finalquestion

```r
library(itsmr)
library(fpp2)
```

```
## Loading required package: ggplot2

## Loading required package: forecast

##
## Attaching package: 'forecast'

## The following object is masked from 'package:itsmr':
##
##     forecast

## Loading required package: fma

##
## Attaching package: 'fma'

## The following objects are masked from 'package:itsmr':
##
##     airpass, strikes

## Loading required package: expsmooth
```

```r
library(expsmooth)
library(forecast)
library(tibbletime)
```

```
##
## Attaching package: 'tibbletime'

## The following object is masked from 'package:stats':
##
##     filter
```

```r
library(tidyverse)
```

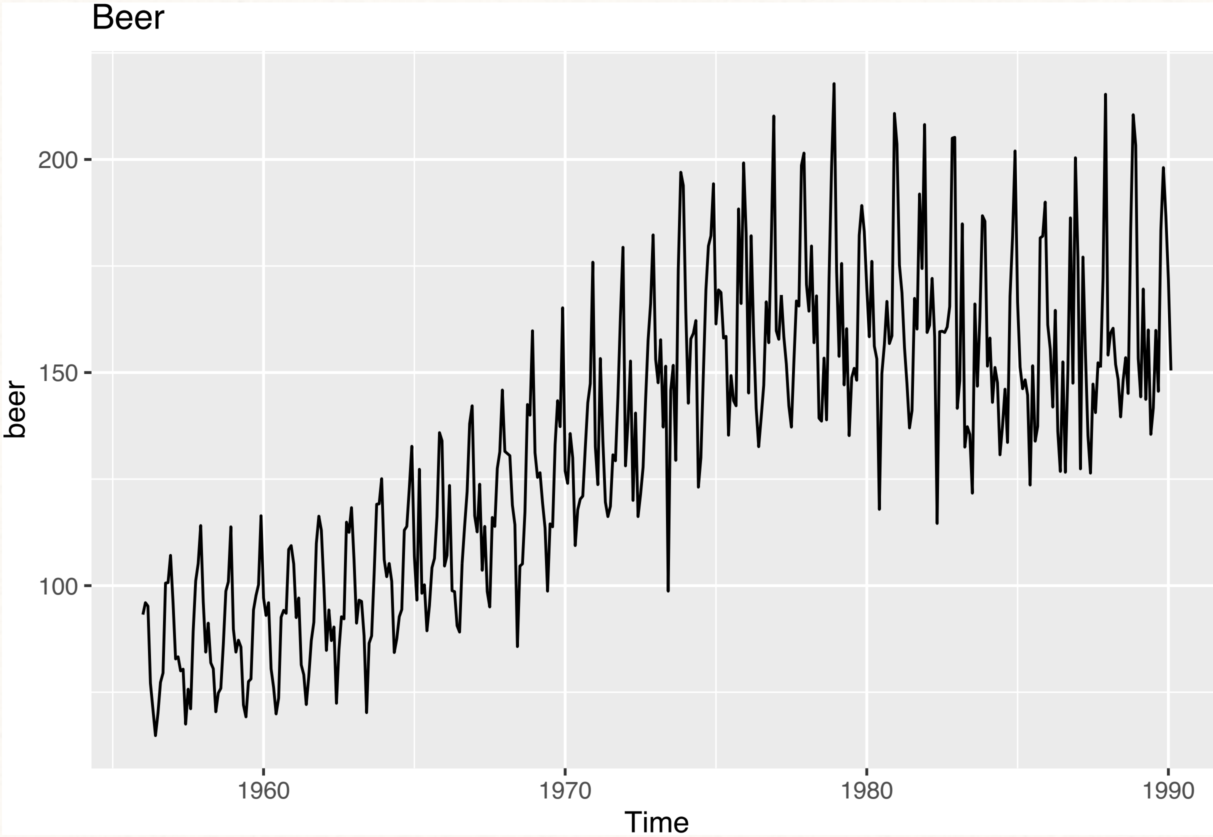
```
## -- Attaching packages ----------------------------------------------------------------------------------

## v tibble  2.1.3     v dplyr   0.8.4
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
## v purrr   0.3.3

## -- Conflicts -------------------------------------------------------------------------------------------
## x dplyr::filter() masks tibbletime::filter(), stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(tsbox)
library(knitr)
```

