US Retail Sales Time Series Forecast

```
library('fpp2')
```

```
## Warning: package 'fpp2' was built under R version 4.1.3
```

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

```
## -- Attaching packages --------------------------------------------- fpp2 2.4 --
```

```
## v ggplot2   3.3.5      v fma       2.4
## v forecast  8.17.0     v expsmooth 2.3
```

```
## Warning: package 'forecast' was built under R version 4.1.3
```

```
## Warning: package 'fma' was built under R version 4.1.3
```

```
## Warning: package 'expsmooth' was built under R version 4.1.3
```

```
##
```

We will be using real data i.e., deflated data so that we can isolate real growth. This will prevent inflation from distorting the picture for us.

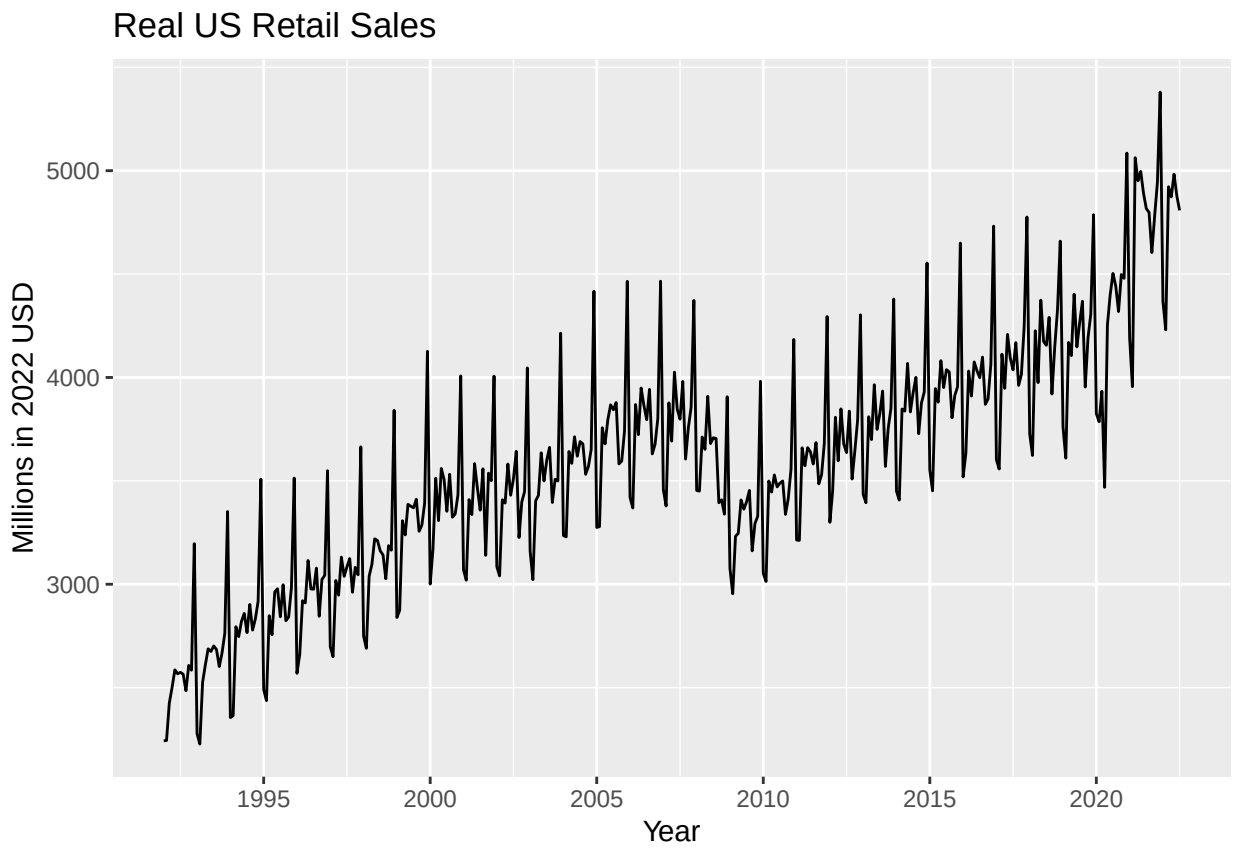
```
sales = read.csv('RSXFSN.csv')
cpi = read.csv('USACPIALLMINMEI.csv')
real = sales[,2]/cpi[,2]
data = ts(real,start=c(1992,1),frequency=12)
data
```

```
##           Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
## 1992 2242.874 2244.377 2424.413 2500.572 2585.979 2567.104 2574.055 2564.897
## 1993 2277.424 2227.099 2525.733 2610.726 2687.710 2674.878 2701.567 2684.267
## 1994 2355.190 2364.224 2794.643 2746.358 2820.534 2858.761 2766.231 2902.186
## 1995 2491.659 2437.019 2847.805 2756.122 2963.008 2978.006 2842.790 2997.513
## 1996 2569.301 2661.522 2920.485 2909.804 3114.776 2977.339 2976.148 3077.906
## 1997 2697.510 2649.779 3018.263 2947.459 3131.348 3038.357 3084.263 3124.143
## 1998 2749.236 2690.297 3039.034 3095.982 3219.748 3211.609 3161.490 3141.926
## 1999 2839.155 2873.755 3307.177 3238.790 3386.434 3377.421 3369.594 3410.974
## 2000 3000.750 3169.816 3512.573 3307.598 3559.829 3506.669 3352.872 3531.759
## 2001 3069.864 3019.621 3409.157 3337.023 3583.372 3466.240 3358.351 3558.059
## 2002 3085.450 3040.483 3408.551 3392.546 3581.382 3430.554 3511.102 3642.690
## 2003 3160.283 3022.297 3403.832 3430.041 3635.750 3499.671 3600.016 3661.977
```

```
## 2004 3235.538 3229.248 3641.816 3584.567 3713.220 3619.151 3690.094 3678.867
## 2005 3274.601 3278.694 3756.762 3678.928 3795.344 3867.520 3843.787 3878.788
## 2006 3420.216 3368.779 3869.059 3723.178 3949.031 3864.744 3795.824 3942.398
## 2007 3457.599 3378.411 3877.149 3692.314 4025.729 3847.169 3798.378 3980.637
## 2008 3453.702 3450.363 3712.032 3651.988 3908.966 3680.563 3708.393 3704.701
## 2009 3075.744 2954.042 3232.164 3246.046 3407.818 3363.069 3399.194 3453.721
## 2010 3052.245 3013.450 3499.260 3445.737 3528.672 3470.823 3488.921 3499.353
## 2011 3213.989 3212.076 3659.743 3572.696 3660.493 3639.647 3581.462 3685.055
## 2012 3299.510 3450.891 3807.510 3597.145 3848.222 3677.813 3636.611 3837.330
## 2013 3434.575 3394.331 3809.930 3699.663 3964.928 3748.723 3824.948 3934.393
## 2014 3449.897 3406.951 3847.422 3837.270 4067.726 3833.374 3926.573 4000.407
## 2015 3552.654 3451.985 3946.051 3880.594 4081.251 3951.187 4037.653 4027.129
## 2016 3519.220 3633.829 4030.997 3910.408 4075.135 4032.636 3998.729 4098.625
## 2017 3602.324 3557.079 4112.316 3947.375 4208.068 4093.561 4036.743 4168.512
## 2018 3726.797 3622.950 4225.730 3975.185 4373.822 4173.675 4155.524 4290.579
## 2019 3760.443 3609.761 4169.303 4106.014 4402.268 4147.555 4264.590 4368.728
## 2020 3824.718 3786.025 3932.775 3468.898 4254.693 4400.835 4503.022 4442.133
## 2021 4191.195 3955.477 5062.564 4951.269 4996.300 4891.145 4817.079 4798.284
## 2022 4368.141 4230.477 4922.276 4873.216 4982.647 4878.809 4807.923
##           Sep      Oct      Nov      Dec
## 1992 2485.206 2607.304 2584.220 3196.088
## 1993 2602.091 2667.735 2763.248 3351.628
## 1994 2778.334 2832.187 2917.542 3507.741
## 1995 2823.798 2841.675 2984.393 3513.596
## 1996 2845.165 3023.382 3042.978 3548.948
## 1997 2961.463 3082.174 3045.749 3664.565
## 1998 3026.675 3186.708 3164.596 3841.122
## 1999 3256.196 3286.011 3391.991 4126.504
## 2000 3324.296 3339.583 3432.662 4007.453
## 2001 3139.977 3537.089 3500.530 4006.164
## 2002 3225.919 3398.311 3447.898 4046.204
## 2003 3395.678 3507.608 3499.322 4214.260
## 2004 3531.840 3569.136 3651.774 4416.849
## 2005 3581.949 3595.864 3738.955 4464.856
## 2006 3630.308 3675.948 3800.376 4465.363
## 2007 3605.389 3755.692 3855.026 4372.474
## 2008 3394.200 3408.195 3338.802 3906.686
## 2009 3161.459 3293.155 3329.063 3981.235
## 2010 3338.023 3414.293 3557.167 4183.643
## 2011 3485.353 3528.410 3682.779 4294.668
## 2012 3508.872 3645.910 3794.737 4302.697
## 2013 3569.998 3752.686 3851.006 4379.163
## 2014 3728.061 3877.105 3929.377 4553.330
## 2015 3805.022 3912.488 3955.257 4650.002
## 2016 3869.978 3896.476 4064.651 4731.878
## 2017 3961.624 4014.356 4246.404 4776.077
## 2018 3920.035 4145.054 4326.168 4659.272
## 2019 3953.997 4190.896 4306.268 4786.743
## 2020 4318.940 4497.832 4479.441 5085.082
## 2021 4603.993 4779.407 4943.215 5380.018
## 2022
```

