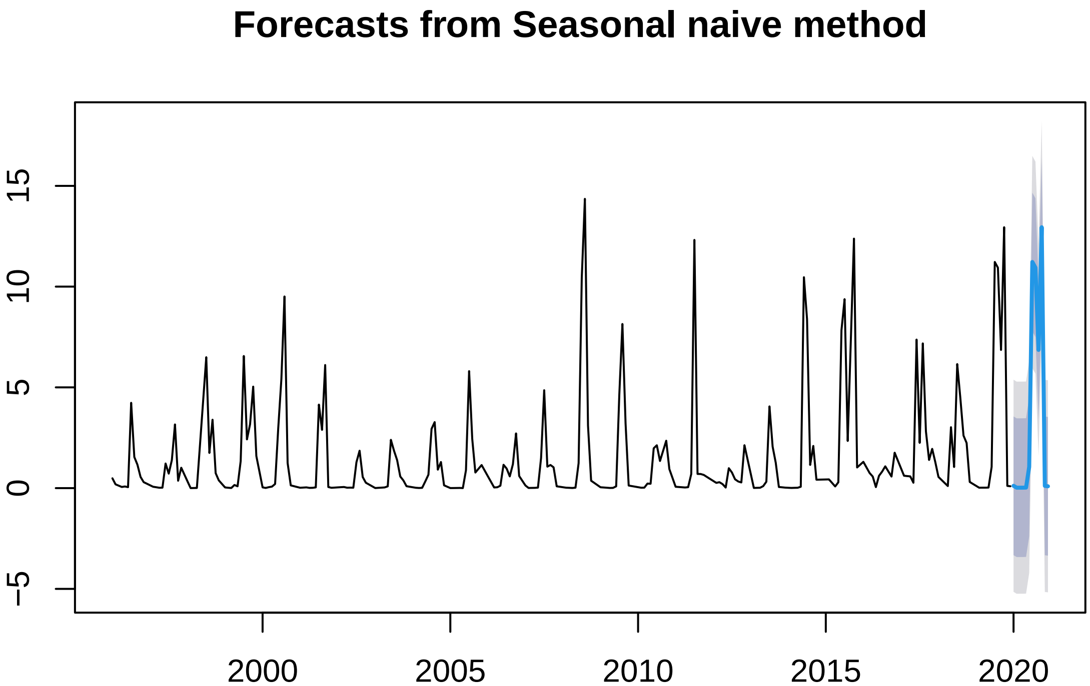

```
# Model 2: Seasonal naive
SNAIVE_seas <- snaive(ts_biomass, h=12, holdout=FALSE)
checkresiduals(SNAIVE_seas)
```



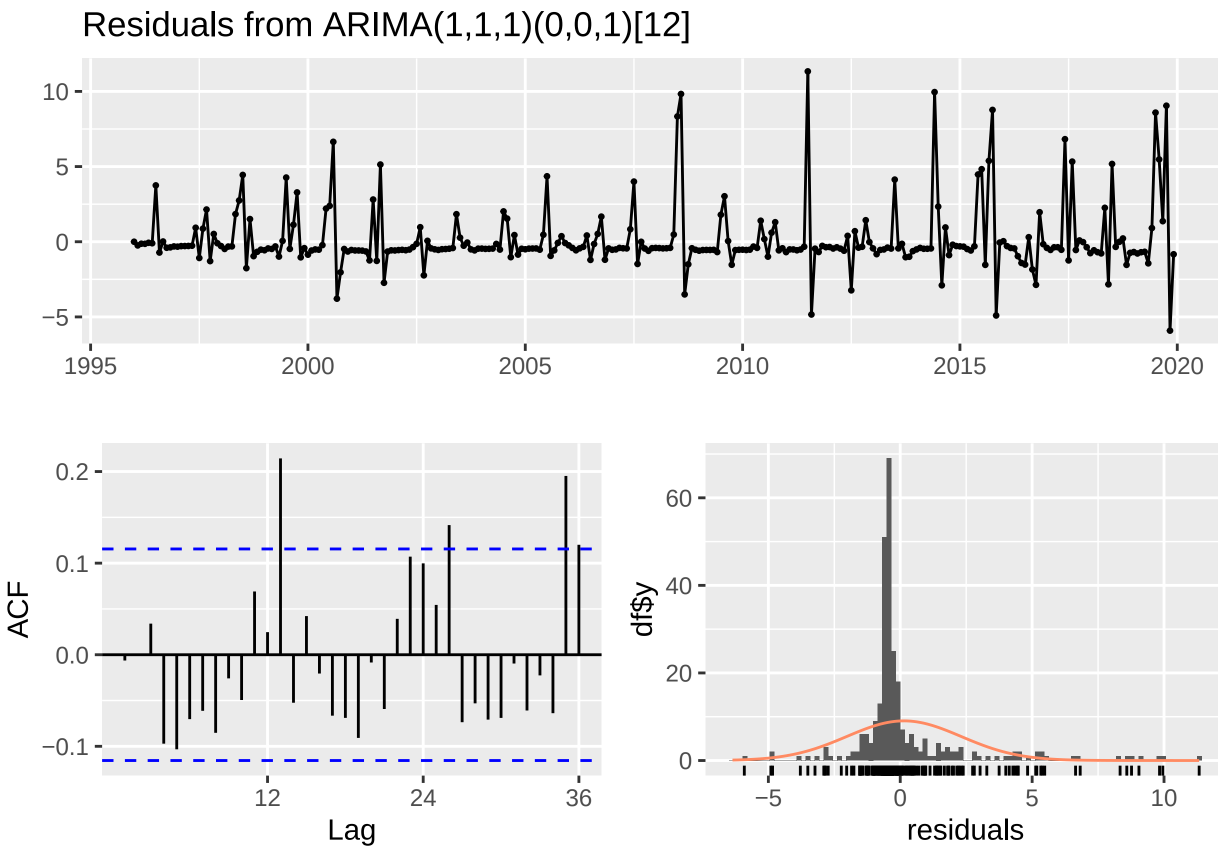

```
##
## 	Ljung-Box test
```

```
##
## data:  Residuals from Seasonal naive method
## Q* = 93.651, df = 24, p-value = 3.556e-10
##
## Model df: 0.   Total lags used: 24
```

```
plot(SNAIVE_seas)
```

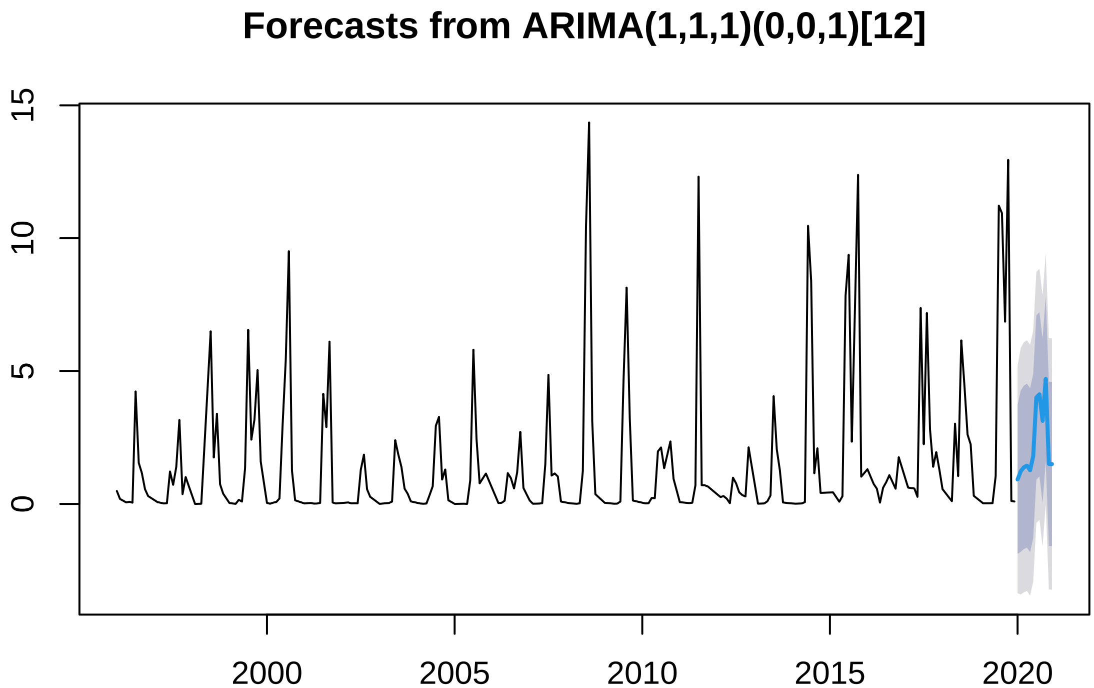



```
# Model 3: SARIMA

SARIMA_autofit <- auto.arima(ts_biomass)
checkresiduals(SARIMA_autofit)
```