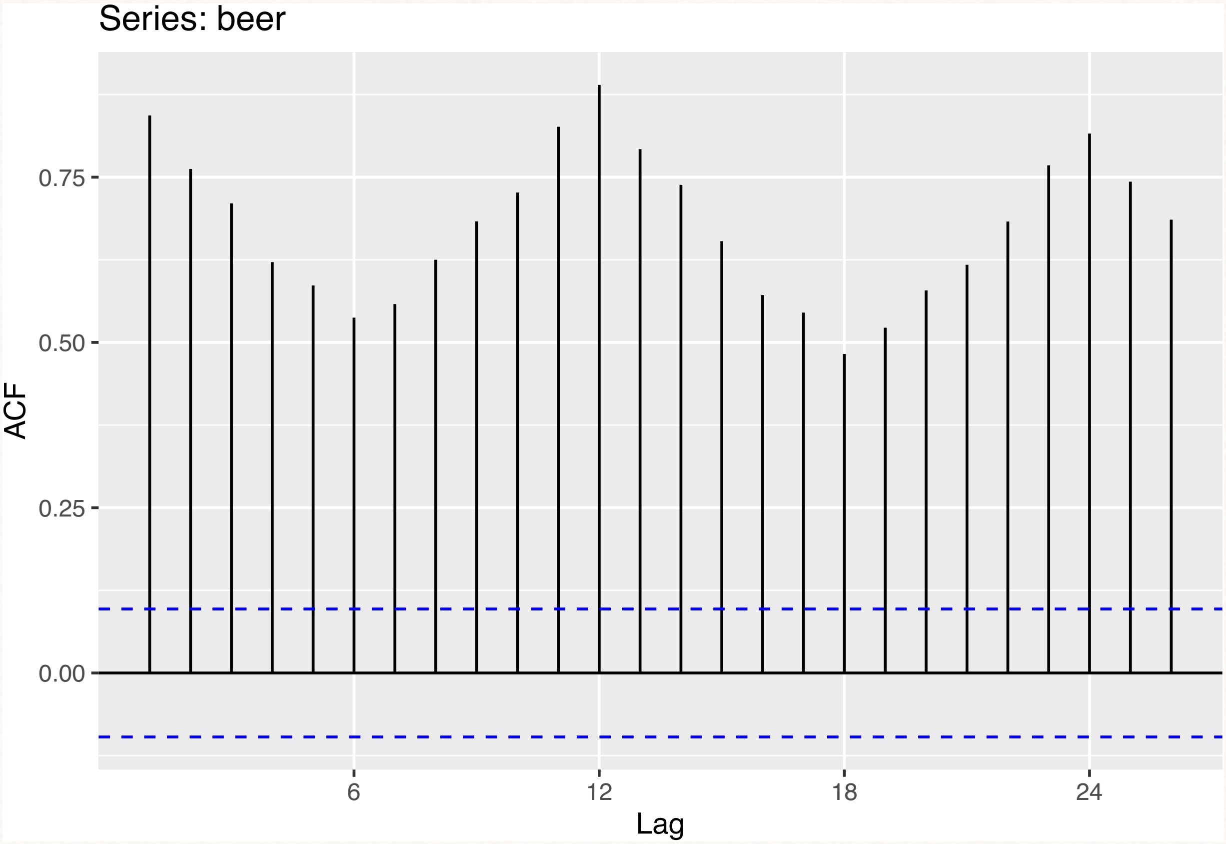
##(a)

```r
beer_dat=dget("beer.Rput")
# remove the last 12 values
beer<-head(beer_dat,-12)
```