Let's start plotting the data to have look:

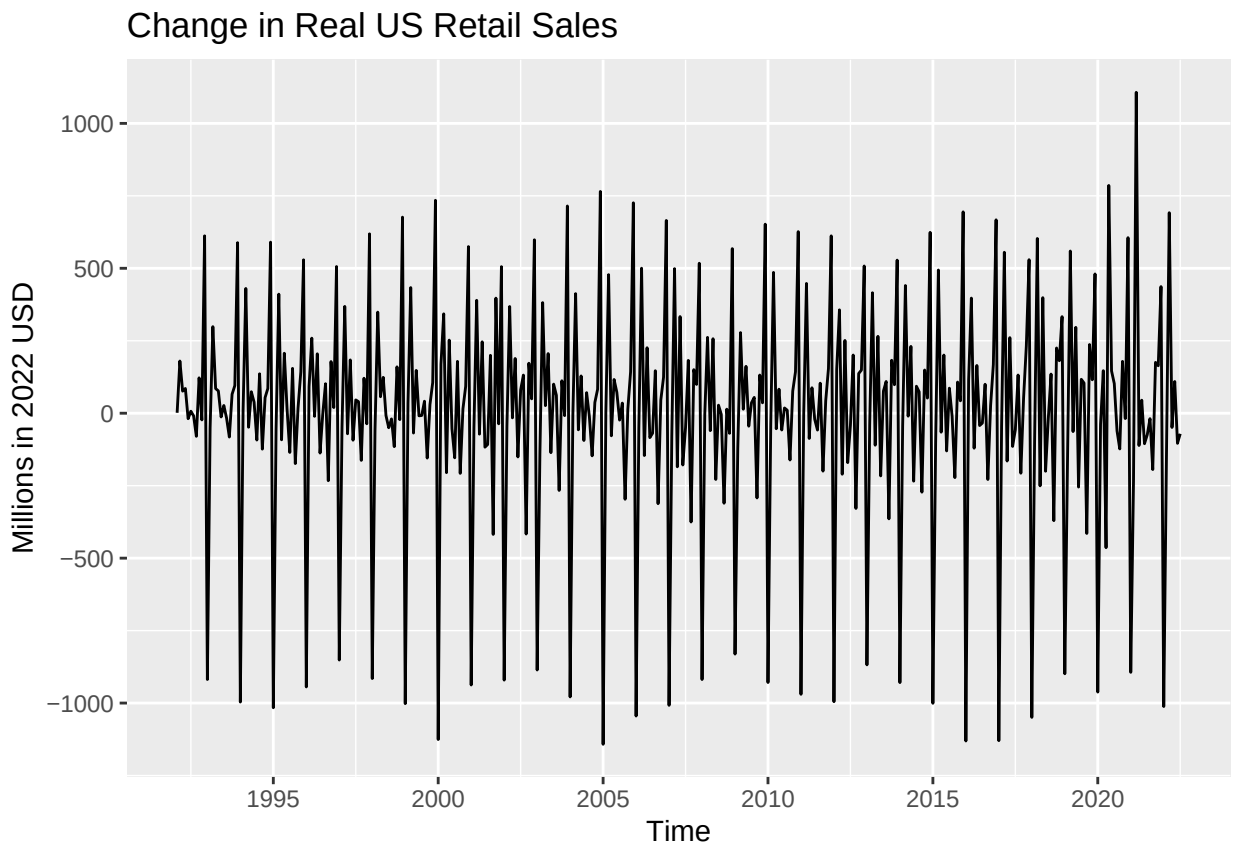
```
autoplot(data)+ggtitle("Real US Retail Sales")+ylab("Millions in 2022 USD")+xlab('Year')
```



We have a strong positive trend in growth. We can see the dip which is probably explained by the 2008 recesion. It seems like there is some seasonality in the data. We will investigate that further.