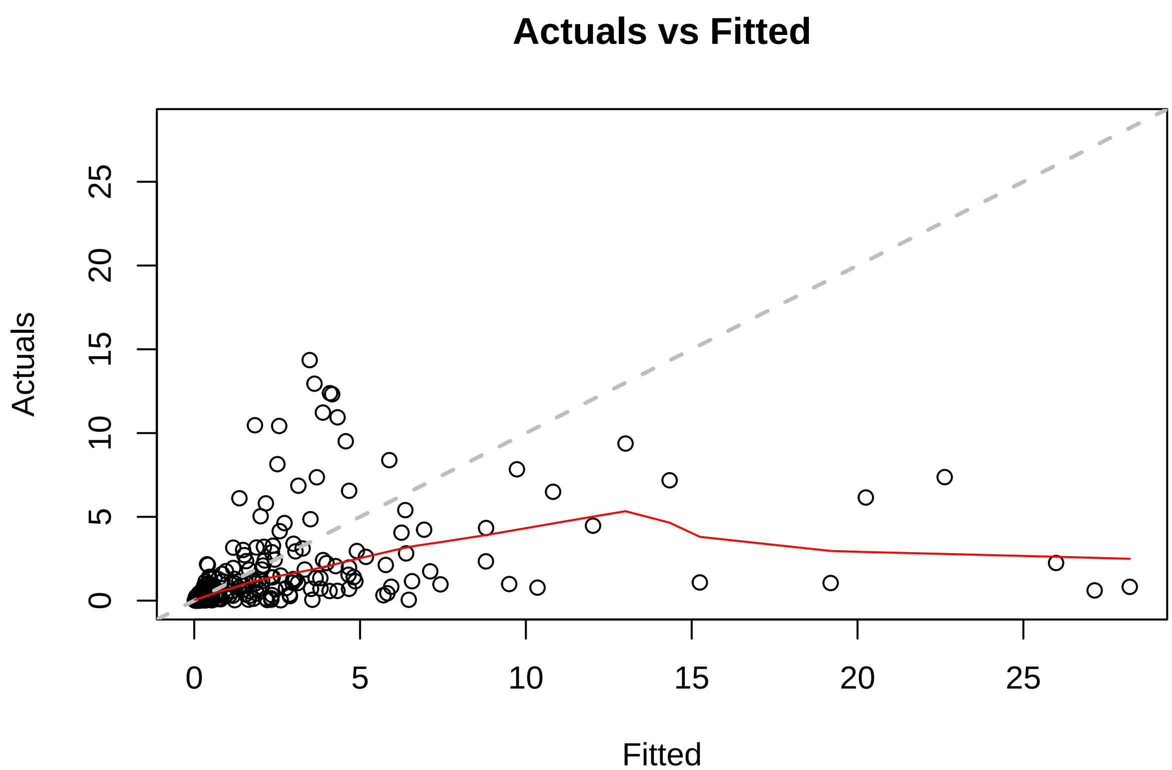


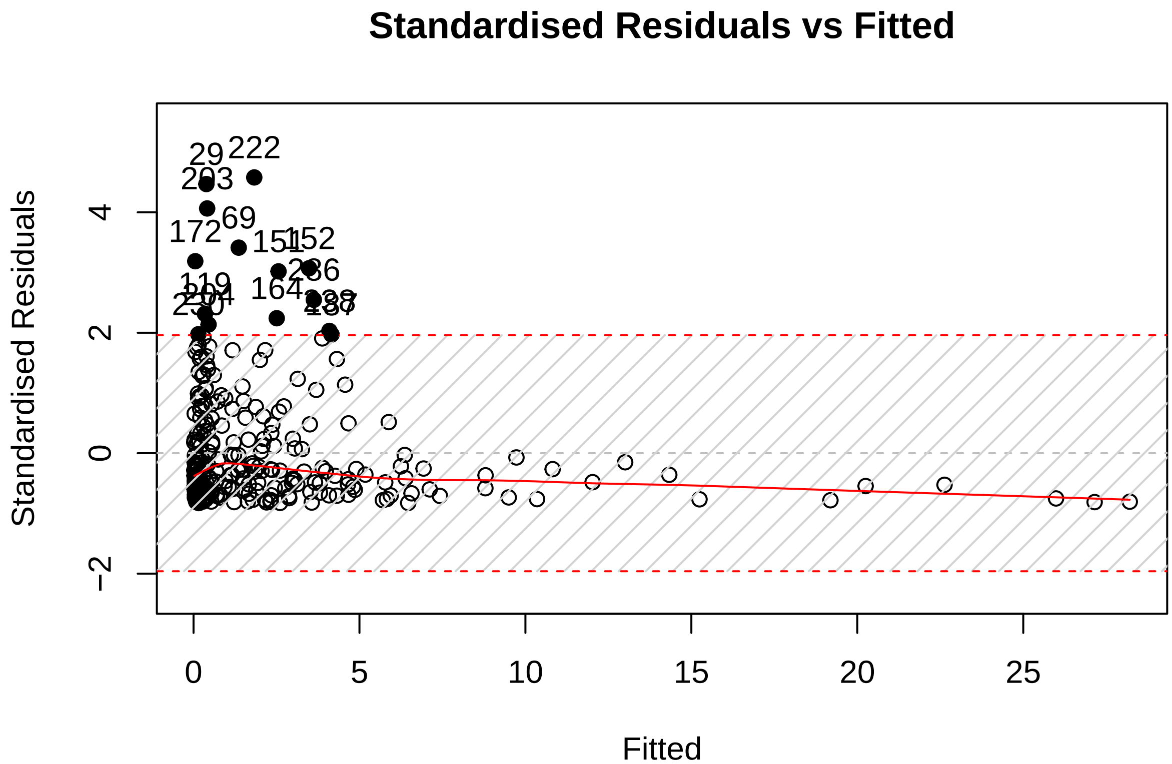

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,1,1)(0,0,1)[12]
## Q* = 42.8, df = 21, p-value = 0.003333
##
## Model df: 3.   Total lags used: 24
```

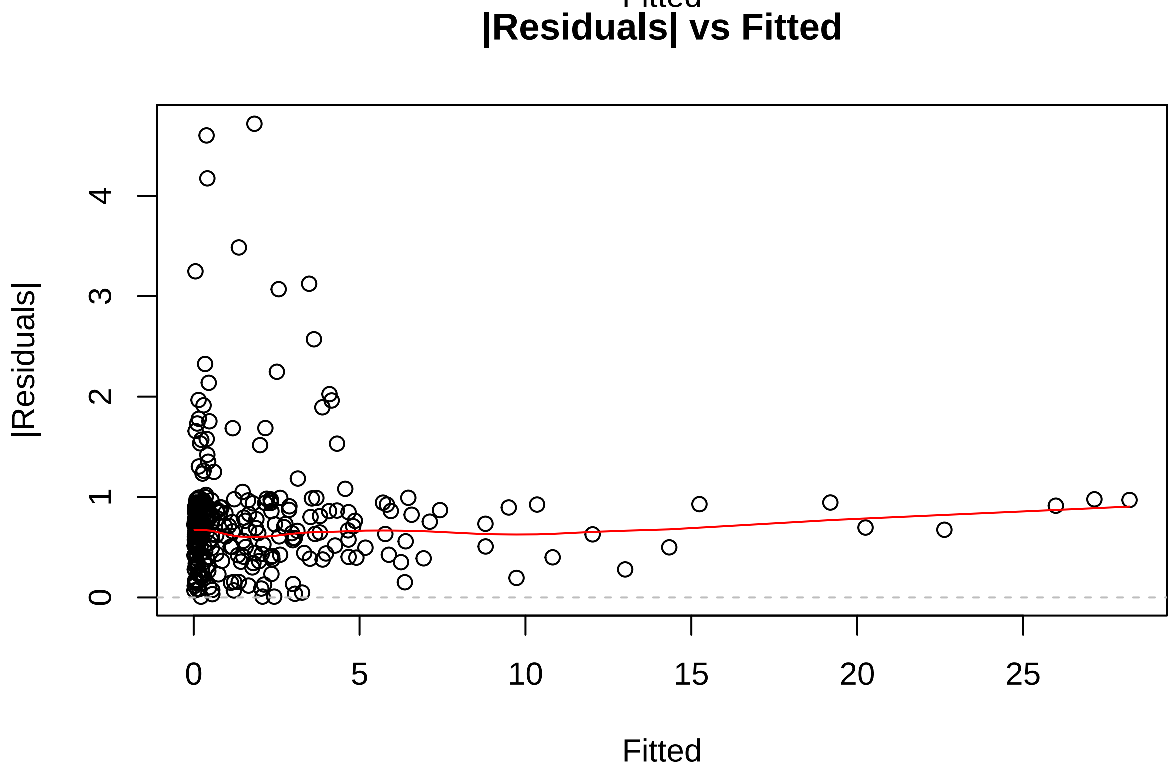
```
#Generating forecasts

SARIMA_for <- forecast(SARIMA_autofit,h=12)
plot(SARIMA_for)
```

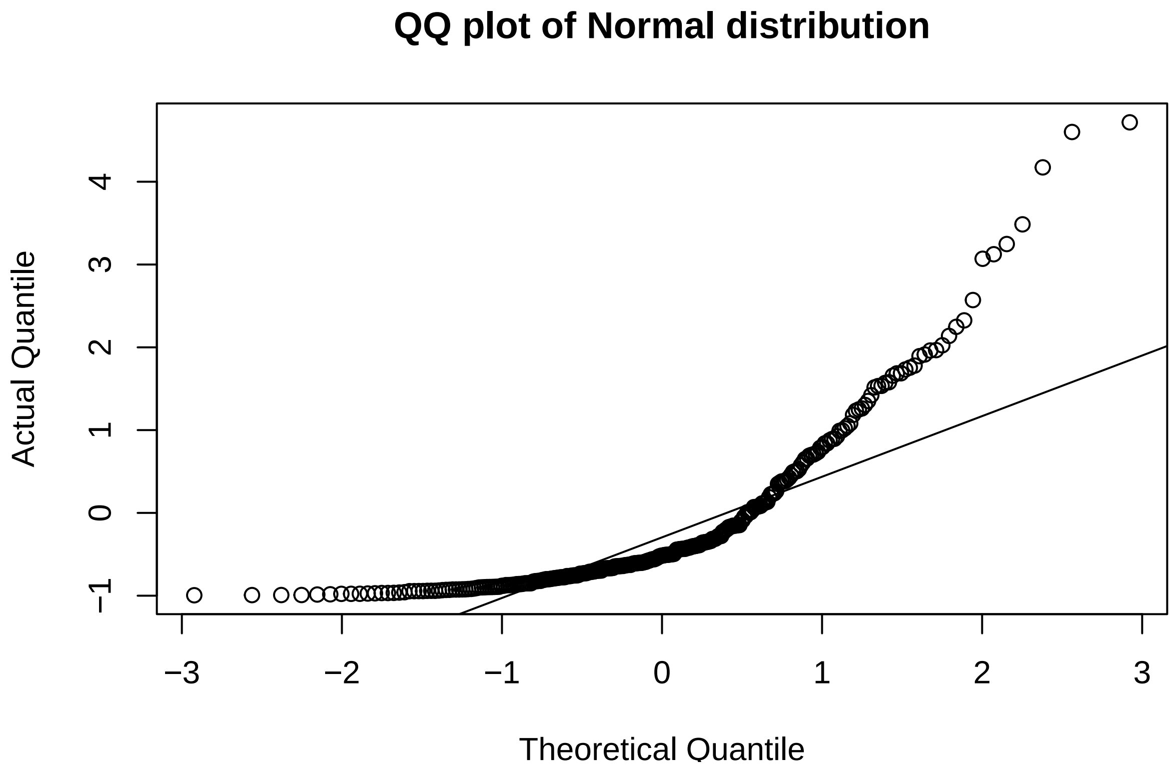



```
# Model 4: SS Exponential smoothing
SSES_seas <- es(ts_biomass,model="ZZZ",h=12,holdout=FALSE)
plot(SSES_seas)
```



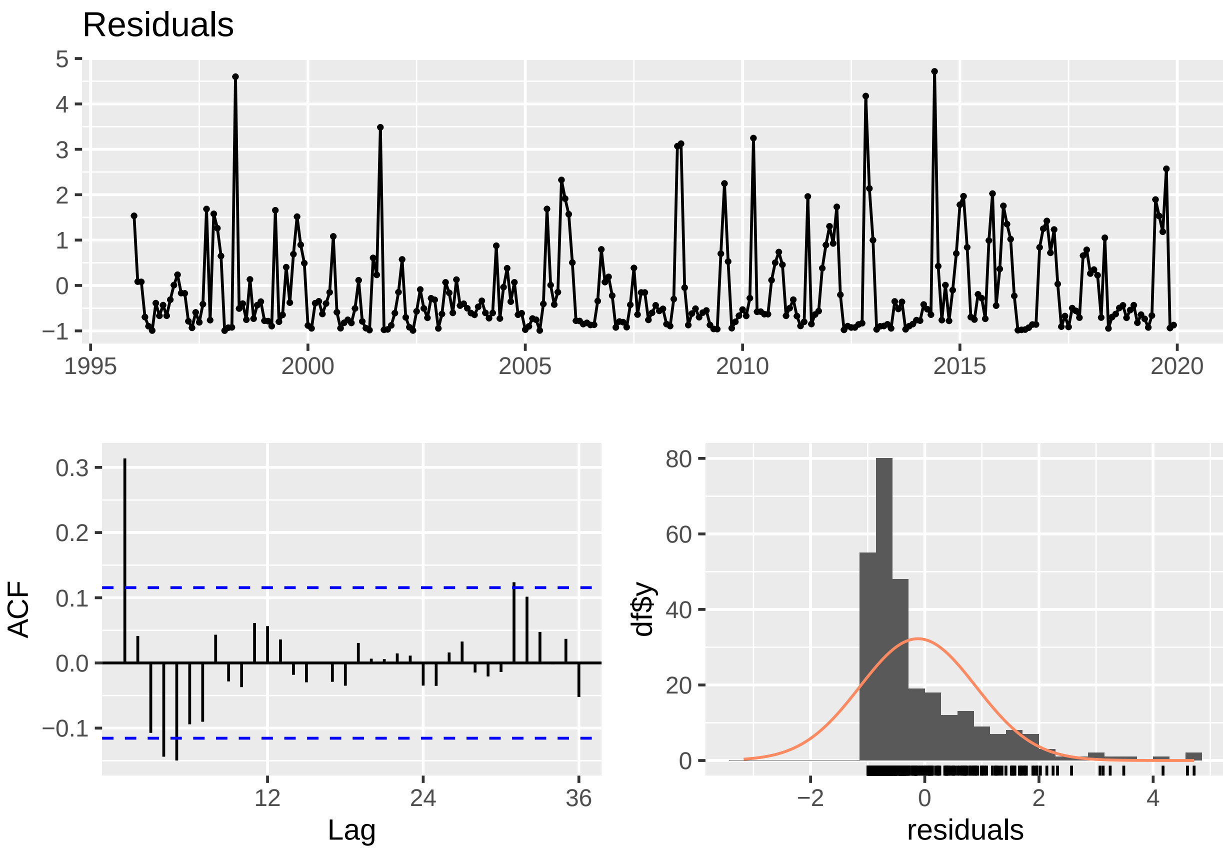

Standardised Residuals vs Fitted
Standardised Residuals
Fitted
29
222
203
69
172
151
152
236
119
164
0
5
10
15
20
25
−2
0
2
4


|Residuals| vs Fitted
|Residuals|
Fitted
0
5
10
15
20
25
0
1
2
3
4



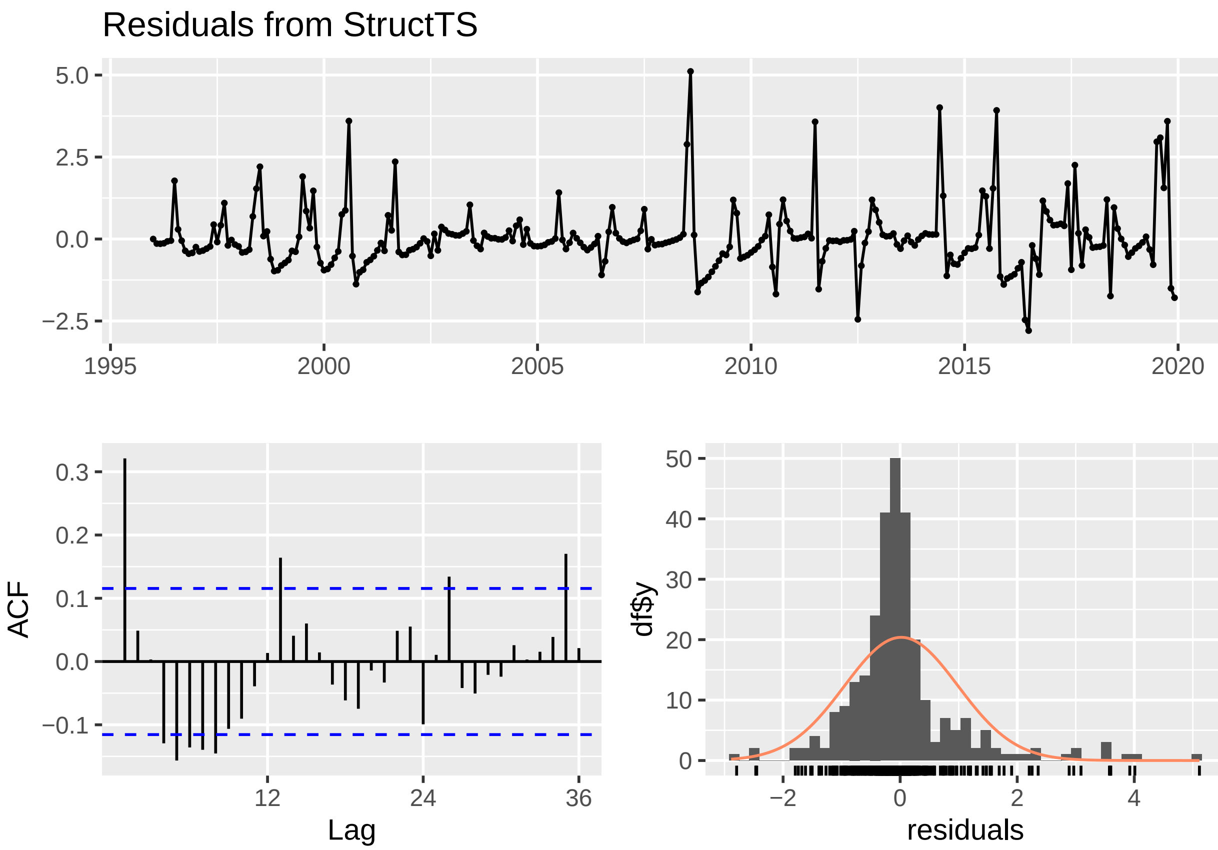