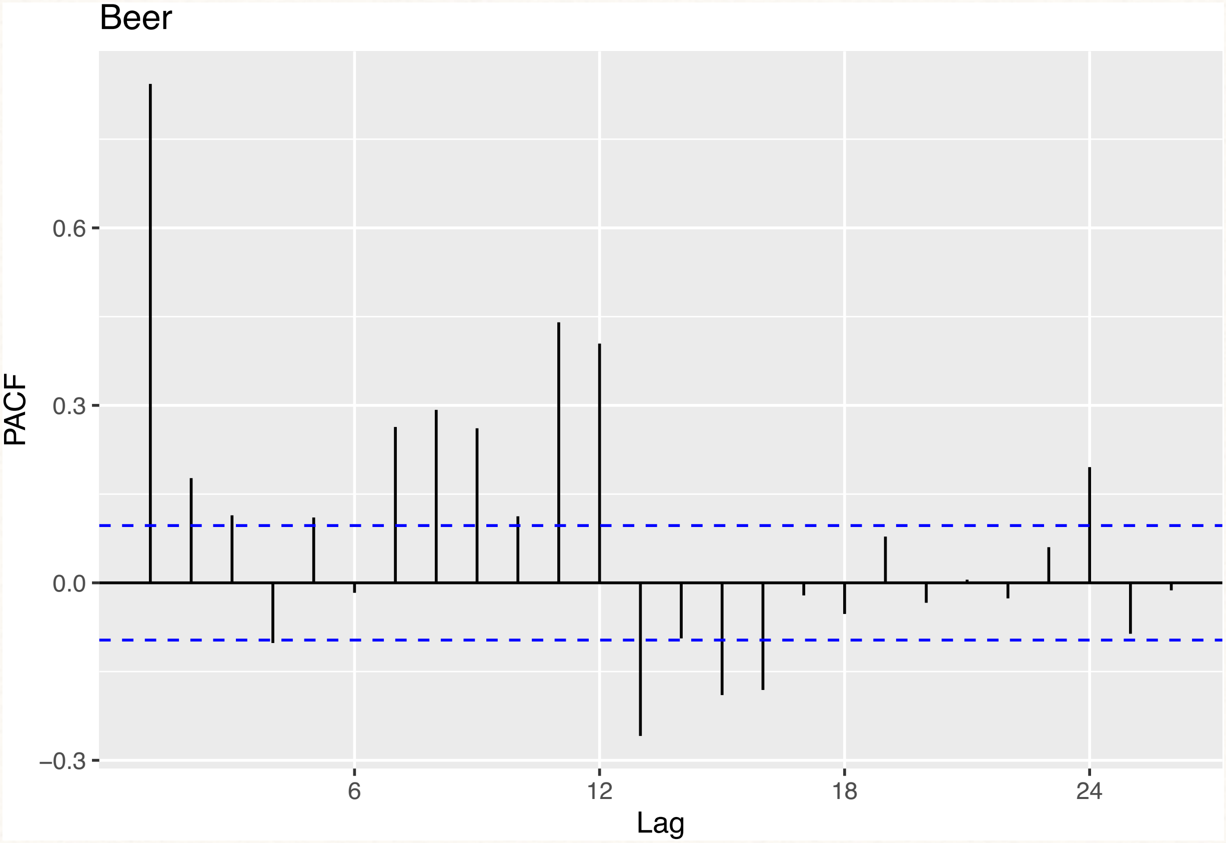
```r
# plot the data
autoplot(beer)+ggtitle("Beer")
```




```r
ggAcf(beer)
```