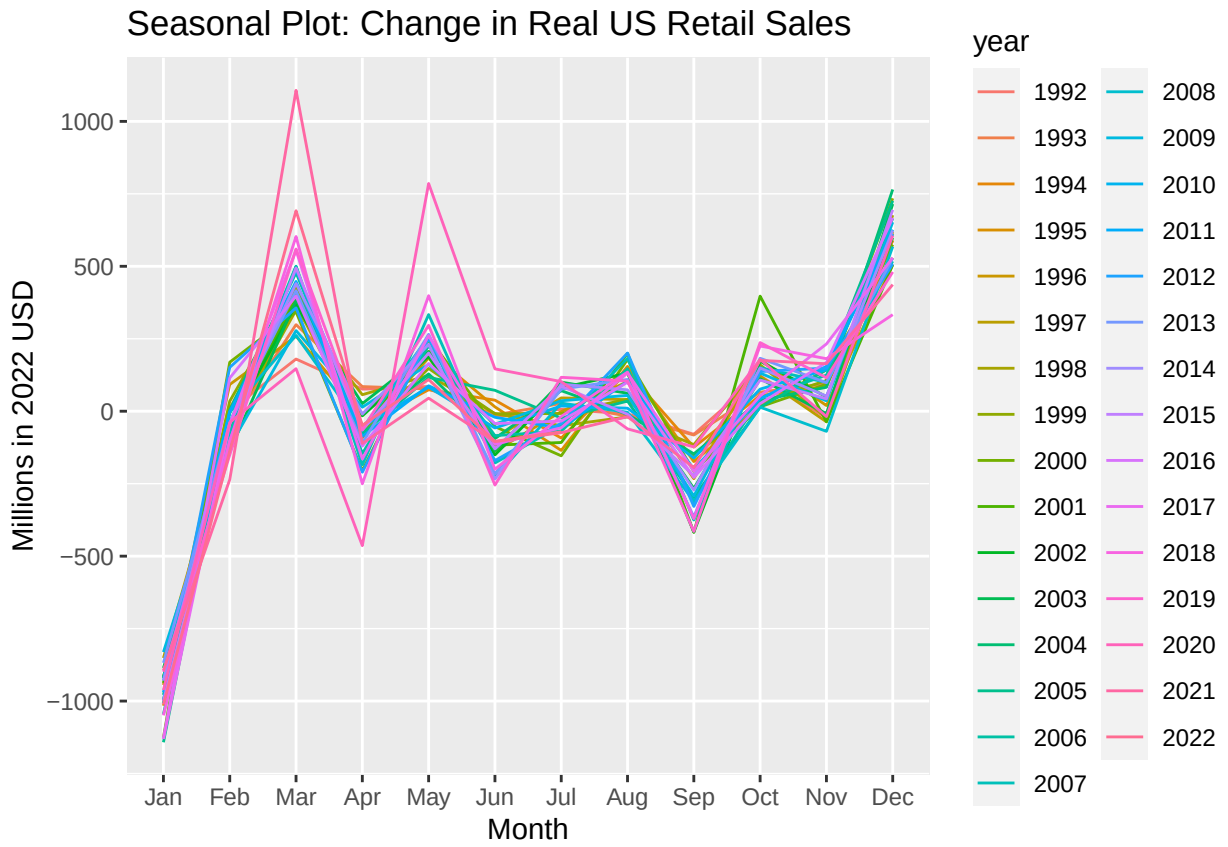
But first let's have a look at the stationary data by taking the first difference:

```
diff  = diff(data)
autoplot(diff)+ggtitle("Change in Real US Retail Sales")+ylab("Millions in 2022 USD")
```



We see that the data is now trend stationary. Now we have got rid of the trend. We can see large fluctuations in the data. We need to check whether these fluctuations are regualar or irregualar. Let's create a seasonal plot for the same.

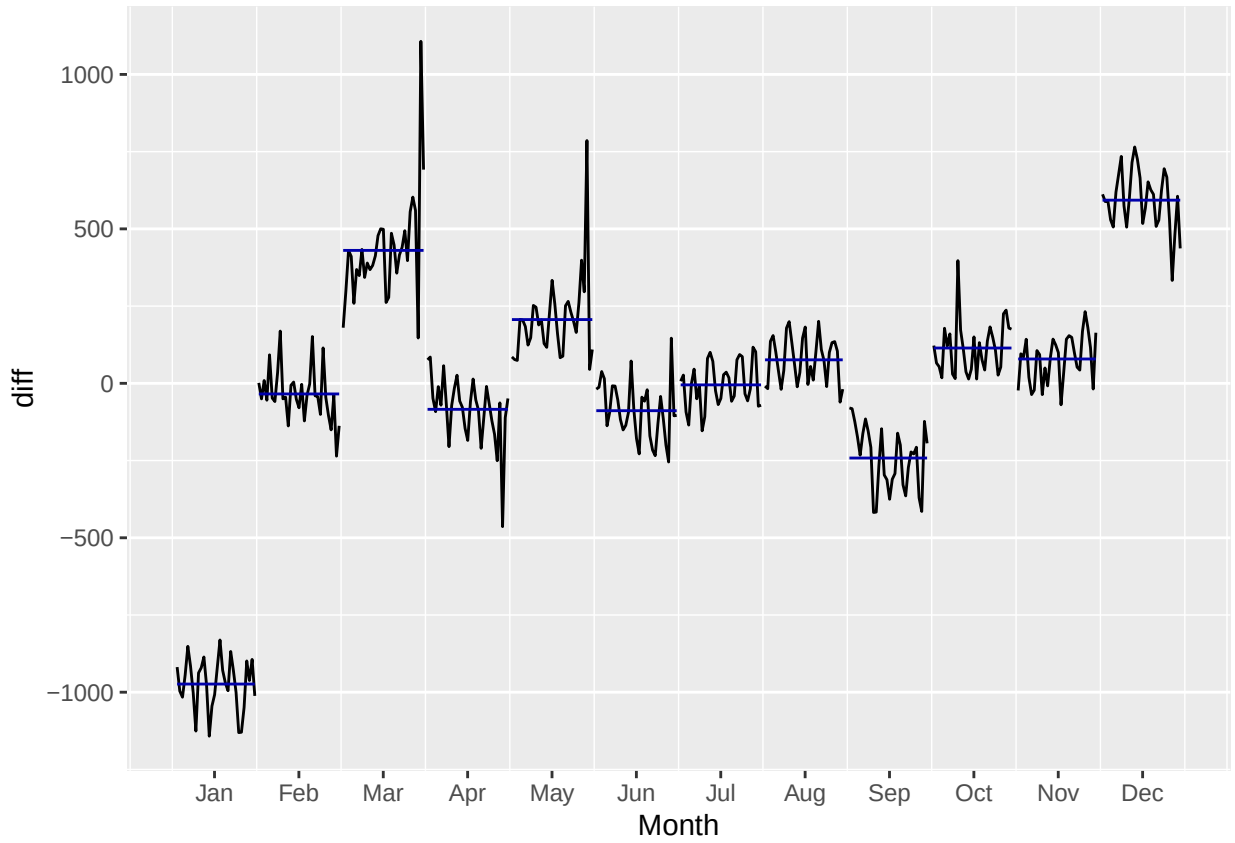
```
ggseasonplot(diff)+ggtitle("Seasonal Plot: Change in Real US Retail Sales")+ylab("Millions in 2022 USD")
```



We can see strong seasonality in the data. There is consistent growth in November to December months succeded by a sharp fall in January and which it picks up again in February to March. This upsurge and fall can be attributed to Christman shopping. The next surge and fall can probably be explained by the upcoming summer, change in retail stock and Easter.

Let's confirm our observationss with a subseries plot:

```
ggsubseriesplot(diff)
```



As we can see when the data is coupled by months for each year we can confirm our observations about the surge and fall in January and other months. We can also see some rather exception growth in sales in the month of March and June in the recent times. I went back to the previuos graph to confirm what I had missed earlier. This is indicative of strong positive growth in future but also of greater volatility. We have confirmed that our data has both trend and seasonality!

This is enough for a preliminary analysis. We can now proceed with modeling and eventually making our projections based on the model of our choice.

Let's start with the benchmark method, i.e., Naive methods. Since our data is seasonal we will go with seasonal naive, `snaive()` function which will use the corresponding season from last year as a benchmark. Mathematically, we can write this model as $y_t = y_{t-s} + \epsilon_t$, where t is time, s is the duration going back for the benchmark, and y_t is the sales in year t. Now because the data has a trend therefore that would imply compounding errors for the model hence it would be better to use the first difference data i.e., stationary data for this model.