```
checkresiduals(SSES_seas)
```

```
## Warning in modeldf.default(object): Could not find appropriate degrees of
## freedom for this model.
```

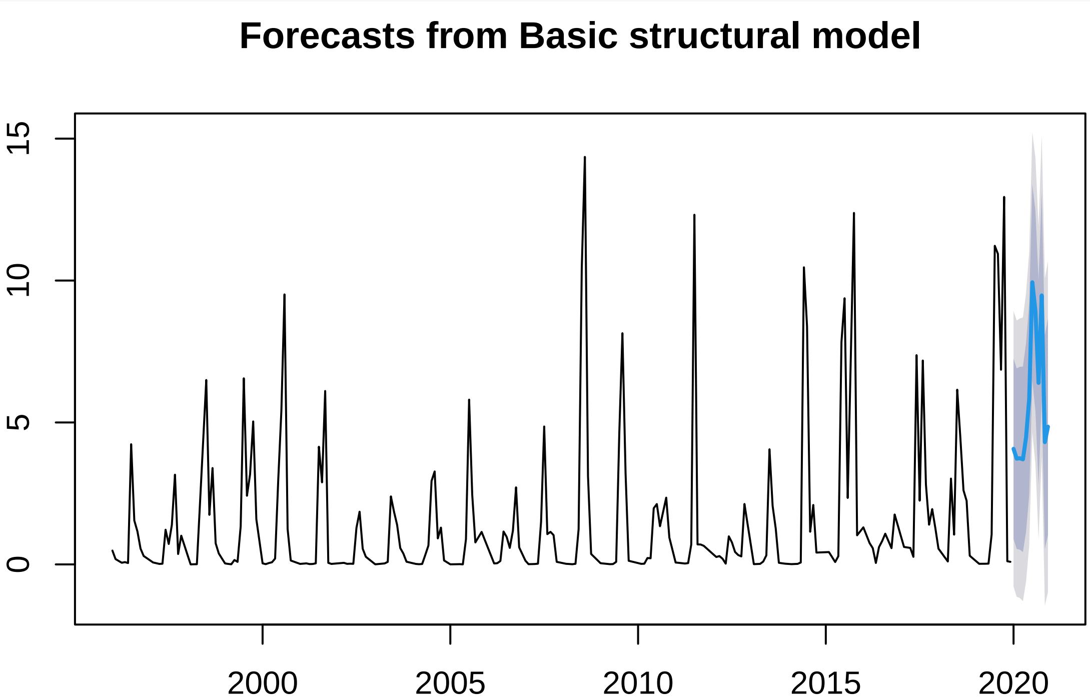



```r
# Model 5: SS with StructTS()

SS_seas <- StructTS(ts_biomass,
                    type="BSM",fixed=c(0,0.001,0.3,NA))    #this function has convergence issues
checkresiduals(SS_seas)
```

```
## Warning in modeldf.default(object): Could not find appropriate degrees of
## freedom for this model.
```




```
#Generating forecasts
SS_for <- forecast(SS_seas,h=12)
plot(SS_for)
```



Based on the residuals analysis plots, SARIMA, Exponential smoothing, and BSM shows better distribution in residuals with a near-random distribution and, most points inside the ACF confidence interval without

Table 1: Forecast Accuracy for Seasonal Data

	ME	RMSE	MAE	MPE	MAPE
MEAN	0.09077	1.91972	1.45817	-2993.9993	3030.8195
SNAIVE	-2.12360	4.54636	2.58677	-122.4177	164.9659
SARIMA	-0.75532	1.58841	1.35418	-2867.0051	2879.9488
SSES	-2.55079	4.40694	2.64893	-998.5011	1005.9502
BSM	-4.29032	4.61610	4.29032	-8743.8441	8743.8441

seasonality. However, it is not intuitive which model is the best. We further conduct a more rigorous accuracy test for each model by comparing the forecasted value in the last year which is 2020 with the observed value. The result of the accuracy test is conducted as follows:

```r
#Model 1: Arithmetic mean
MEAN_scores <- accuracy(MEAN_seas$mean,last_obs)

#Model 2: Seasonal naive
SNAIVE_scores <- accuracy(SNAIVE_seas$mean,last_obs)

# Model 3:  SARIMA
SARIMA_scores <- accuracy(SARIMA_for$mean,last_obs)

# Model 4:  SSES
SSES_scores <- accuracy(SSES_seas$forecast,last_obs)

# Model 5:  BSM
SS_scores <- accuracy(SS_for$mean,last_obs)
```

To present the accuracy result in a comparable way, I created a compare performance metrics to select the model with the lowest RMSE in Table 5. The process of compare performance metrics are shown as follows:

```r
#create data frame
seas_scores <- as.data.frame(rbind(MEAN_scores, SNAIVE_scores, SARIMA_scores,SSES_scores,SS_scores))
row.names(seas_scores) <- c("MEAN", "SNAIVE","SARIMA","SSES","BSM")

#choose model with lowest RMSE
best_model_index <- which.min(seas_scores[,"RMSE"])
cat("The best model by RMSE is:", row.names(seas_scores[best_model_index,]))
```

```
## The best model by RMSE is: SARIMA
```

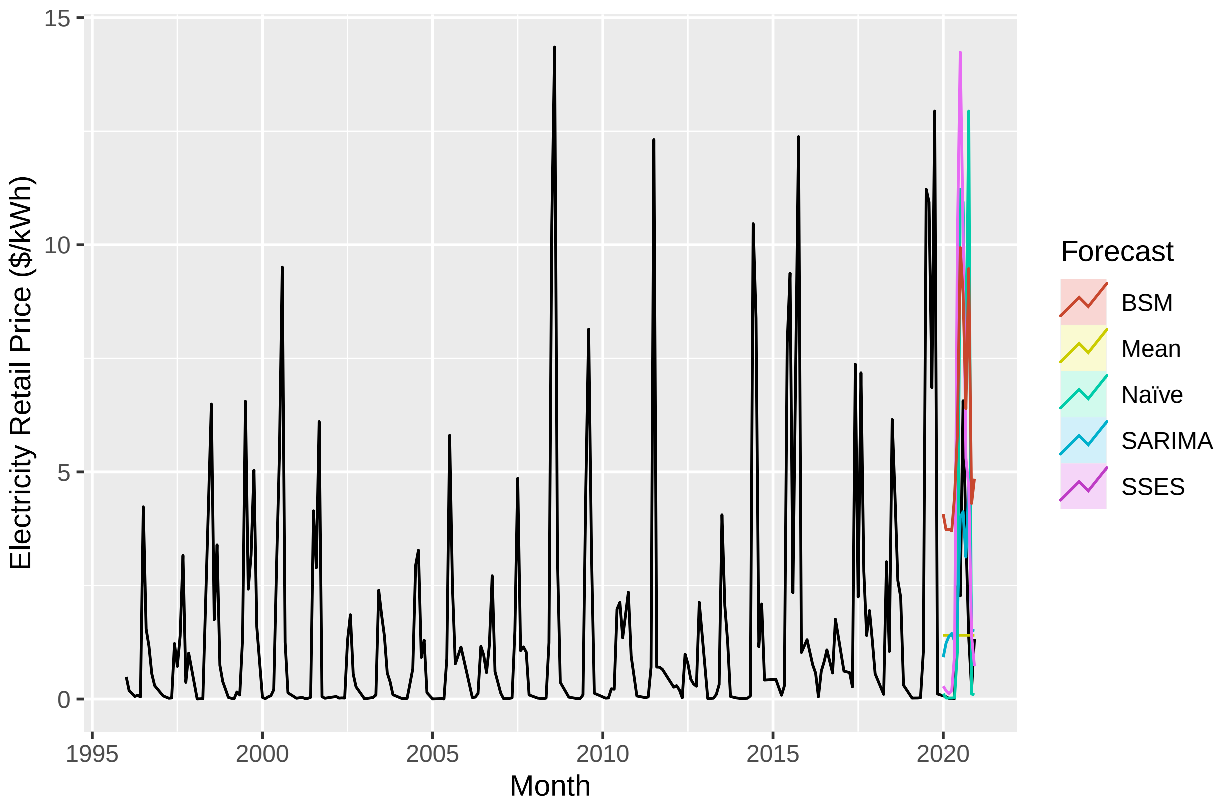
```r
kbl(seas_scores,
      caption = "Forecast Accuracy for Seasonal Data",
      digits = array(5,ncol(seas_scores))) %>%
  kable_styling(full_width = FALSE, position = "center") %>%
  #highlight model with lowest RMSE
  kable_styling(latex_options="striped", stripe_index = which.min(seas_scores[,"RMSE"]))
```

Based on highlighted lowest RMSE in the compare performance metrics, the model with the best fit is the SARIMA model.

To visually compare the result of forecasts using different models, we further jointly plot the forecast generated using five models and compare them with the actual observed value. The plot is shown as follows:

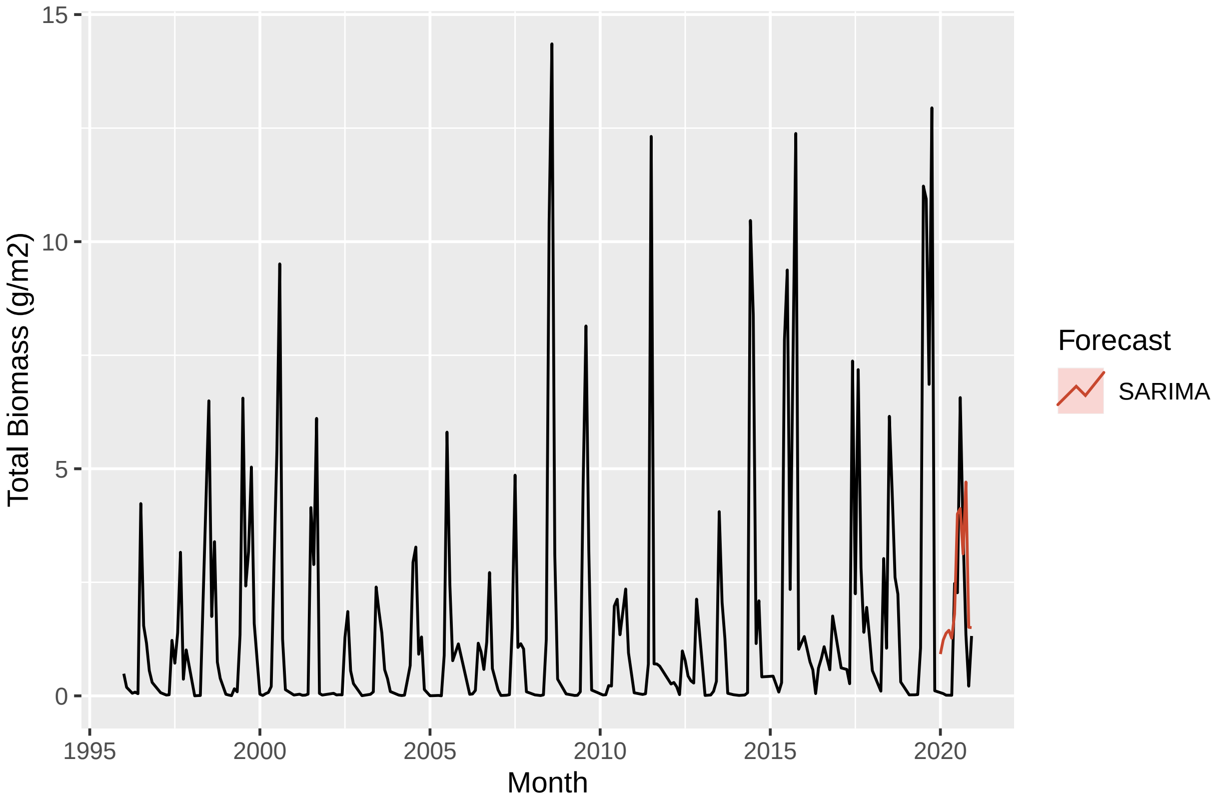
```r
autoplot(ts_biomass_data) +
  autolayer(MEAN_seas, PI=FALSE, series="Mean") +
```

```
  autolayer(SNAIVE_seas, PI=FALSE, series="Naïve") +
  autolayer(SARIMA_for,PI=FALSE, series="SARIMA") +
  autolayer(SSES_seas$forecast, series="SSES") +
  autolayer(SS_for,PI=FALSE,series="BSM") +
  xlab("Month") + ylab("Electricity Retail Price ($/kWh)") +
  guides(colour=guide_legend(title="Forecast"))
```



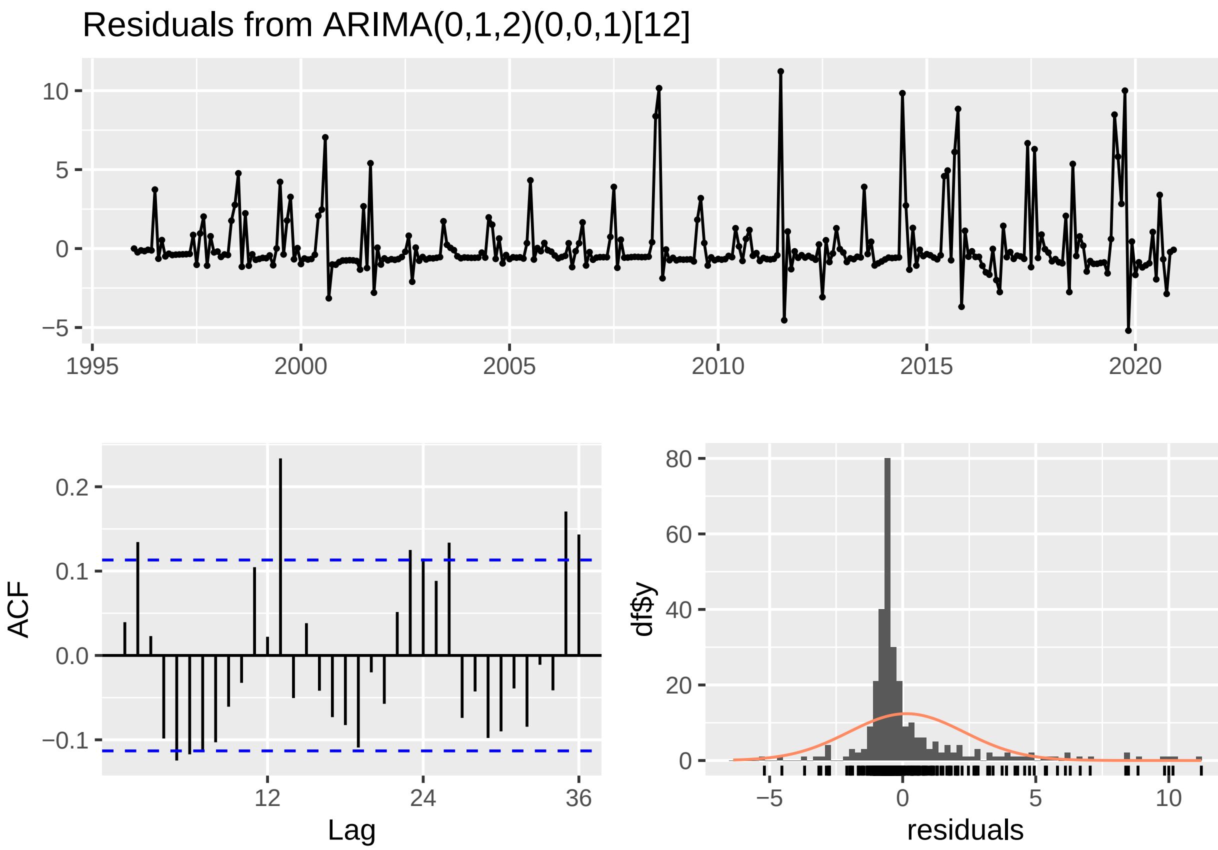
From the plot, we can see the SARIMA is the one with the closest forecasted value to the actual observation. If we only plot the actual value and the forecasted value using our outperforming model, SARIMA is shown as follows:

```
autoplot(ts_biomass_data) +

autolayer(SARIMA_for,PI=FALSE, series="SARIMA") +
  xlab("Month") + ylab("Total Biomass (g/m2)") +
  guides(colour=guide_legend(title="Forecast"))
```



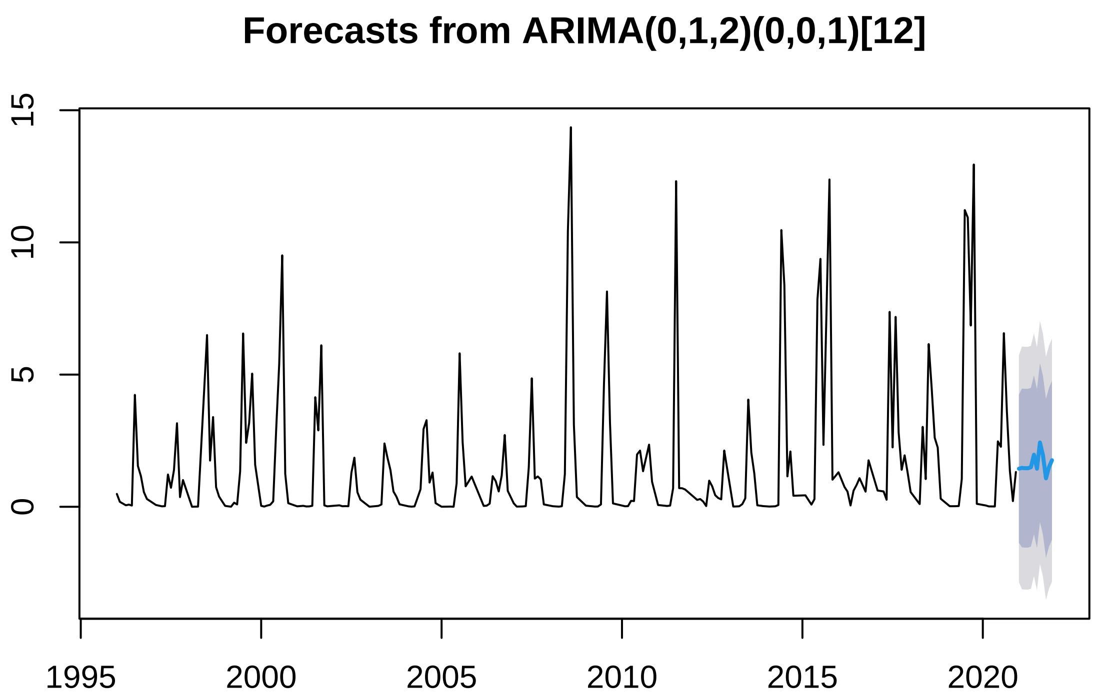
Then, we use our selected SARIMA model to conduct the forecast for the year 2021 using the full sample. The residual general shows random and most of the values are within the confidence interval of the ACF plot. The parameter of the fitted SARIMA model is ARIMA(0,1,2)(0,0,1). The forecast plot with a confidence interval of 95% for 2021 is shown as follows:

```
# Forecast

SARIMA_autofit_new <- auto.arima(ts_biomass_data)
checkresiduals(SARIMA_autofit_new)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,2)(0,0,1)[12]
## Q* = 68.766, df = 21, p-value = 5.534e-07
##
## Model df: 3.   Total lags used: 24
```

```
SARIMA_for_new <- forecast(SARIMA_autofit_new,h=12)
plot(SARIMA_for_new)
```



The predicted value for the year 2021 is presented as follows:

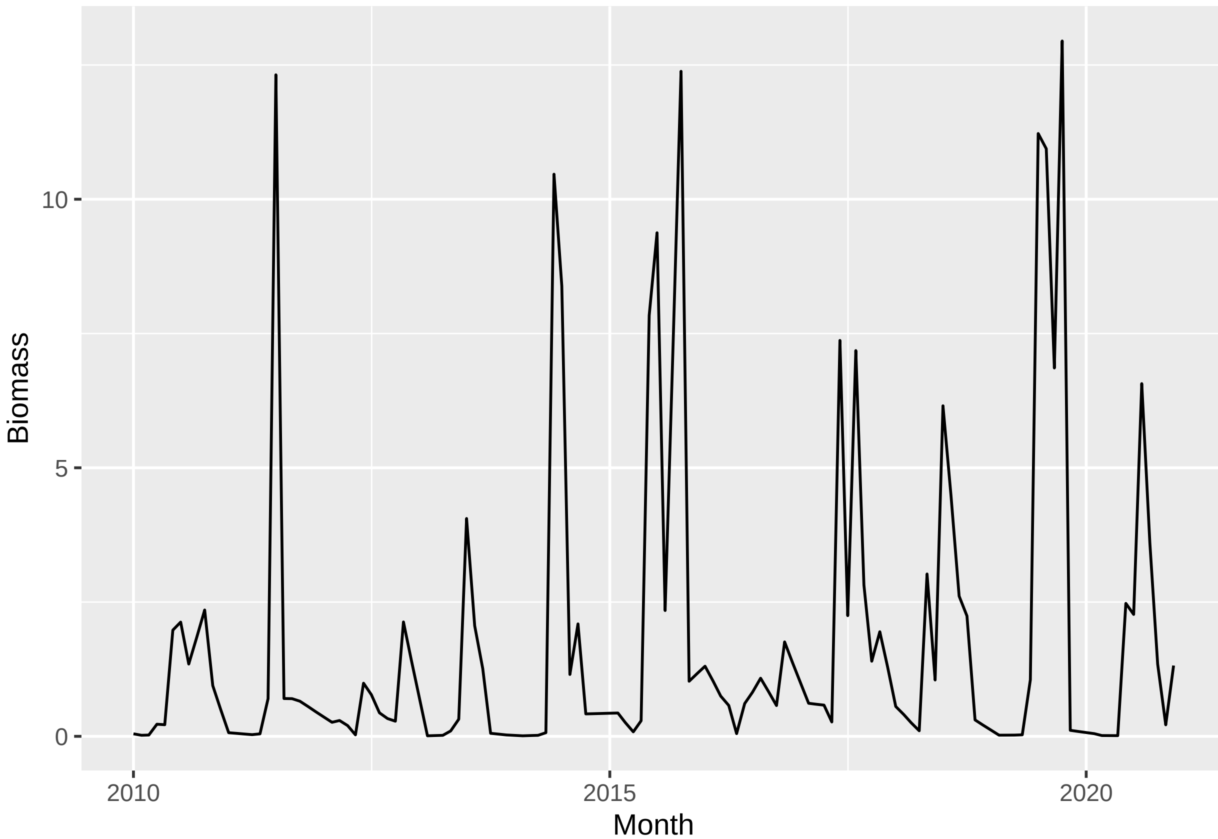
```
print(SARIMA_for_new$mean)
```

```
##           Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
## 2021 1.441261 1.470341 1.459956 1.457654 1.495943 1.971274 1.440000 2.432793
##           Sep      Oct      Nov      Dec
## 2021 1.948802 1.076378 1.499118 1.760980
```

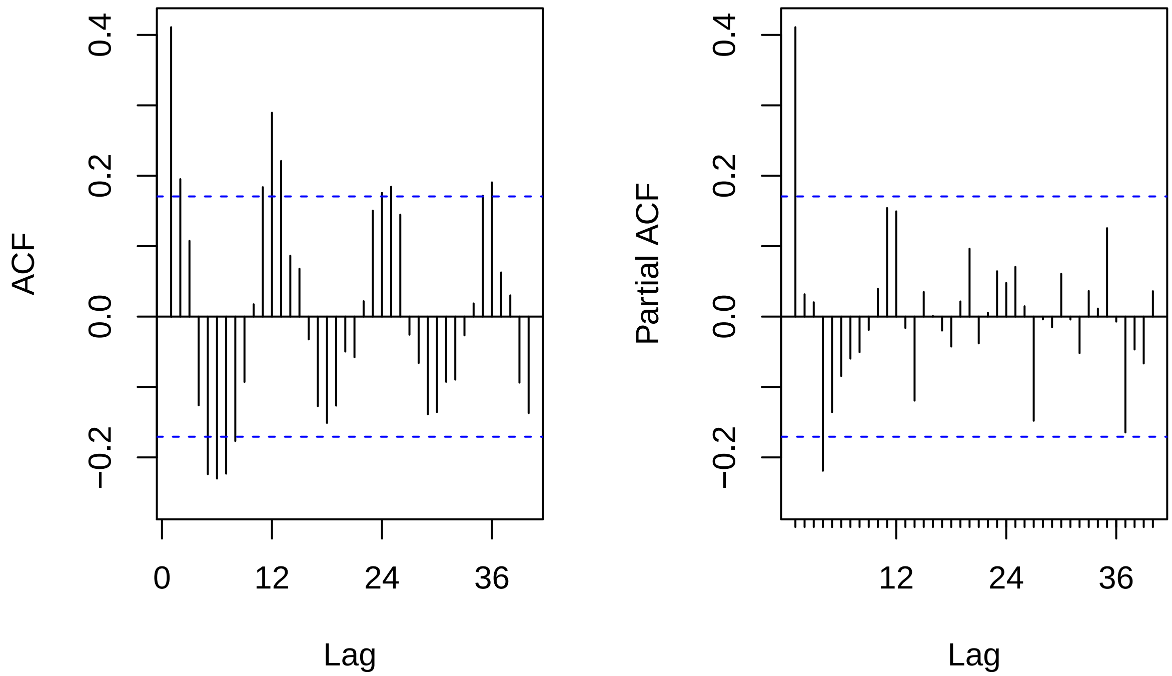
From the original plot of the biomass at the beginning, it seems to start to show a slightly different pattern since 2010. In order to make sure our forecast is not biased by some sudden changes in the previous old years. To be prudent, we further limit our sample to the recent ten years. We use samples from 2010 to 2020 to predict again. All the procedures are the same as the above full sample analysis only by changing the time span to the recent 10 years.

```
# Change the time span

# Transform to time series format

ts_biomass_data <- ts(
  biomass_data_frame[169:300,2],
  start=c(year(biomass_data_frame$Month[169]),month(biomass_data_frame$Month[169])),
  frequency=12)

ts_biomass <- ts(
  biomass_data_frame[169:288,2],
  start=c(year(biomass_data_frame$Month[169]),month(biomass_data_frame$Month[169])),
  frequency=12)


last_obs <- ts_biomass_data[121:132]

# Plot the time series, ACF, and PACF
```

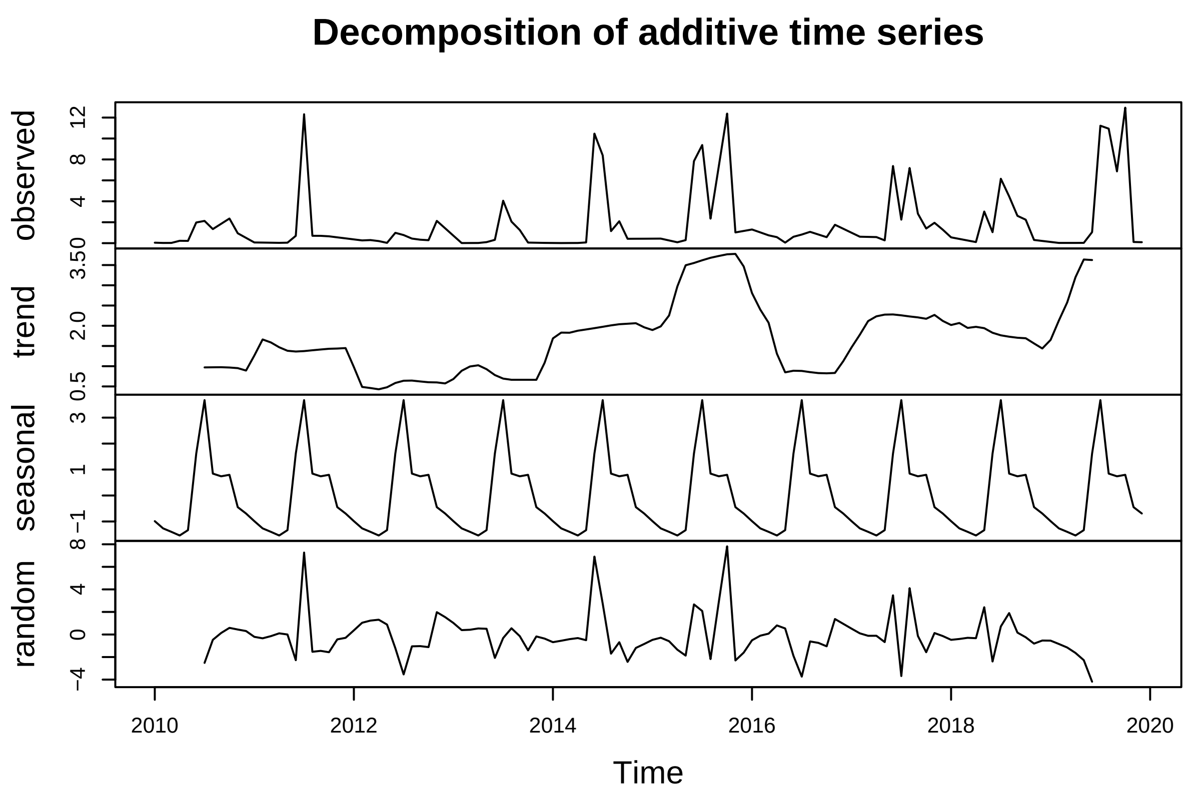
```
TS_Plot <- ggplot(biomass_data_frame[169:300,], aes(x=Month, y=Biomass)) +
      geom_line()