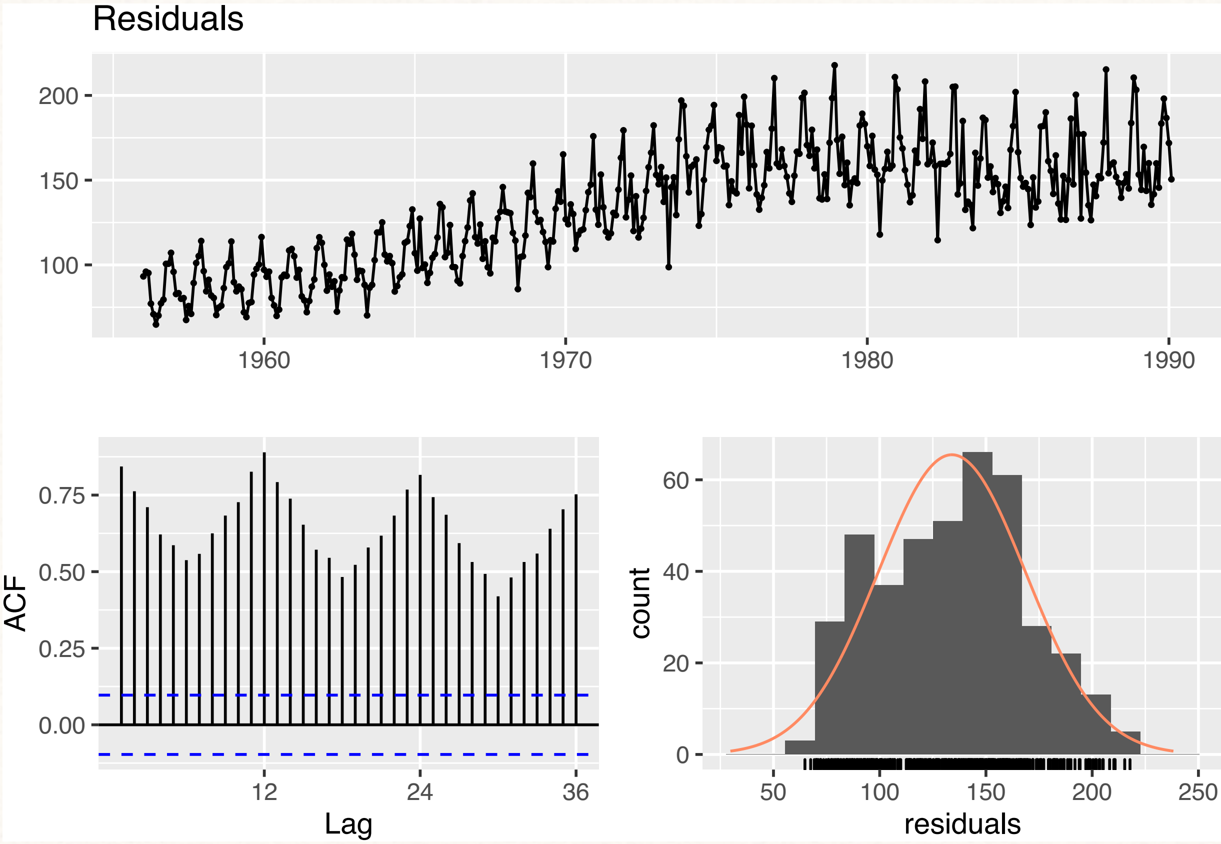

```r
ggPacf(beer)+ggtitle("Beer")
```



It shows that the data is not stationay but contain the seasonal component and noise.Also the above ACF plot shows the seasonal componet and the data are correlated. It is obvious that this dataset is not a white noise process.

```
# check the residual by deseasonalizing
#sales_ma5<-ma(beer,order=5)
#tslm<-tslm(beer~trend+I(trend^2)+season)
#stl<-stl(beer,s.window=12)
#checkresiduals(stl$time.series[,'remainder'])
checkresiduals(beer)
```

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



It shows that the data is not stationay but contain the seasonal component and noise.Also the above ACF plot and the residual plot shows the seasonal componet and the data are correlated. It is obvious that this dataset is not a white noise process.

```
# find the arima model
beer_ari<-auto.arima(beer,stepwise=FALSE, seasonal=FALSE, ic='aic',trace=TRUE)
```

```
##
##  Fitting models using approximations to speed things up...
##
##  ARIMA(0,1,0)                                : 3579.226
##  ARIMA(0,1,0)            with drift          : 3581.204
##  ARIMA(0,1,1)                                : 3547.203
##  ARIMA(0,1,1)            with drift          : 3549.134
##  ARIMA(0,1,2)                                : 3493.171
##  ARIMA(0,1,2)            with drift          : 3489.323
##  ARIMA(0,1,3)                                : 3490.168
##  ARIMA(0,1,3)            with drift          : 3486.407
##  ARIMA(0,1,4)                                : 3468.429
##  ARIMA(0,1,4)            with drift          : 3464.997
##  ARIMA(0,1,5)                                : 3469.946
##  ARIMA(0,1,5)            with drift          : 3466.611
##  ARIMA(1,1,0)                                : 3556.574
##  ARIMA(1,1,0)            with drift          : 3558.536
##  ARIMA(1,1,1)                                : 3483.419
##  ARIMA(1,1,1)            with drift          : 3480.55
##  ARIMA(1,1,2)                                : 3485.297
##  ARIMA(1,1,2)            with drift          : 3482.359
```