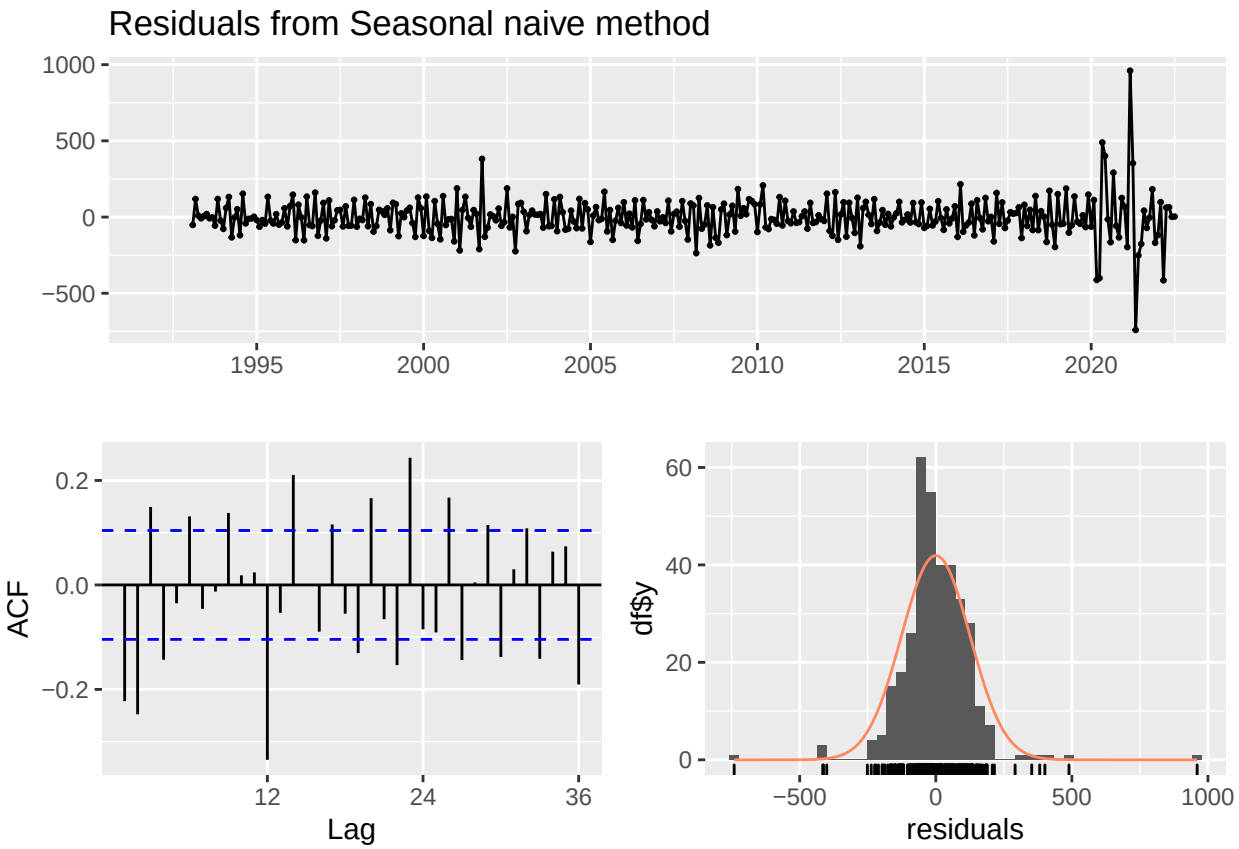
We'll fit the model, summarize it and check its residuals.

```
fit_naive = snaive(diff)
summary(fit_naive)
```

```
##
## Forecast method: Seasonal naive method
##
## Model Information:
## Call: snaive(y = diff)
##
```

```
## Residual sd: 125.9141
##
## Error measures:
##                       ME     RMSE      MAE      MPE     MAPE MASE       ACF1
## Training set -0.1234631 125.9141 86.10822 -25.8476 163.9355    1 -0.2224476
##
## Forecasts:
##          Point Forecast        Lo 80      Hi 80       Lo 95      Hi 95
## Aug 2022      -18.79436  -180.159710  142.57099  -265.58137  227.99265
## Sep 2022     -194.29125  -355.656599  -32.92590  -441.07826   52.49576
## Oct 2022      175.41414    14.048790  336.77949   -71.37287  422.20115
## Nov 2022      163.80768     2.442328  325.17303   -82.97933  410.59469
## Dec 2022      436.80272   275.437369  598.16807   190.01571  683.58973
## Jan 2023    -1011.87677 -1173.242125 -850.51142 -1258.66378 -765.08977
## Feb 2023     -137.66373  -299.029081   23.70162  -384.45074  109.12328
## Mar 2023      691.79857   530.433223  853.16393   445.01157  938.58558
## Apr 2023      -49.05961  -210.424956  112.30575  -295.84661  197.72740
## May 2023      109.43095   -51.934404  270.79630  -137.35606  356.21795
## Jun 2023     -103.83776  -265.203111   57.52759  -350.62477  142.94925
## Jul 2023      -70.88651  -232.251857   90.47885  -317.67351  175.90050
## Aug 2023      -18.79436  -246.999427  209.41071  -367.80389  330.21518
## Sep 2023     -194.29125  -422.496316   33.91382  -543.30078  154.71829
## Oct 2023      175.41414   -52.790927  403.61921  -173.59539  524.42368
## Nov 2023      163.80768   -64.397389  392.01275  -185.20185  512.81721
## Dec 2023      436.80272   208.597652  665.00779    87.79319  785.81225
## Jan 2024    -1011.87677 -1240.081842 -783.67171 -1360.88631 -662.86724
## Feb 2024     -137.66373  -365.868798   90.54134  -486.67326  211.34580
## Mar 2024      691.79857   463.593506  920.00364   342.78904 1040.80811
## Apr 2024      -49.05961  -277.264673  179.14546  -398.06914  299.94993
## May 2024      109.43095  -118.774121  337.63601  -239.57859  458.44048
## Jun 2024     -103.83776  -332.042828  124.36731  -452.84729  245.17177
## Jul 2024      -70.88651  -299.091574  157.31856  -419.89604  278.12303
```

```
checkresiduals(fit_naive)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Seasonal naive method
## Q* = 190.42, df = 24, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 24
```

We have a residual standard deviation of 125.9141. This is our benchmark for other models that we may try. This tells us that the seasonal naive model fits the data relatively well and misses on average by about 126 Million USD.

Cheching residuals we see that the data seems pretty random. However, looking at the ACF graph we see a lot of autocorrelation in the graph over the 95 percent confidence intervals. This means that the model is failing to account for some information in the data. We will have to lower our auto correlation.

For our next model, we'll try out the exponential smoothing model. These are a class of time dseries forecasting model. This model with try out every single possible exponential smoothing model and it will return the best fitter. This model will test for trend and include it in the fit. Therefore, we can use the real sales data for this one.