plot(TS_Plot)
```




```
#ACF and PACF plots
par(mfrow=c(1,2))
ACF_Plot <- Acf(ts_biomass_data, lag = 40, plot = TRUE,main="")
PACF_Plot <- Pacf(ts_biomass_data, lag = 40, plot = TRUE,main="")
```

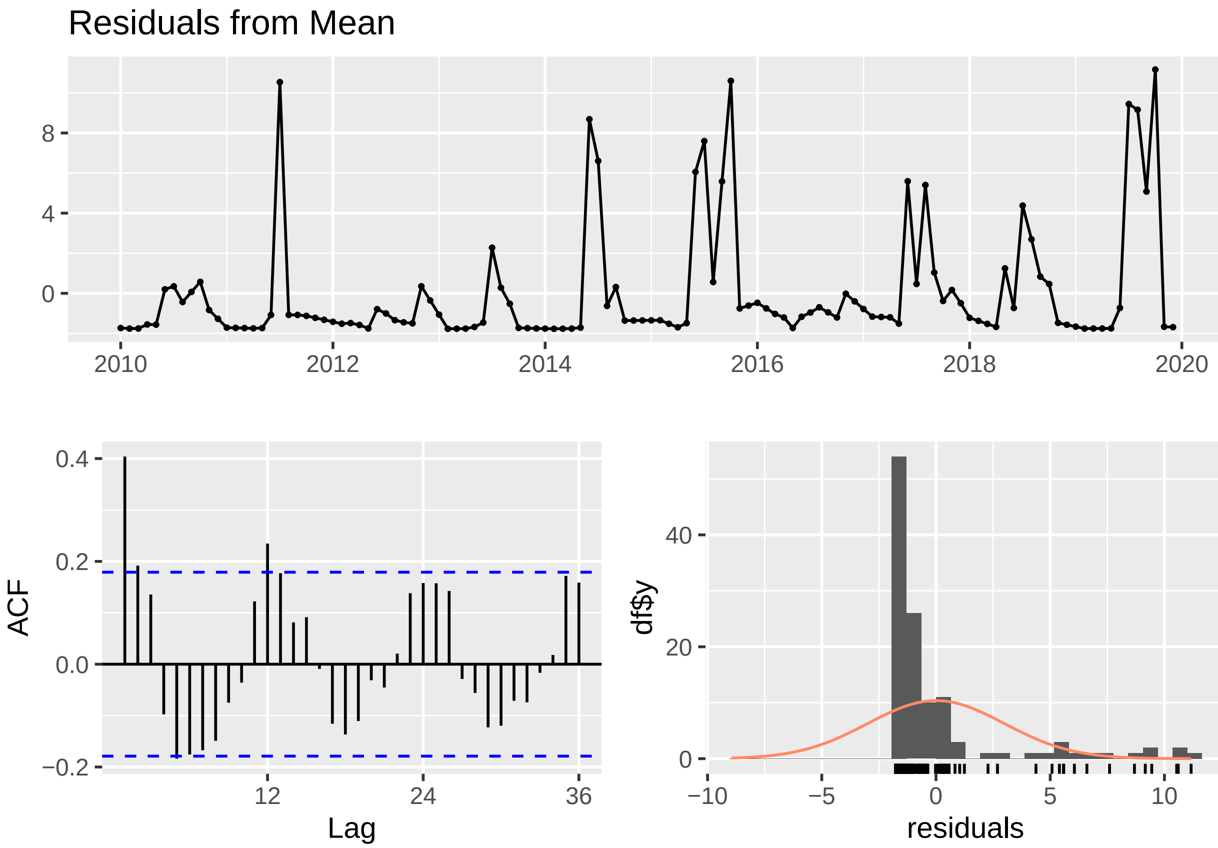



```r
#Plot ts decompose
decompose_biomass_data <- decompose(ts_biomass,"additive")

plot(decompose_biomass_data)
```



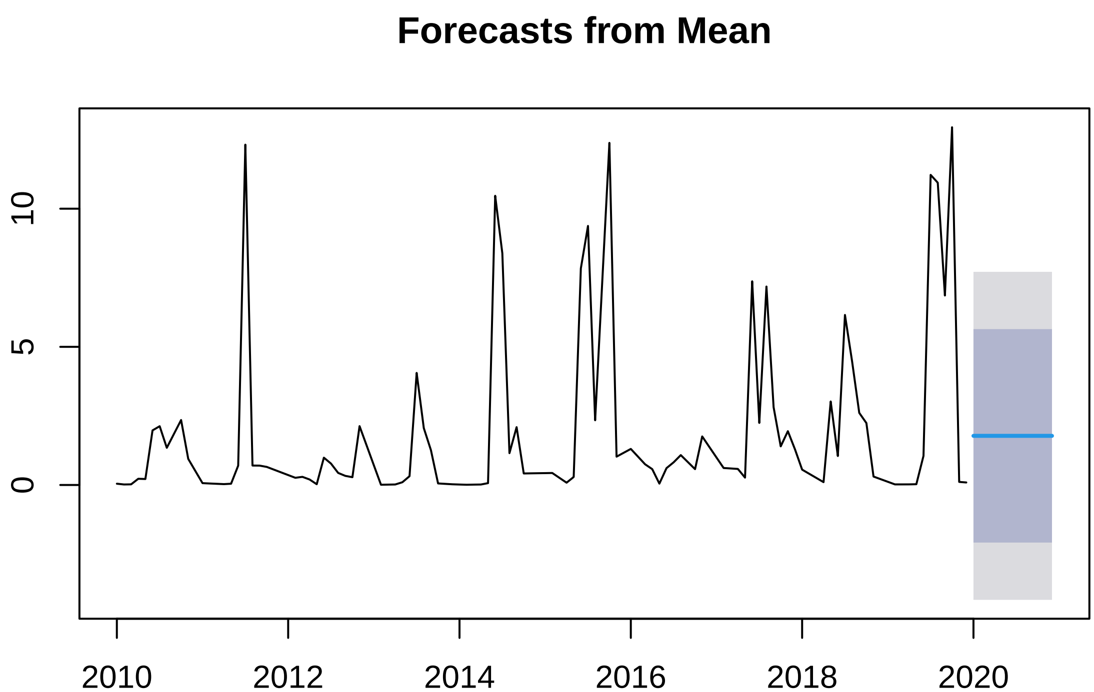
The ACF and PACF plot pattern show similar to previous We still use those five models to fit the data. The fitting process and forecast plot is shown as follows:

```r
# Model 1: Arithmetic mean
# The meanf() has no holdout option
MEAN_seas <- meanf(y = ts_biomass, h = 12)
checkresiduals(MEAN_seas)
```

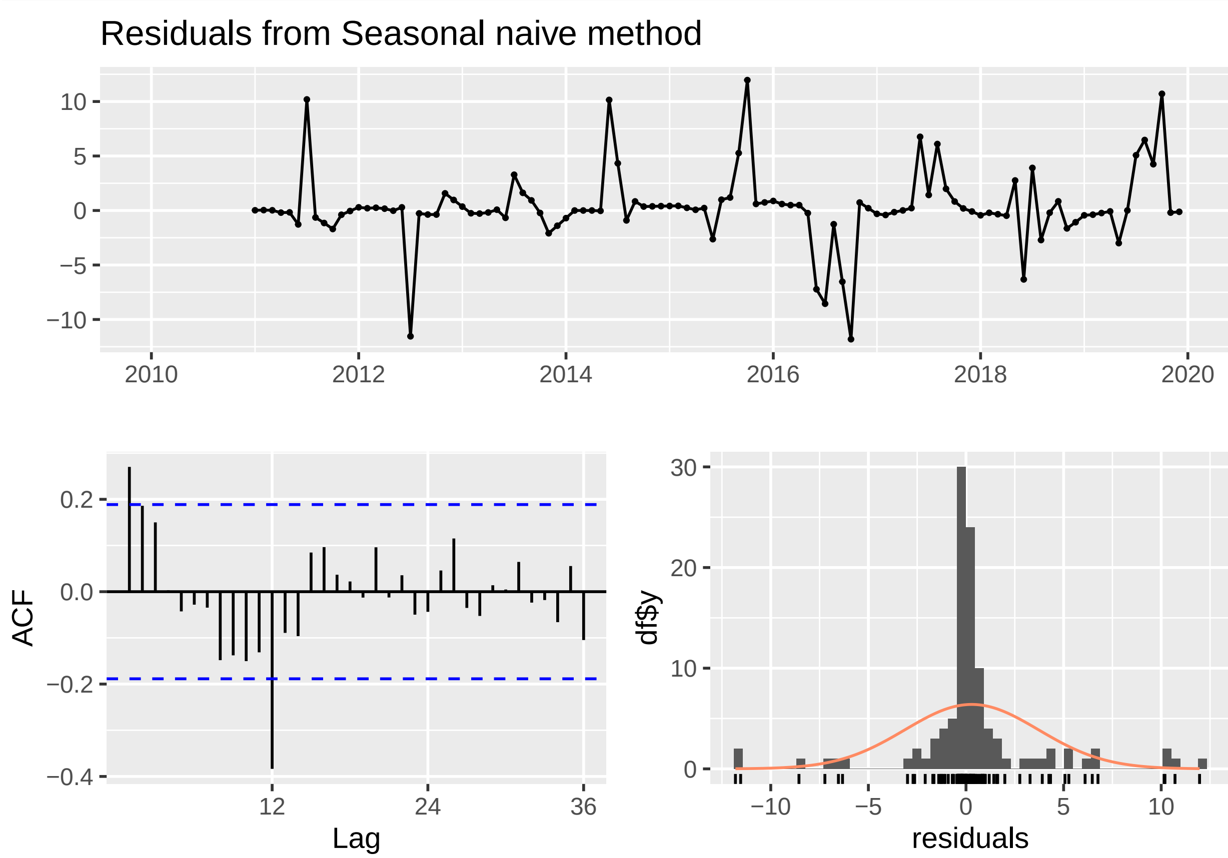



```
##
##  Ljung-Box test
##
## data:  Residuals from Mean
## Q* = 73.247, df = 23, p-value = 3.795e-07
##
## Model df: 1.   Total lags used: 24
```

```r
plot(MEAN_seas)
```



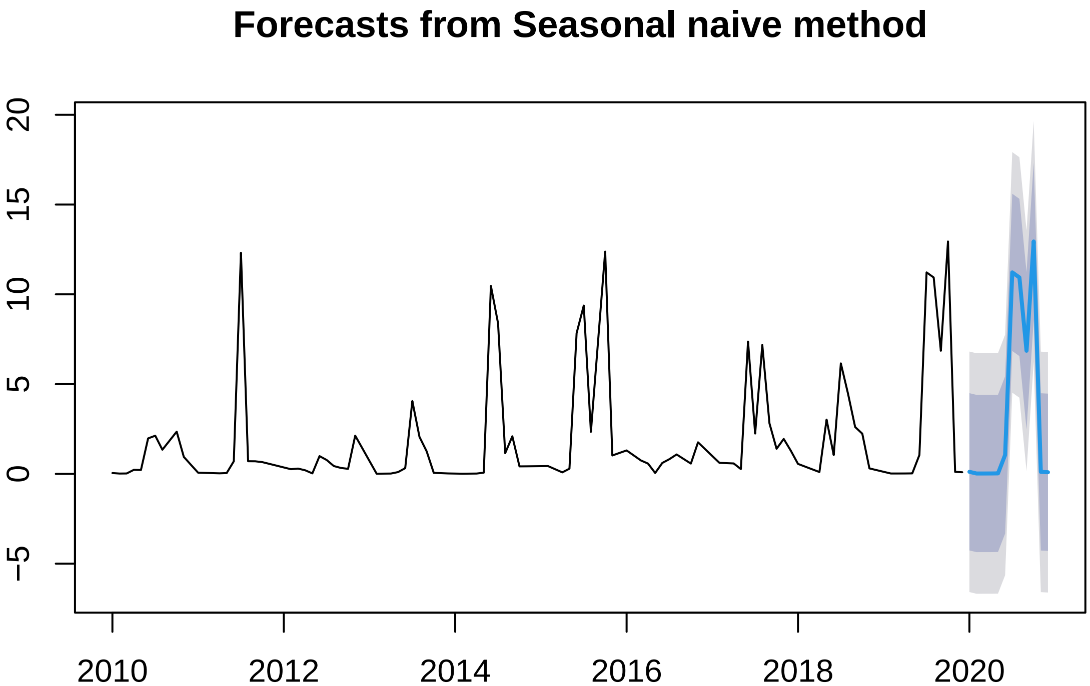

```r
# Model 2: Seasonal naive
SNAIVE_seas <- snaive(ts_biomass, h=12, holdout=FALSE)
checkresiduals(SNAIVE_seas)
```



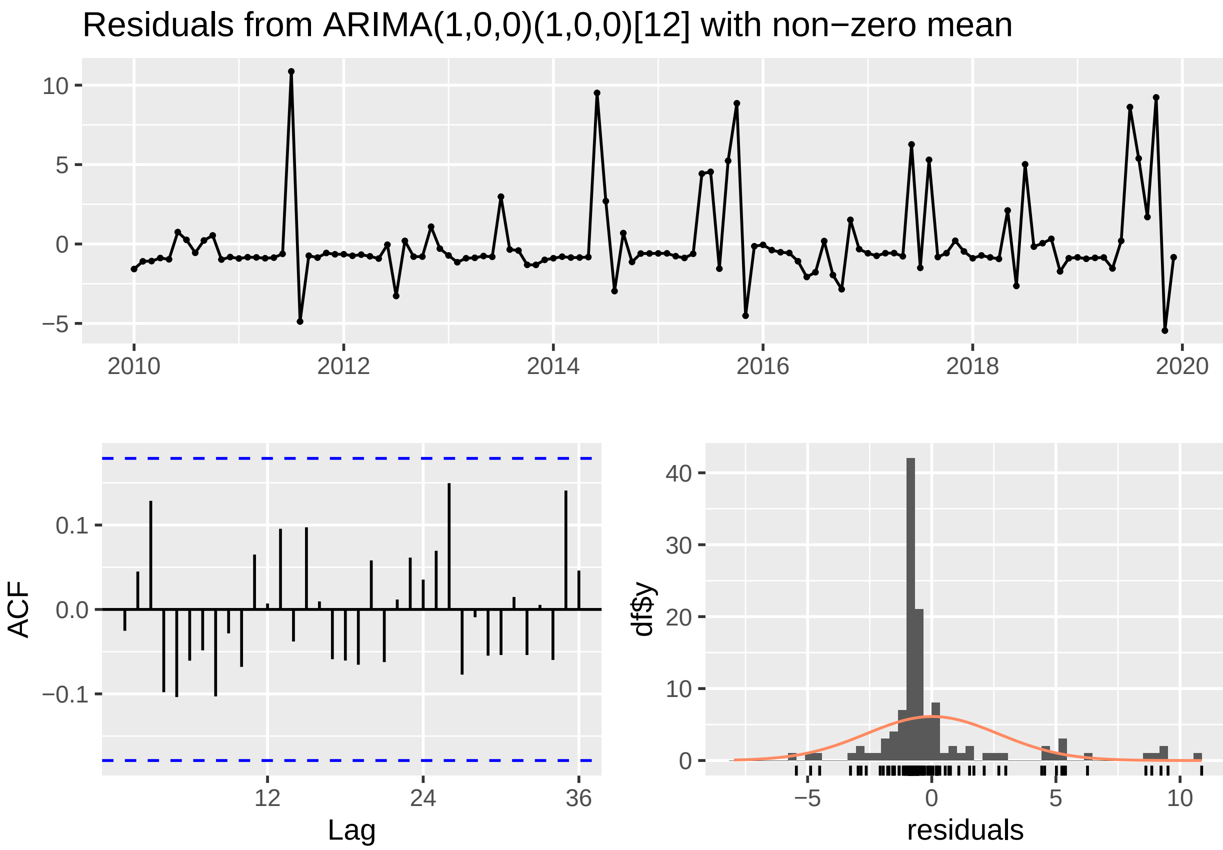