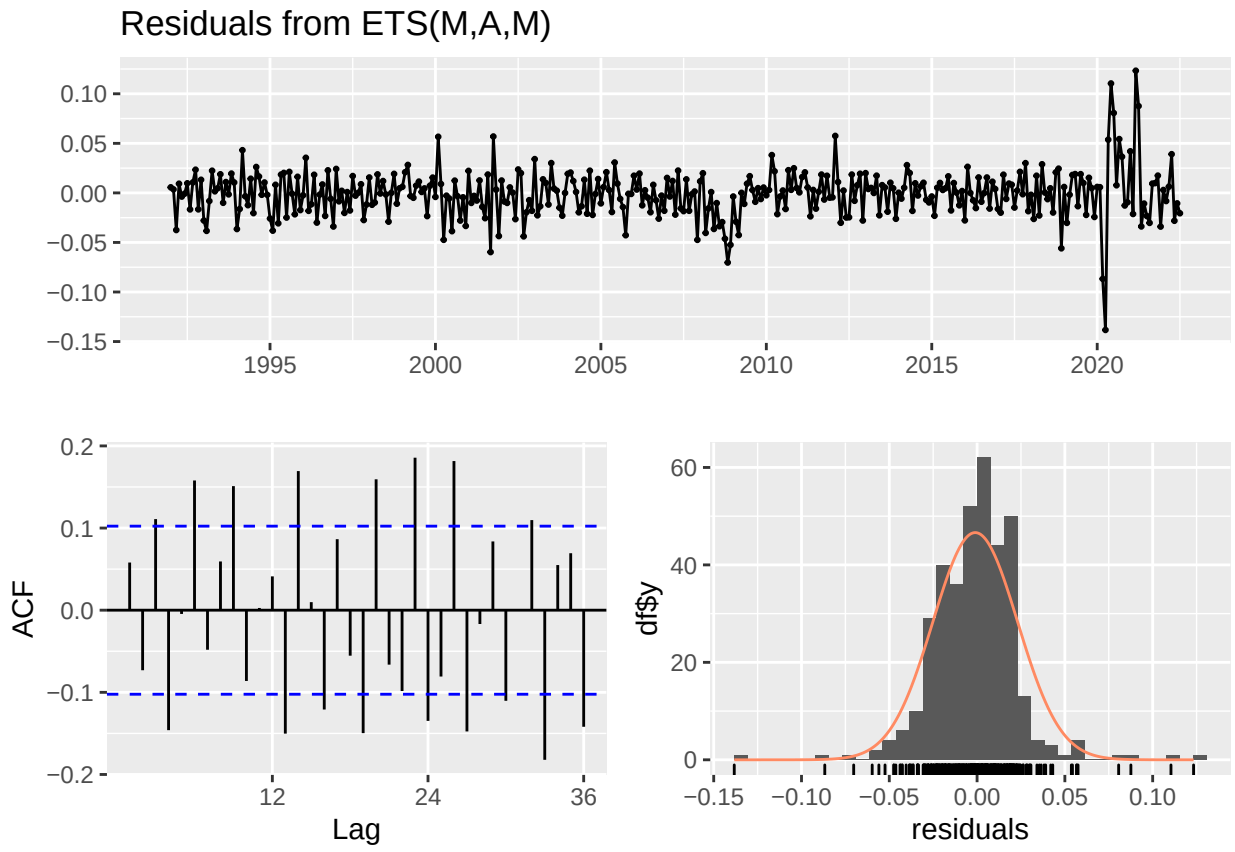
Let's try it out:

```
fit_ets = ets(data)
summary(fit_ets)
```

```
## ETS(M,A,M)
```

```
##
## Call:
##  ets(y = data)
##
##   Smoothing parameters:
##     alpha = 0.4086
##     beta  = 0.0021
##     gamma = 0.1567
##
##   Initial states:
##     l = 2492.5071
##     b = 10.393
##     s = 1.2121 1.007 0.9895 0.9632 1.0187 1.0036
##            1.0152 1.0282 0.9892 0.9943 0.8878 0.8911
##
##   sigma:  0.0247
##
##      AIC     AICc      BIC
## 5465.513 5467.266 5531.904
##
## Training set error measures:
##                     ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -4.535841 93.22968 62.61398 -0.1638317 1.716813 0.4982476
##                    ACF1
## Training set 0.08720487
```

```
checkresiduals(fit_ets)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,A,M)
## Q* = 113.7, df = 8, p-value < 2.2e-16
##
## Model df: 16.   Total lags used: 24
```

The model picked by ETS is ETS(M,A,M) model i.e., the multiplicative, additive and multiplicative model. The first letter denotes the error type; the second letter denotes the trend type; and the third letter denotes the season type. So, the errors are being compounded and so is the seasonality.

This model can be described mathematically as:

$$y_t = (l_{t-1} + b_{t-1})s_{t-m}(1 + \epsilon_t)$$

$$l_t = (l_{t-1} + b_{t-1})s_{t-m}(1 + \alpha\epsilon_t)$$

$$b_t = (l_{t-1} + b_{t-1})s_{t-m}(1 + \epsilon_t)$$

$$s_t = s_{t-m}(1 + \gamma\epsilon_t)$$

where, α, β, γ are the soothing parameters, and, $l_0, b_0, s_0, s_{-1}, ..$ are the initial states.

We can see a residual standard deviation dropped to a whopping 0.0247, which is a massive improvement from the benchmark. It is almost too good to be true, I haven't figured out why this is so. Although, when it comes to auto correlation it is not an inprovement in any case if not degradation. We would like to see these within the 95 percent confidence intervals. Although, this model performed quite well at least in sample we can try out one more model known as the ARIMA model.

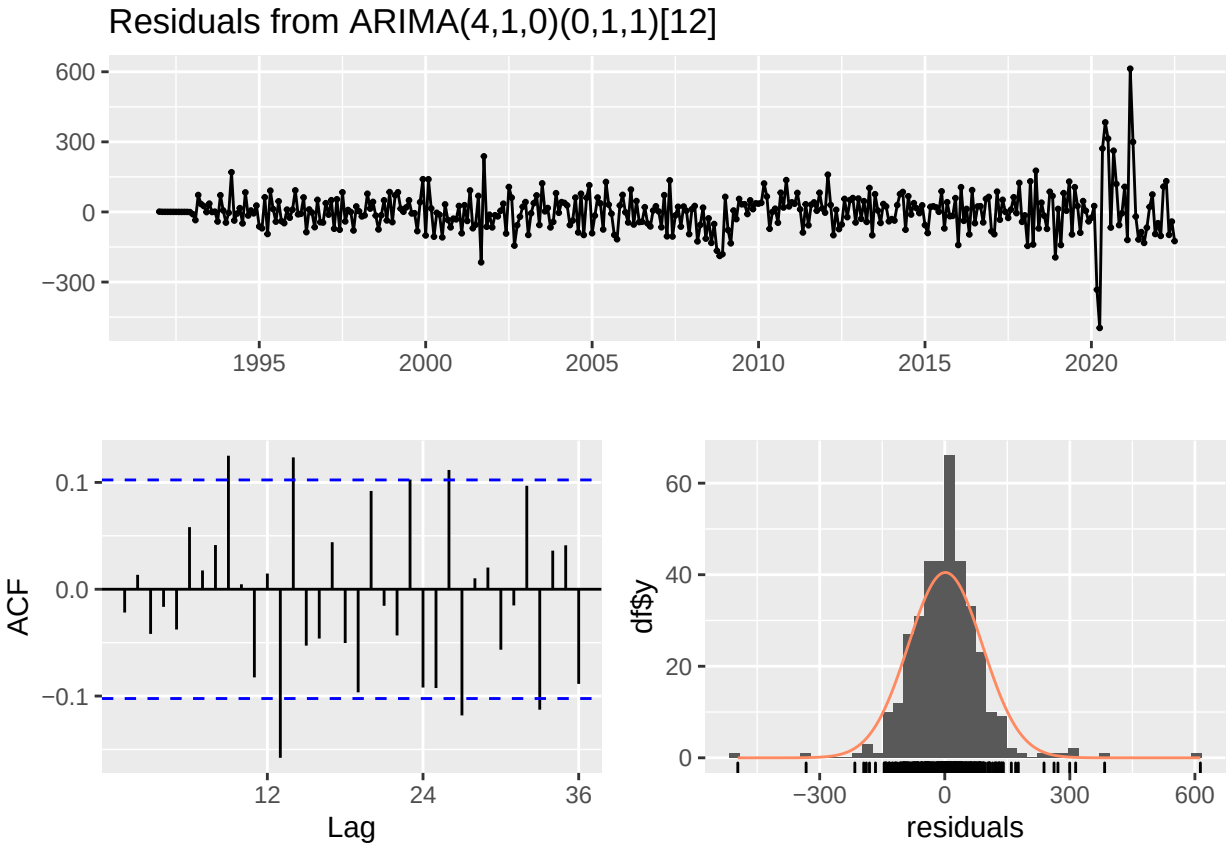
The `auto.arima()` function uses a variation of the Hyndman Khandakar algoritm which combines unit root tests, minimisation of the AICs and MLE to obtain an ARIMA model. We need stationary data for the ARIMA model therefore, we wll tell the function to take the first difference of the data we provide by supplying the constant `d=1`, we will also take the first seasonal difference by supplying `D=1`. We don't want it to approximate so we set the to `approximation=FALSE`, and setting `stepwise=FALSE` will force it to try all the models, while `trace=FALSE` implies it will not print out each model it tries but rather only the final selection.

Let's try it out:

```
fit_arima = auto.arima(data,d=1,D=1,stepwise=FALSE,approximation=FALSE, trace=FALSE)
summary(fit_arima)
```

```
## Series: data
## ARIMA(4,1,0)(0,1,1)[12]
##
## Coefficients:
##           ar1      ar2      ar3      ar4     sma1
##       -0.4088  -0.3656  -0.0386  -0.2042  -0.8091
## s.e.   0.0530   0.0569   0.0574   0.0527   0.0365
##
## sigma^2 = 8341:  log likelihood = -2104.57
## AIC=4221.14   AICc=4221.38   BIC=4244.35
##
## Training set error measures:
##                     ME     RMSE      MAE         MPE     MAPE      MASE
## Training set 1.365128 89.06185 60.60952 -0.01337592 1.649582 0.4822972
##                      ACF1
## Training set -0.02179607
```

```
checkresiduals(fit_arima)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(4,1,0)(0,1,1)[12]
## Q* = 46.393, df = 19, p-value = 0.0004359
##
## Model df: 5.   Total lags used: 24
```

The model selected is `ARIMA(4,1,0)(0,1,1)[12]`. This gives us a residual standard deviation of $\sqrt{8341} = 91.32908$ which is certainly lower than the benchmark but much much worse than ETS model.

$ARIMA(p,d,q)(P,D,Q)_S$, where (p,d,q) is the non seasonal part and (P,D,Q) is the seasonal part of the model, and S is the number of observations per year. Mathematically, $ARIMA(p,d,q)(P,D,Q)_S$ can be written as:

$$\Phi(B^S)\phi(B)(x_t - \mu) = \Theta(B^S)\theta(B)w_t$$

The non seasonal parts are: AR:

$$\phi(B) = 1 - \phi_1 B - ... - \phi_p B^p$$

MA:

$$\theta(B) = 1 + \theta_1 B + ... + \theta_q B^q$$

The seasonal parts are: Seasonal AR:

$$\Phi(B) = 1 - \Phi_1 B^S - ... - \Phi_P B^{PS}$$

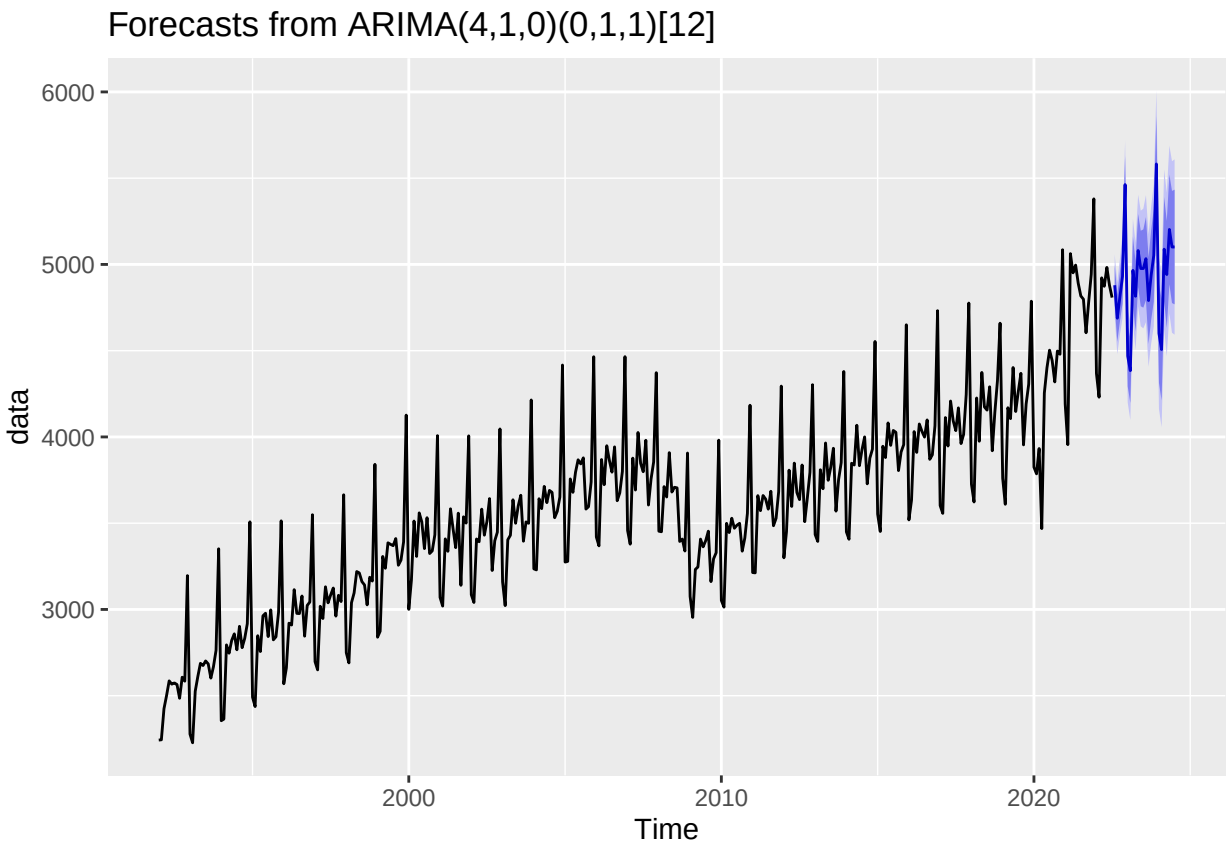
Seasonal MA:

$$\Theta(B) = 1 + \Theta_1 B^S + ... + \Theta_Q B^{QS}$$

We can see quite easily here that the autocorrelation is almost within the 95 percent confidence intervals. This is what we were looking for. Now we can proceed with forecasting the data.