```
##
##  Ljung-Box test
```

```
##
## data:  Residuals from Seasonal naive method
## Q* = 49.511, df = 24, p-value = 0.001633
##
## Model df: 0.   Total lags used: 24
```

```
plot(SNAIVE_seas)
```

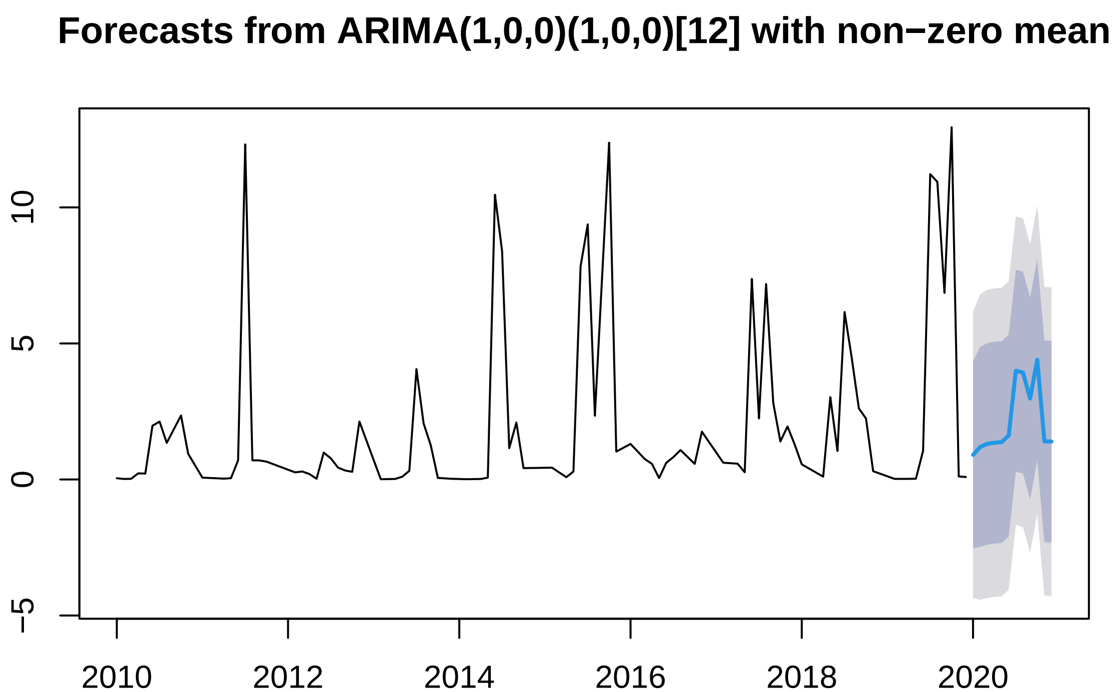



```
# Model 3: SARIMA

SARIMA_autofit <- auto.arima(ts_biomass)
checkresiduals(SARIMA_autofit)
```

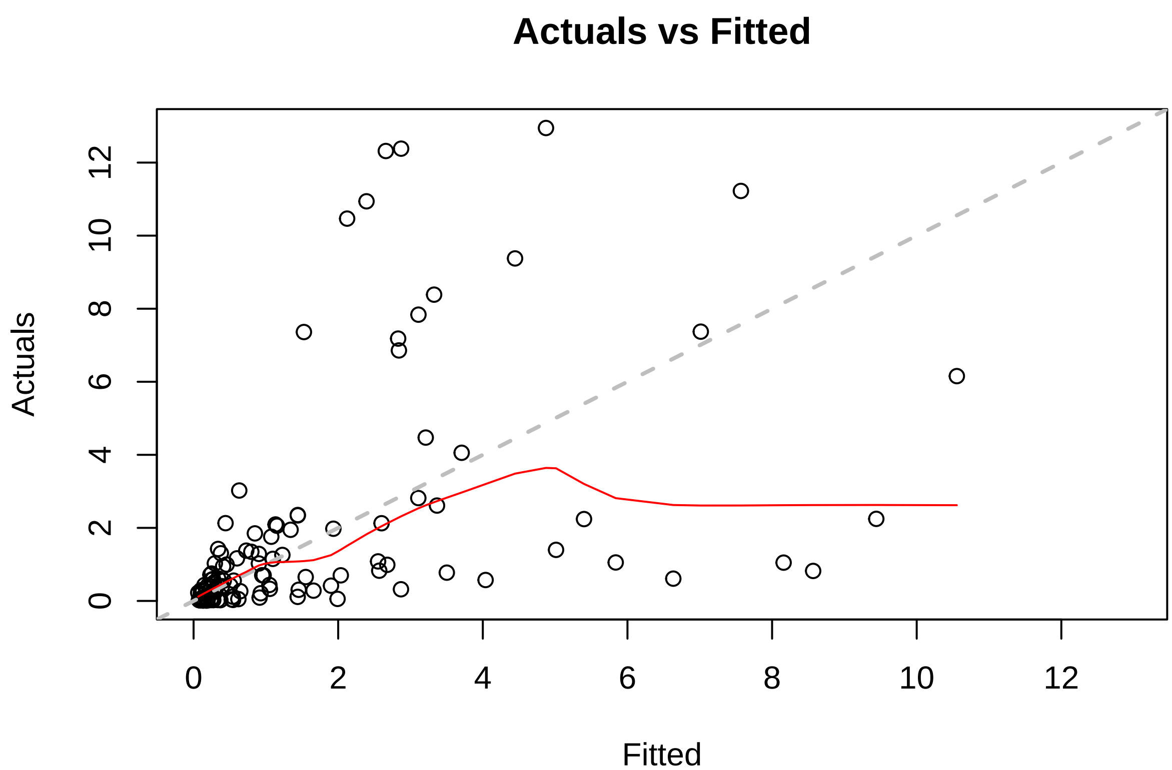


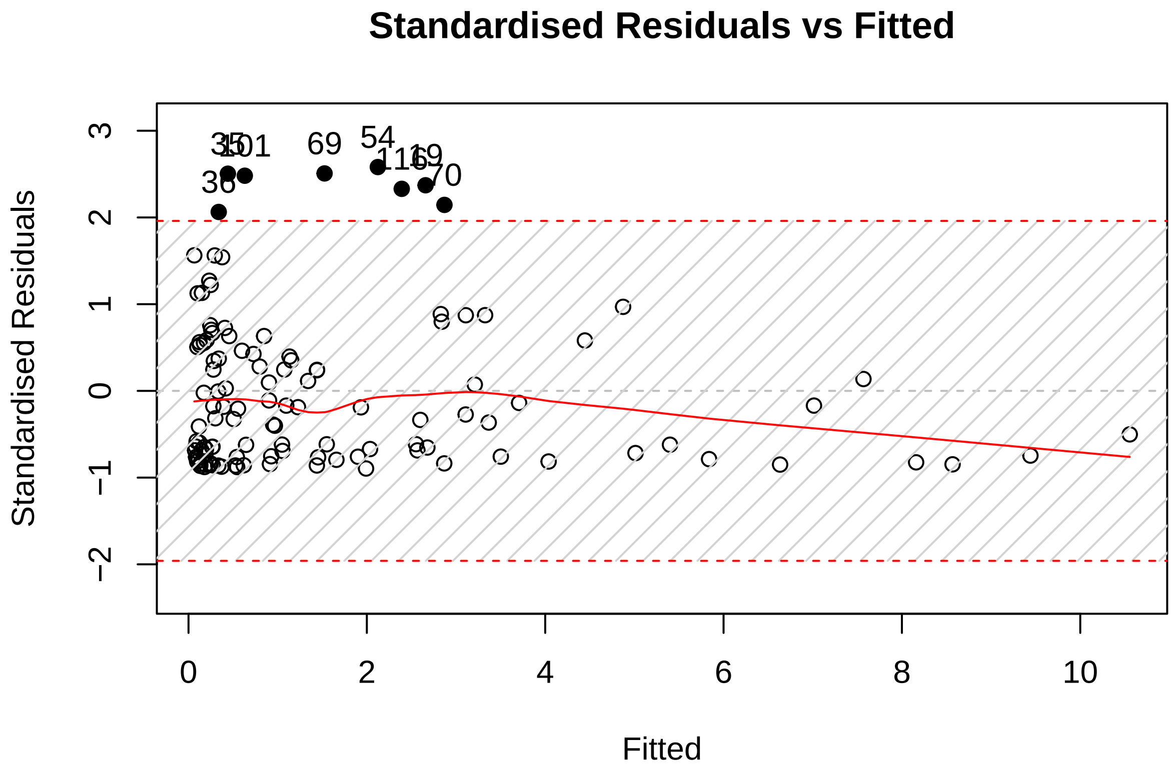

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,0)(1,0,0)[12] with non-zero mean
## Q* = 14.711, df = 22, p-value = 0.8743
##
## Model df: 2.   Total lags used: 24
```

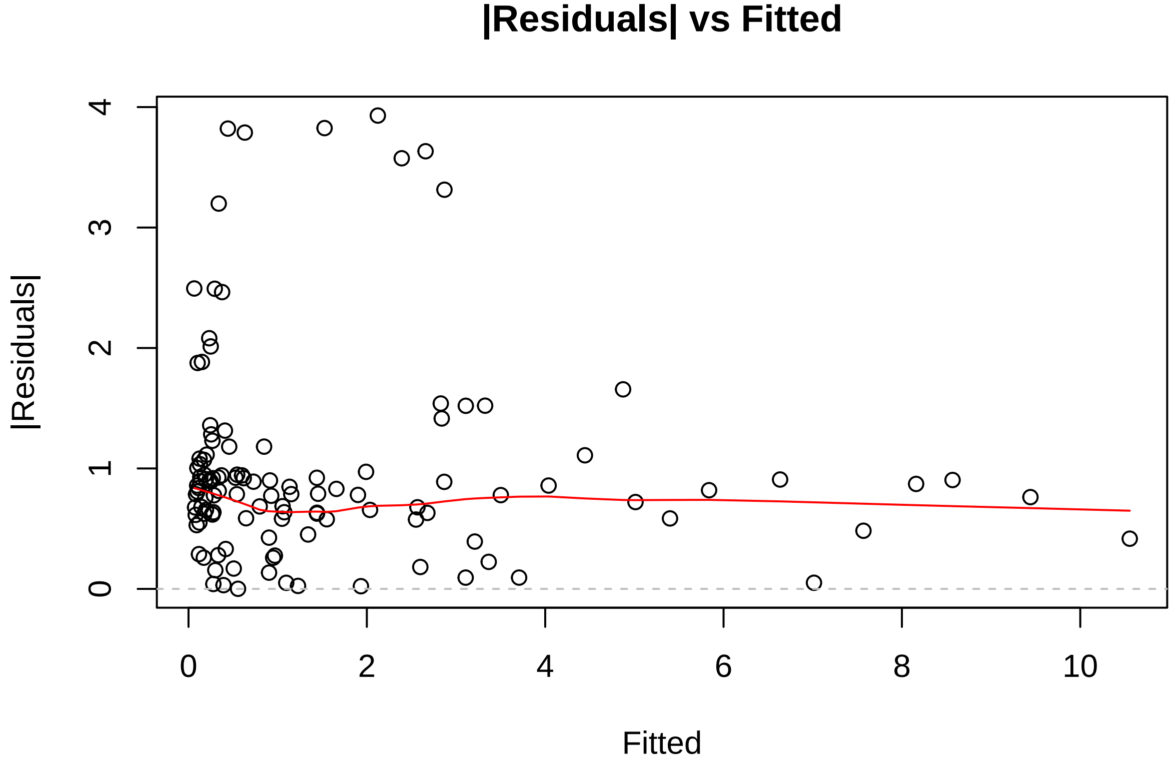
```
#Generating forecasts
#remember auto.arima does not call the forecast() internally so we need one more step
SARIMA_for <- forecast(SARIMA_autofit,h=12)
plot(SARIMA_for)
```