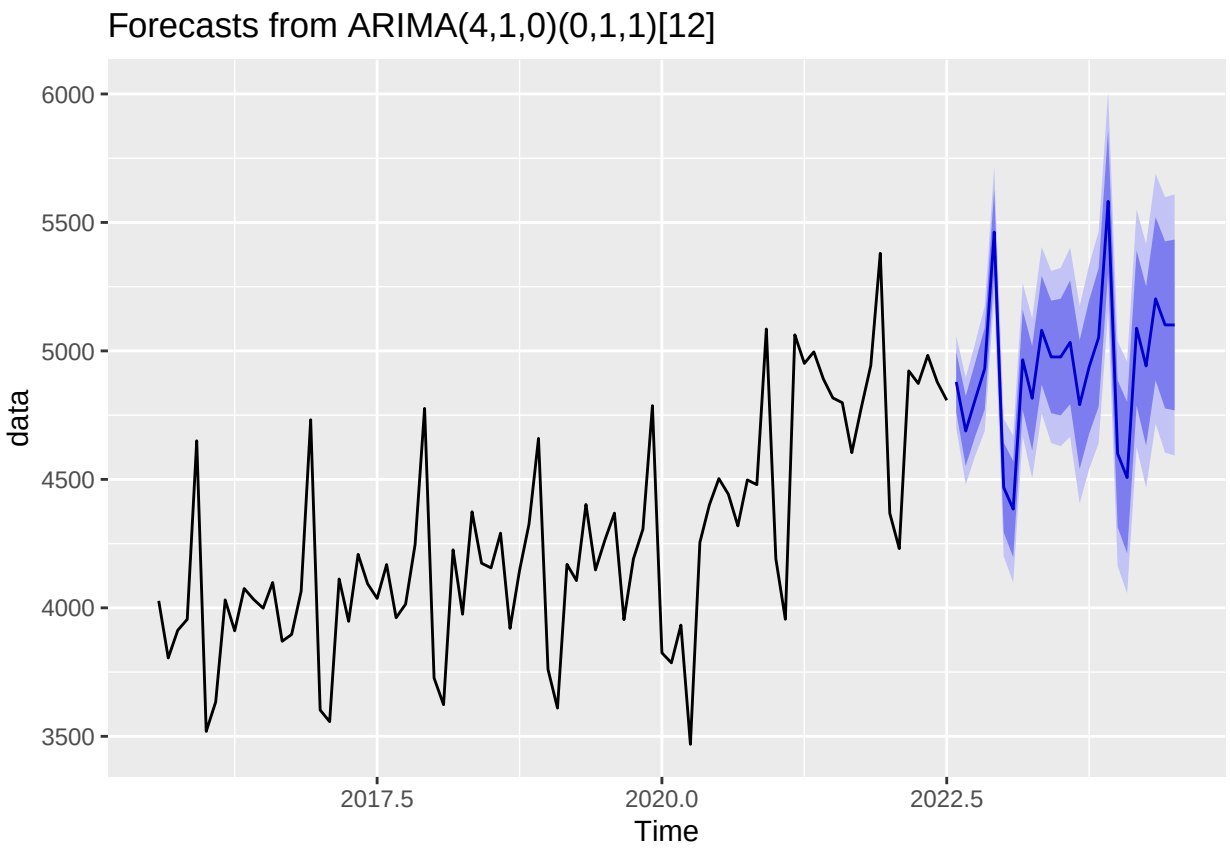
We will forecast using both the ETS and ARIMA models. First, with ARIMA:

```
forecast1 = forecast(fit_arima, h=24)
autoplot(forecast1)
```



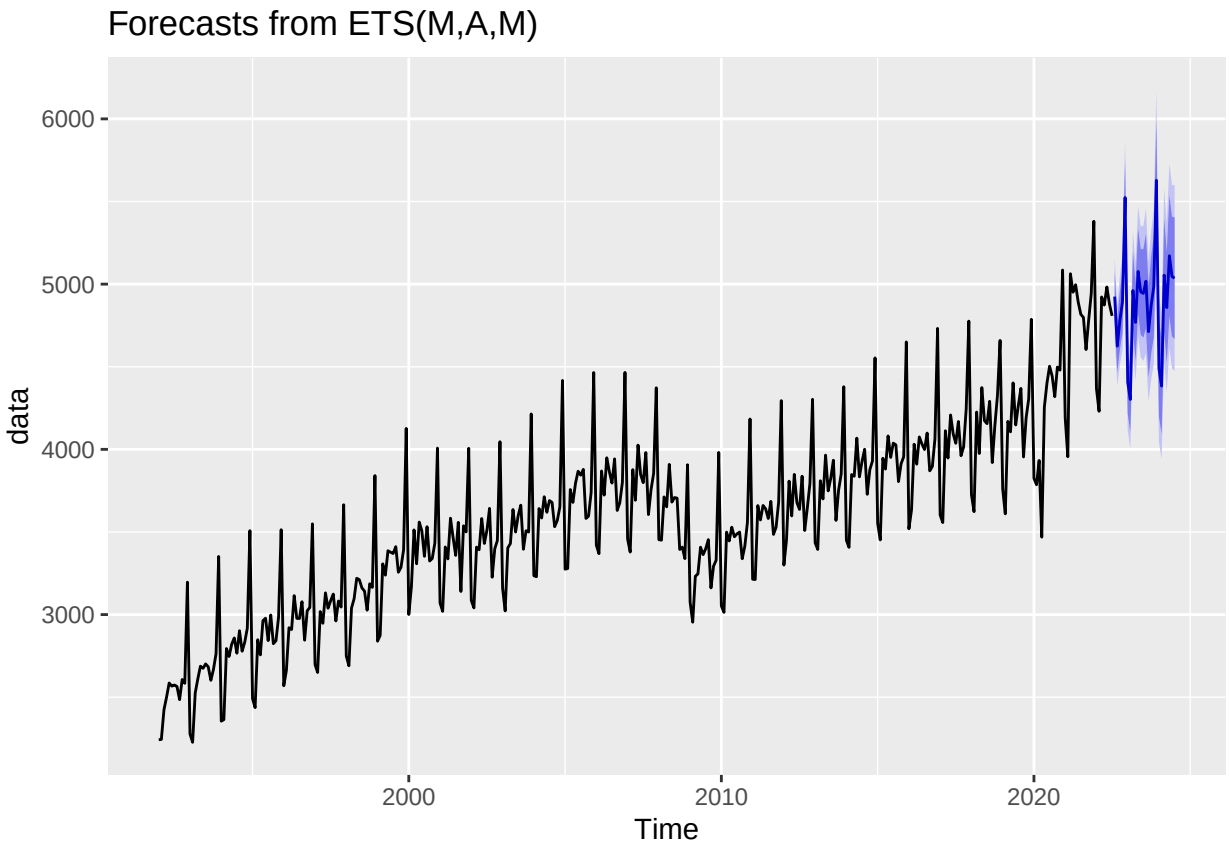
And let's look at the zoomed in plot for the same.

```
autoplot(forecast1, include = 84)
```



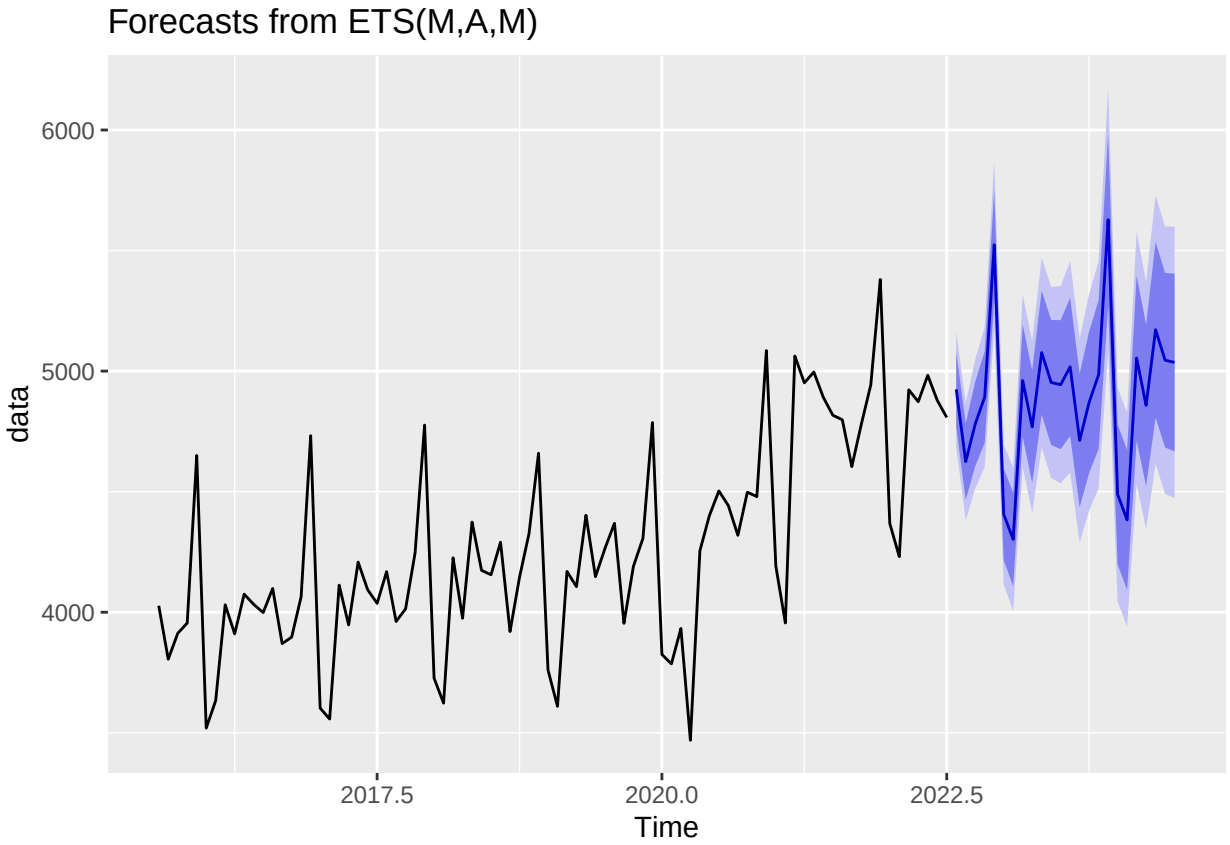
Now with ETS:

```
forecast2 = forecast(fit_ets, h=24)
autoplot(forecast2)
```