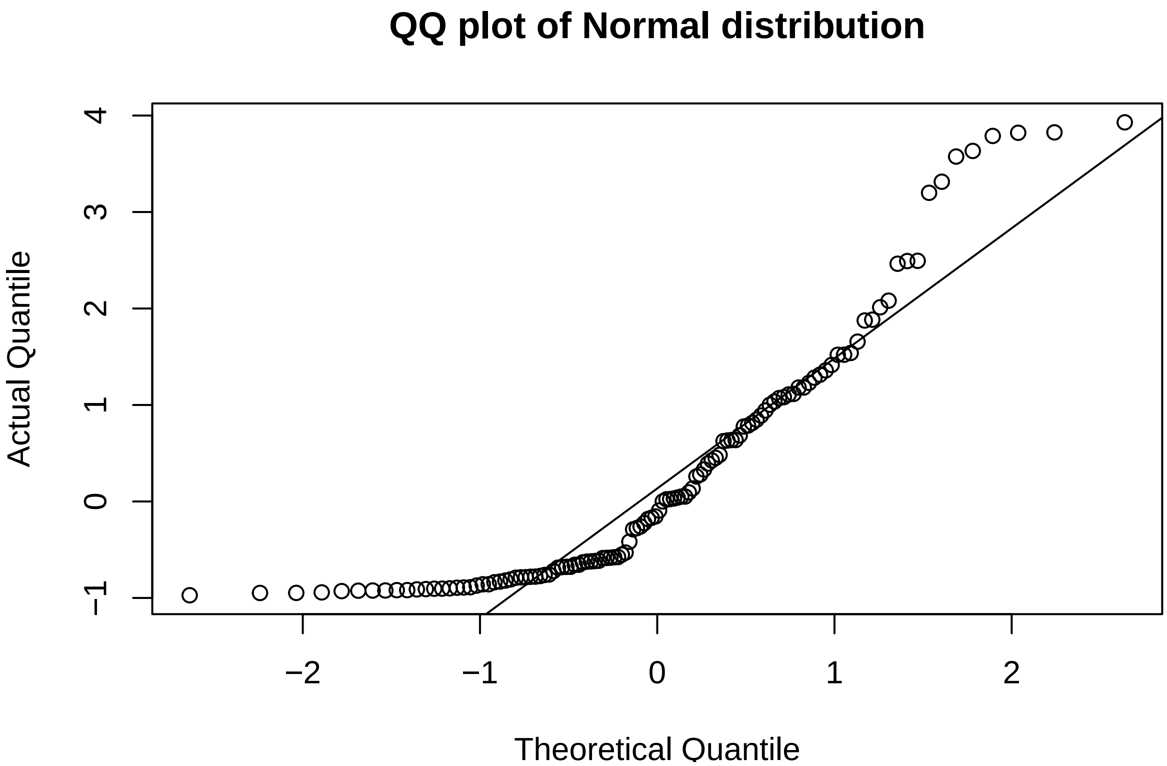



```
# Model 4: SS Exponential smoothing
SSES_seas <- es(ts_biomass,model="ZZZ",h=12,holdout=FALSE)
plot(SSES_seas)
```



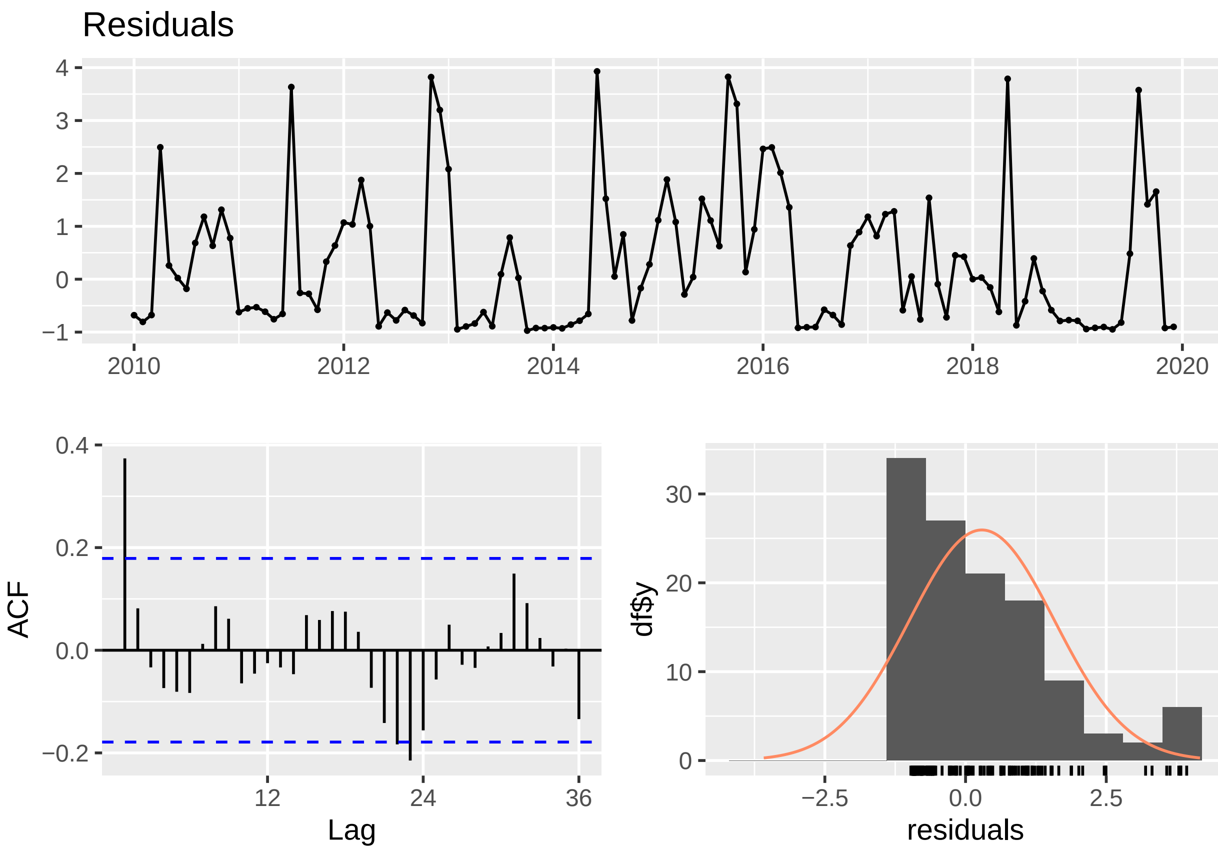

Standardised Residuals vs Fitted
Standardised Residuals
Fitted
3
2
1
0
−1
−2
0
2
4
6
8
10
35
101
36
69
54
116
19
70


|Residuals| vs Fitted
|Residuals|
Fitted
4
3
2
1
0
0
2
4
6
8
10



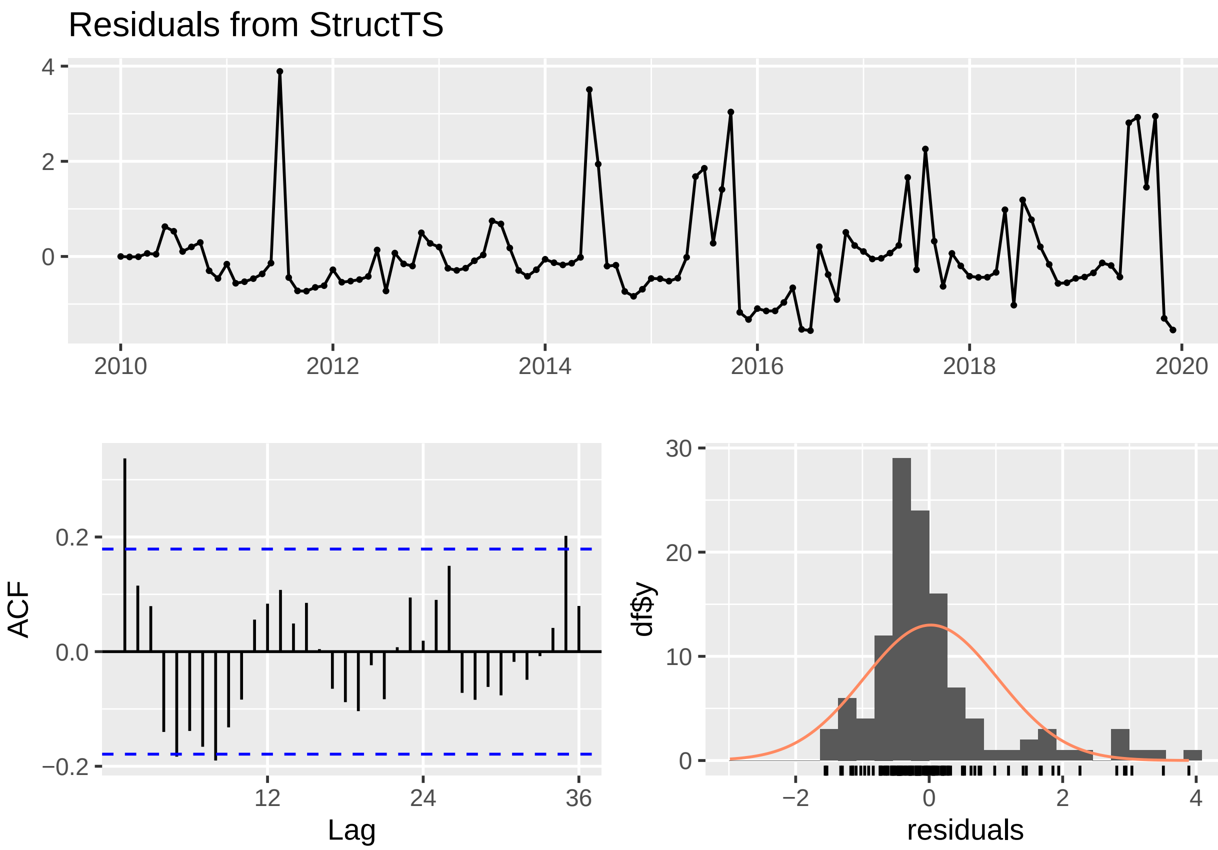

```
checkresiduals(SSES_seas)
```

```
## Warning in modeldf.default(object): Could not find appropriate degrees of
## freedom for this model.
```




```
# Model 5: SS with StructTS()

SS_seas <- StructTS(ts_biomass,
                    type="BSM",fixed=c(0,0.001,0.3,NA))    #this function has convergence issues
checkresiduals(SS_seas)
```

```
## Warning in modeldf.default(object): Could not find appropriate degrees of
## freedom for this model.
```




```
#Generating forecasts
# StructTS() does not call the forecast() internally so we need one more step
SS_for <- forecast(SS_seas,h=12)
plot(SS_for)
```

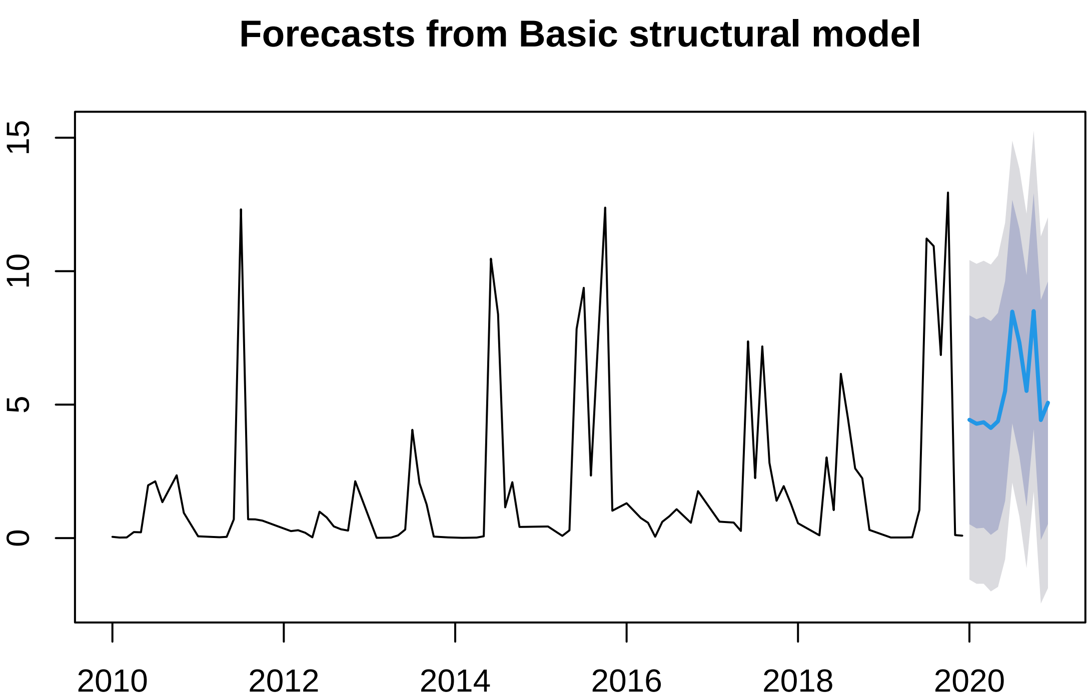


Table 2: Forecast Accuracy for Seasonal Data

	ME	RMSE	MAE	MPE	MAPE
MEAN	-0.28323	1.93838	1.58284	-3817.5199	3846.3960
SNAIVE	-2.12360	4.54636	2.58677	-122.4177	164.9659
SARIMA	-0.65751	1.55783	1.34461	-2824.6163	2839.9915
SSES	-1.10662	2.85708	1.89745	-919.7008	937.4871
BSM	-4.03626	4.34690	4.03626	-9410.6202	9410.6202

```r
#Model 1: Arithmetic mean
MEAN_scores <- accuracy(MEAN_seas$mean,last_obs)  #store the performance metrics

#Model 2: Seasonal naive
SNAIVE_scores <- accuracy(SNAIVE_seas$mean,last_obs)

# Model 3:  SARIMA
SARIMA_scores <- accuracy(SARIMA_for$mean,last_obs)

# Model 4:  SSES
SSES_scores <- accuracy(SSES_seas$forecast,last_obs)

# Model 5:  BSM
SS_scores <- accuracy(SS_for$mean,last_obs)


#create data frame
seas_scores <- as.data.frame(rbind(MEAN_scores, SNAIVE_scores, SARIMA_scores,SSES_scores,SS_scores))
row.names(seas_scores) <- c("MEAN", "SNAIVE","SARIMA","SSES","BSM")

#choose model with lowest RMSE
best_model_index <- which.min(seas_scores[,"RMSE"])
cat("The best model by RMSE is:", row.names(seas_scores[best_model_index,]))
```