And let's look at the zoomed in plot for the same.

```
autoplot(forecast2, include = 84)
```



Next we can look at the summary of each forecast to conclude.

```
summary(forecast1)
```

```
##
## Forecast method: ARIMA(4,1,0)(0,1,1)[12]
##
## Model Information:
## Series: data
## ARIMA(4,1,0)(0,1,1)[12]
##
## Coefficients:
##           ar1      ar2      ar3      ar4     sma1
##       -0.4088  -0.3656  -0.0386  -0.2042  -0.8091
## s.e.   0.0530   0.0569   0.0574   0.0527   0.0365
##
## sigma^2 = 8341:  log likelihood = -2104.57
## AIC=4221.14   AICc=4221.38   BIC=4244.35
##
## Error measures:
##                     ME     RMSE      MAE         MPE     MAPE      MASE
## Training set 1.365128 89.06185 60.60952 -0.01337592 1.649582 0.4822972
##                      ACF1
## Training set -0.02179607
##
## Forecasts:
```

```
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Aug 2022       4879.388 4762.344 4996.432 4700.385 5058.391
## Sep 2022       4688.219 4552.249 4824.190 4480.271 4896.168
## Oct 2022       4811.463 4667.934 4954.992 4591.955 5030.972
## Nov 2022       4930.879 4771.871 5089.886 4687.698 5174.059
## Dec 2022       5462.812 5297.370 5628.254 5209.790 5715.834
## Jan 2023       4468.563 4293.433 4643.693 4200.725 4736.401
## Feb 2023       4384.288 4197.544 4571.032 4098.688 4669.888
## Mar 2023       4965.498 4771.158 5159.838 4668.280 5262.716
## Apr 2023       4815.443 4612.111 5018.775 4504.474 5126.412
## May 2023       5080.344 4868.710 5291.978 4756.678 5404.010
## Jun 2023       4976.529 4757.645 5195.414 4641.774 5311.285
## Jul 2023       4976.219 4749.405 5203.033 4629.337 5323.101
## Aug 2023       5033.439 4792.869 5274.010 4665.519 5401.360
## Sep 2023       4790.544 4539.790 5041.297 4407.049 5174.038
## Oct 2023       4937.407 4677.739 5197.075 4540.279 5334.535
## Nov 2023       5052.221 4783.095 5321.346 4640.629 5463.813
## Dec 2023       5582.307 5305.036 5859.578 5158.257 6006.357
## Jan 2024       4600.145 4314.365 4885.924 4163.083 5037.207
## Feb 2024       4506.959 4212.684 4801.235 4056.904 4957.015
## Mar 2024       5088.403 4786.290 5390.517 4626.360 5550.446
## Apr 2024       4941.421 4631.421 5251.421 4467.317 5415.525
## May 2024       5202.857 4885.239 5520.475 4717.102 5688.611
## Jun 2024       5101.146 4776.173 5426.118 4604.143 5598.148
## Jul 2024       5101.075 4768.783 5433.368 4592.878 5609.273
```

```
summary(forecast2)
```

```
##
## Forecast method: ETS(M,A,M)
##
## Model Information:
## ETS(M,A,M)
##
## Call:
##  ets(y = data)
##
##   Smoothing parameters:
##     alpha = 0.4086
##     beta  = 0.0021
##     gamma = 0.1567
##
##   Initial states:
##     l = 2492.5071
##     b = 10.393
##     s = 1.2121 1.007 0.9895 0.9632 1.0187 1.0036
##            1.0152 1.0282 0.9892 0.9943 0.8878 0.8911
##
##   sigma:  0.0247
##
##      AIC     AICc      BIC
## 5465.513 5467.266 5531.904
##
## Error measures:
```

```
##                       ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -4.535841 93.22968 62.61398 -0.1638317 1.716813 0.4982476
##                     ACF1
## Training set 0.08720487
##
## Forecasts:
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Aug 2022       4924.497 4768.661 5080.333 4686.166 5162.828
## Sep 2022       4624.798 4466.612 4782.984 4382.873 4866.723
## Oct 2022       4780.428 4605.460 4955.396 4512.838 5048.018
## Nov 2022       4893.122 4702.946 5083.297 4602.274 5183.970
## Dec 2022       5524.087 5297.485 5750.689 5177.529 5870.645
## Jan 2023       4405.996 4216.177 4595.816 4115.692 4696.300
## Feb 2023       4302.160 4108.293 4496.027 4005.666 4598.655
## Mar 2023       4961.199 4728.157 5194.242 4604.791 5317.607
## Apr 2023       4768.679 4535.867 5001.491 4412.624 5124.734
## May 2023       5076.724 4819.771 5333.676 4683.749 5469.698
## Jun 2023       4952.790 4693.478 5212.103 4556.206 5349.375
## Jul 2023       4943.870 4676.628 5211.111 4535.159 5352.580
## Aug 2023       5017.896 4731.286 5304.507 4579.564 5456.229
## Sep 2023       4712.375 4435.762 4988.989 4289.332 5135.419
## Oct 2023       4870.810 4577.345 5164.276 4421.994 5319.627
## Nov 2023       4985.490 4677.530 5293.450 4514.505 5456.475
## Dec 2023       5628.203 5272.129 5984.277 5083.634 6172.771
## Jan 2024       4488.909 4198.316 4779.503 4044.485 4933.333
## Feb 2024       4382.993 4092.916 4673.069 3939.359 4826.627
## Mar 2024       5054.269 4712.566 5395.973 4531.678 5576.861
## Apr 2024       4857.998 4522.742 5193.255 4345.268 5370.729
## May 2024       5171.666 4807.599 5535.732 4614.874 5728.457
## Jun 2024       5045.271 4683.206 5407.337 4491.540 5599.003
## Jul 2024       5036.041 4667.839 5404.243 4472.925 5599.157
```

I, for one, can't spot any major differences between the projections of these two models. So either of them will be useful and the choice of which one will depend on the data one is working with and the problem that one is trying to solve. ETS has a spectacularly low residual standard deviation at the expense of some auto correlation. While ARIMA minimises auto correlation for a reasonably low residual standard deviation. This concludes this project.

References: [1] Sales data RSXFSN taken from: FRED ECONOMIC DATA [2] CPI Data USACPIALLMINMEI taken from: FRED ECONOMIC DATA