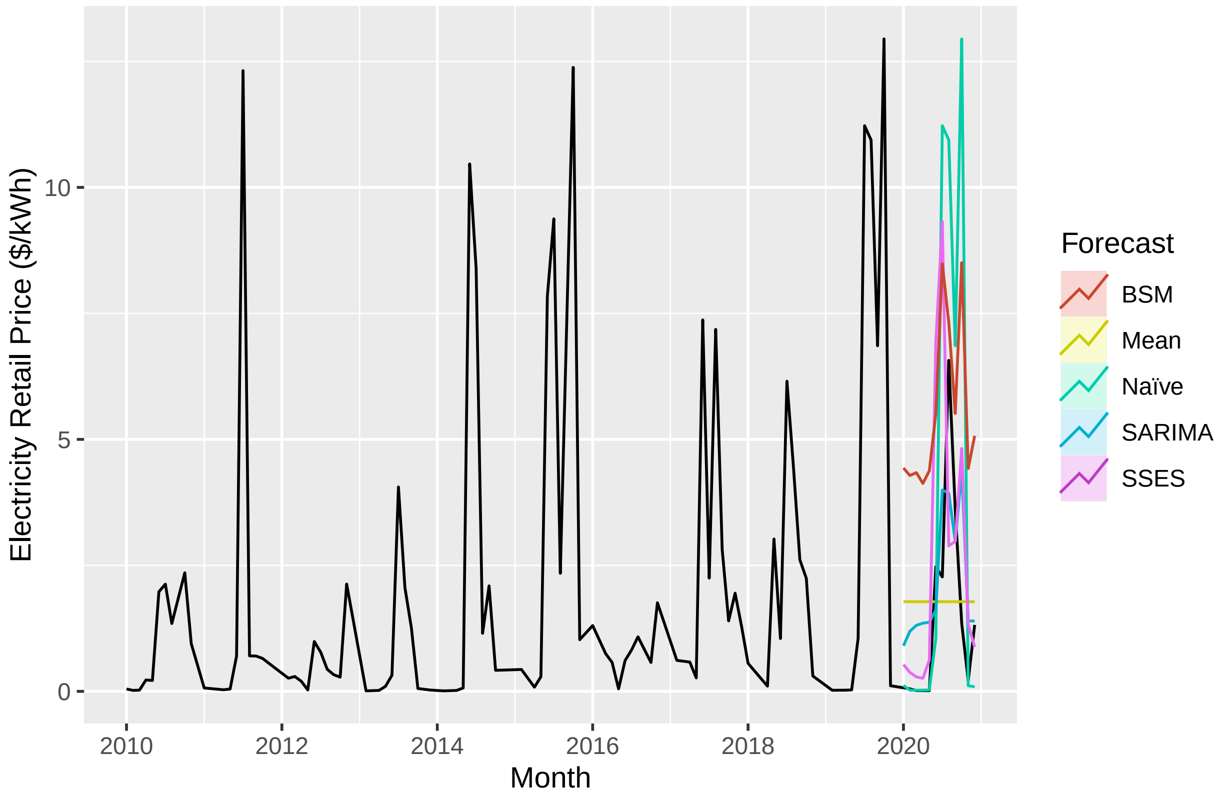
```
## The best model by RMSE is: SARIMA
```

```r
kbl(seas_scores,
      caption = "Forecast Accuracy for Seasonal Data",
      digits = array(5,ncol(seas_scores))) %>%
  kable_styling(full_width = FALSE, position = "center") %>%
  #highlight model with lowest RMSE
  kable_styling(latex_options="striped", stripe_index = which.min(seas_scores[,"RMSE"]))
```

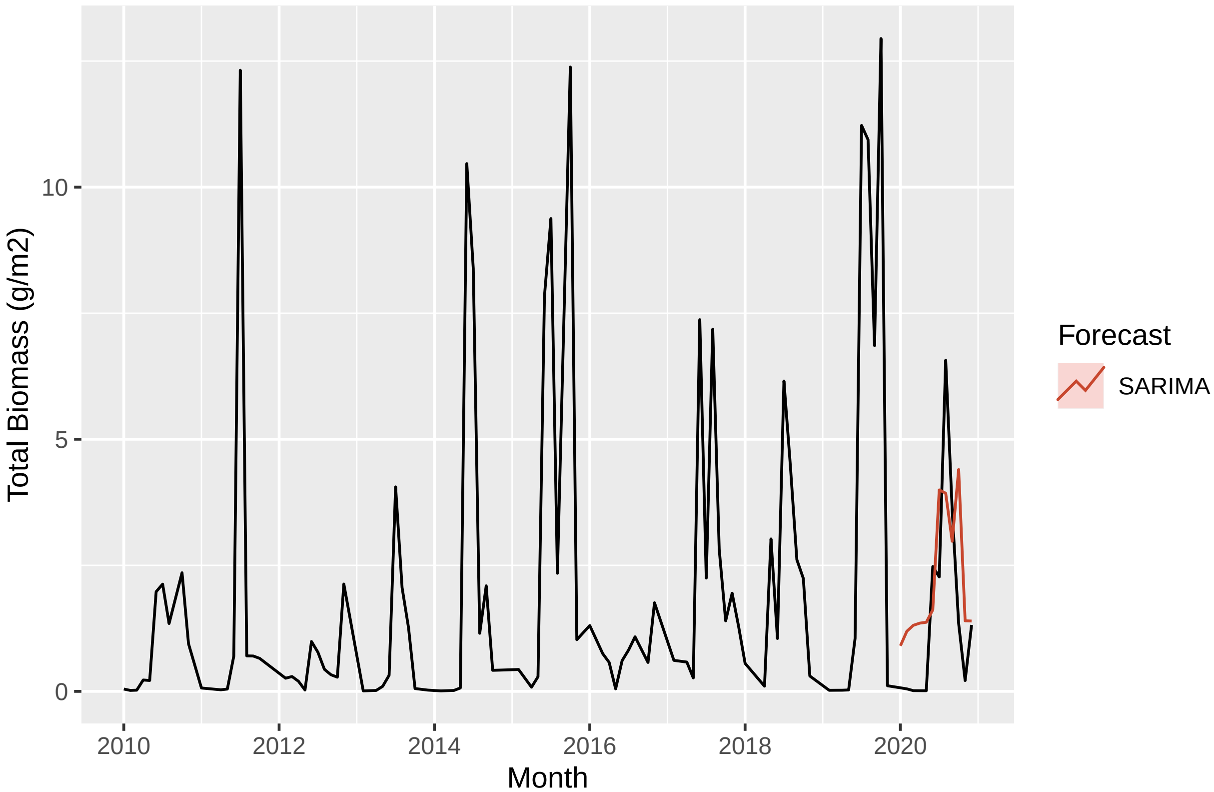
Based on the compare performance metrics in Table 6, the SARIMA still shows the best fit with the lowest RMSE value which is consistent with the exercise using the full sample. The parameter for the SARIMA model is ARIMA(1,0,0)(1,0,0). To visually compare the result of forecasts using different models, we further jointly plot the forecast generated using five models and compare them with the actual observed value, and again also isolate the SARIMA forecast plot alone with the actual values. The plots are shown as follows:

```r
autoplot(ts_biomass_data) +
  autolayer(MEAN_seas, PI=FALSE, series="Mean") +
  autolayer(SNAIVE_seas, PI=FALSE, series="Naïve") +
  autolayer(SARIMA_for,PI=FALSE, series="SARIMA") +
  autolayer(SSES_seas$forecast, series="SSES") +
  autolayer(SS_for,PI=FALSE,series="BSM") +
```

```
  xlab("Month") + ylab("Electricity Retail Price ($/kWh)") +
  guides(colour=guide_legend(title="Forecast"))
```

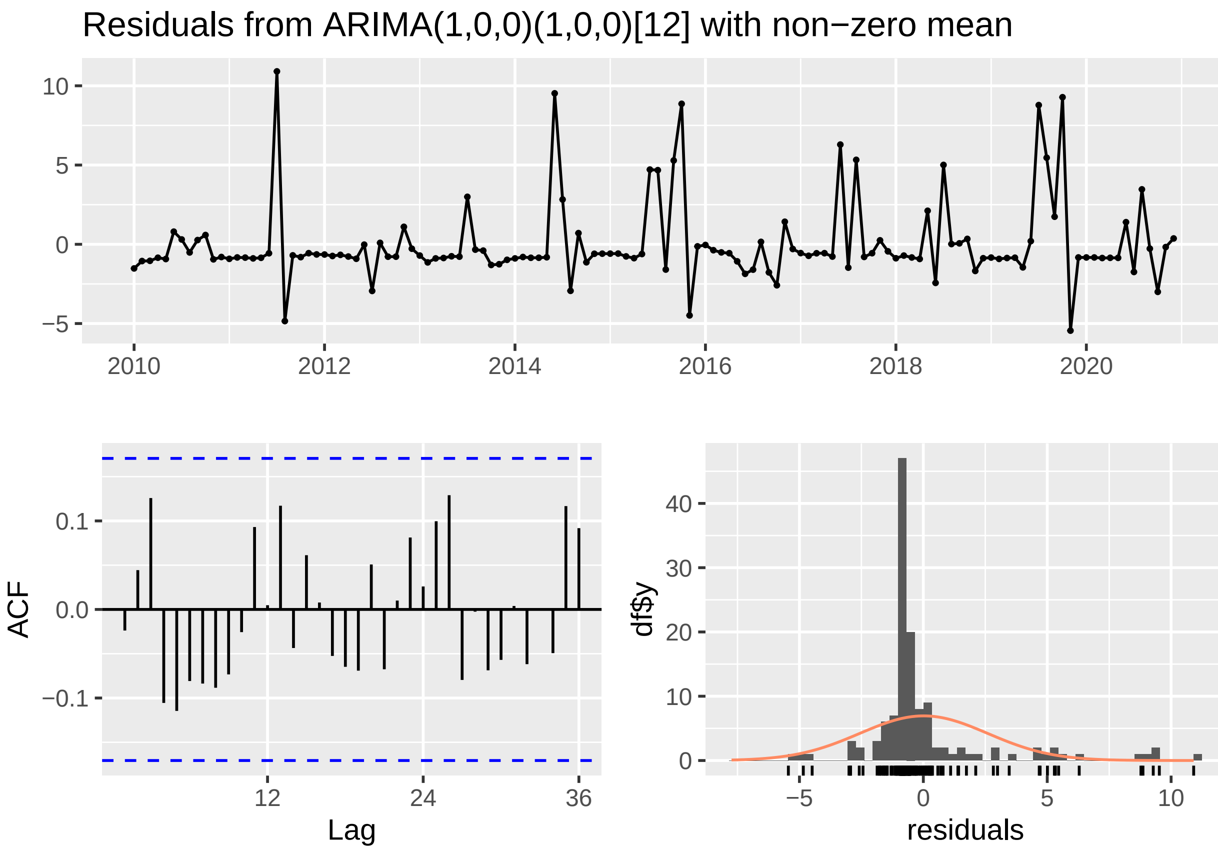



```
autoplot(ts_biomass_data) +

autolayer(SARIMA_for,PI=FALSE, series="SARIMA") +
  xlab("Month") + ylab("Total Biomass (g/m2)") +
  guides(colour=guide_legend(title="Forecast"))
```



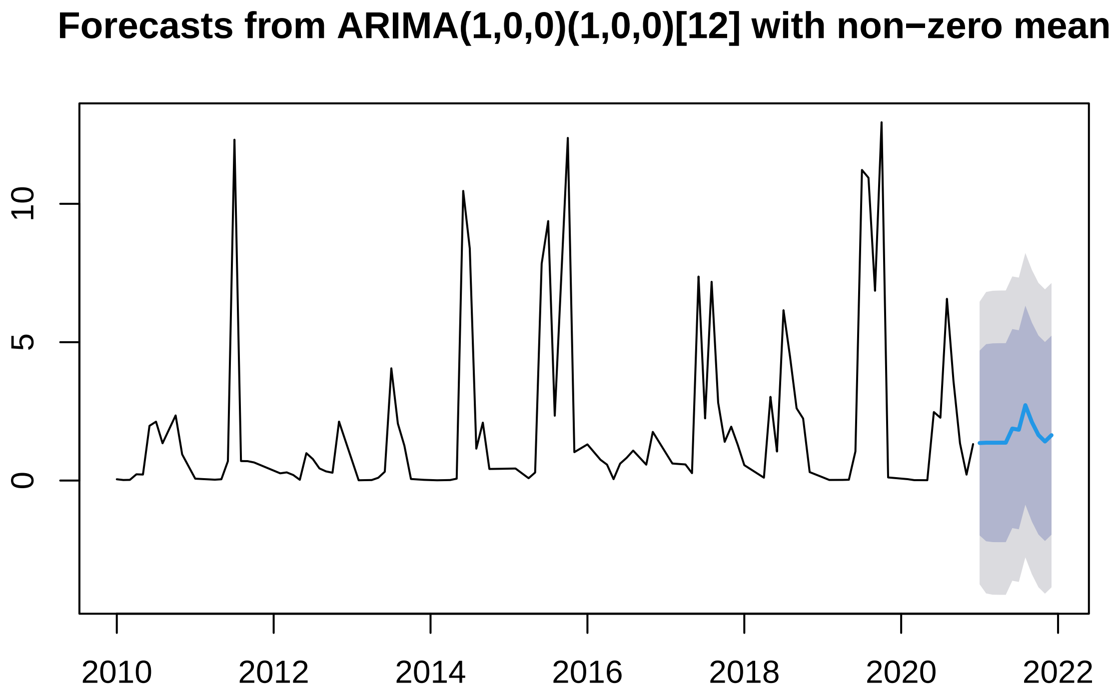
Using the SARIMA model to conduct the forecast for the year 2021 using the full sample. The residual general shows random and most of the values are within the confidence interval of the ACF plot. Then, the forecast plot with a confidence interval of 95% for 2021 is shown as follows:

```
# Forecast

SARIMA_autofit_new <- auto.arima(ts_biomass_data)
checkresiduals(SARIMA_autofit_new)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,0)(1,0,0)[12] with non-zero mean
## Q* = 18.099, df = 22, p-value = 0.7001
##
## Model df: 2.   Total lags used: 24
```

```
SARIMA_for_new <- forecast(SARIMA_autofit_new,h=12)
plot(SARIMA_for_new)
```



The predicted value for the year 2021 using recent ten-year data from 2010 is presented as follows:

```
print(SARIMA_for_new$mean)
```

```
##           Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
## 2021 1.355864 1.367419 1.366069 1.368040 1.368640 1.877550 1.835275 2.722854
##           Sep      Oct      Nov      Dec
## 2021 2.110541 1.644246 1.410719 1.638655
```

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.