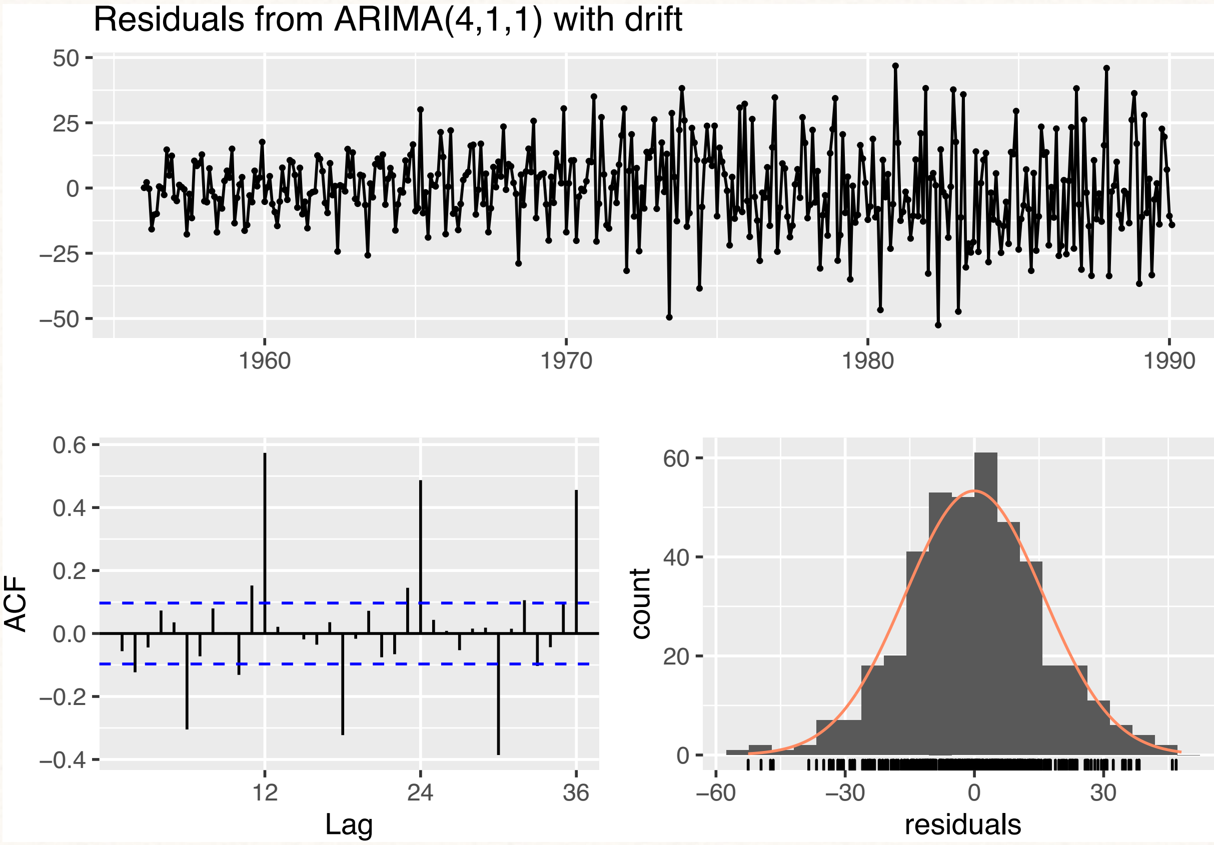
```
##  ARIMA(1,1,3)                              : 3486.94
##  ARIMA(1,1,3)           with drift         : 3483.973
##  ARIMA(1,1,4)                              : 3471.249
##  ARIMA(1,1,4)           with drift         : 3468.278
##  ARIMA(2,1,0)                              : 3547.96
##  ARIMA(2,1,0)           with drift         : 3549.898
##  ARIMA(2,1,1)                              : 3540.196
##  ARIMA(2,1,1)           with drift         : 3542.157
##  ARIMA(2,1,2)                              : 3481.216
##  ARIMA(2,1,2)           with drift         : 3477.923
##  ARIMA(2,1,3)                              : 3482.351
##  ARIMA(2,1,3)           with drift         : 3479.135
##  ARIMA(3,1,0)                              : 3549.09
##  ARIMA(3,1,0)           with drift         : 3551.011
##  ARIMA(3,1,1)                              : 3543.732
##  ARIMA(3,1,1)           with drift         : 3545.695
##  ARIMA(3,1,2)                              : 3481.842
##  ARIMA(3,1,2)           with drift         : 3476.954
##  ARIMA(4,1,0)                              : 3541.237
##  ARIMA(4,1,0)           with drift         : 3543.099
##  ARIMA(4,1,1)                              : 3449.546
##  ARIMA(4,1,1)           with drift         : 3443.193
##  ARIMA(5,1,0)                              : 3543.721
##  ARIMA(5,1,0)           with drift         : 3545.553
##
##  Now re-fitting the best model(s) without approximations...
##
##
##
##
##  Best model: ARIMA(4,1,1)            with drift
```

Now analyze the model:

```r
summary(beer_ari)
```

```
## Series: beer
## ARIMA(4,1,1) with drift
##
## Coefficients:
##          ar1     ar2     ar3      ar4      ma1   drift
##       0.4462  0.0028  0.0763  -0.3170  -0.9395  0.1977
## s.e.  0.0477  0.0518  0.0517   0.0475   0.0127  0.0632
##
## sigma^2 estimated as 261.2:  log likelihood=-1716.54
## AIC=3447.08   AICc=3447.36   BIC=3475.18
##
## Training set error measures:
##                       ME     RMSE      MAE       MPE     MAPE     MASE
## Training set -0.0588431 16.02236 12.37635 -1.226062 9.171119 1.306263
##                      ACF1
## Training set -0.05614064
```

```r
checkresiduals(beer_ari)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(4,1,1) with drift
## Q* = 377.63, df = 18, p-value < 2.2e-16
##
## Model df: 6.   Total lags used: 24
```

```r
#test(checkresiduals(beer_ari))
```

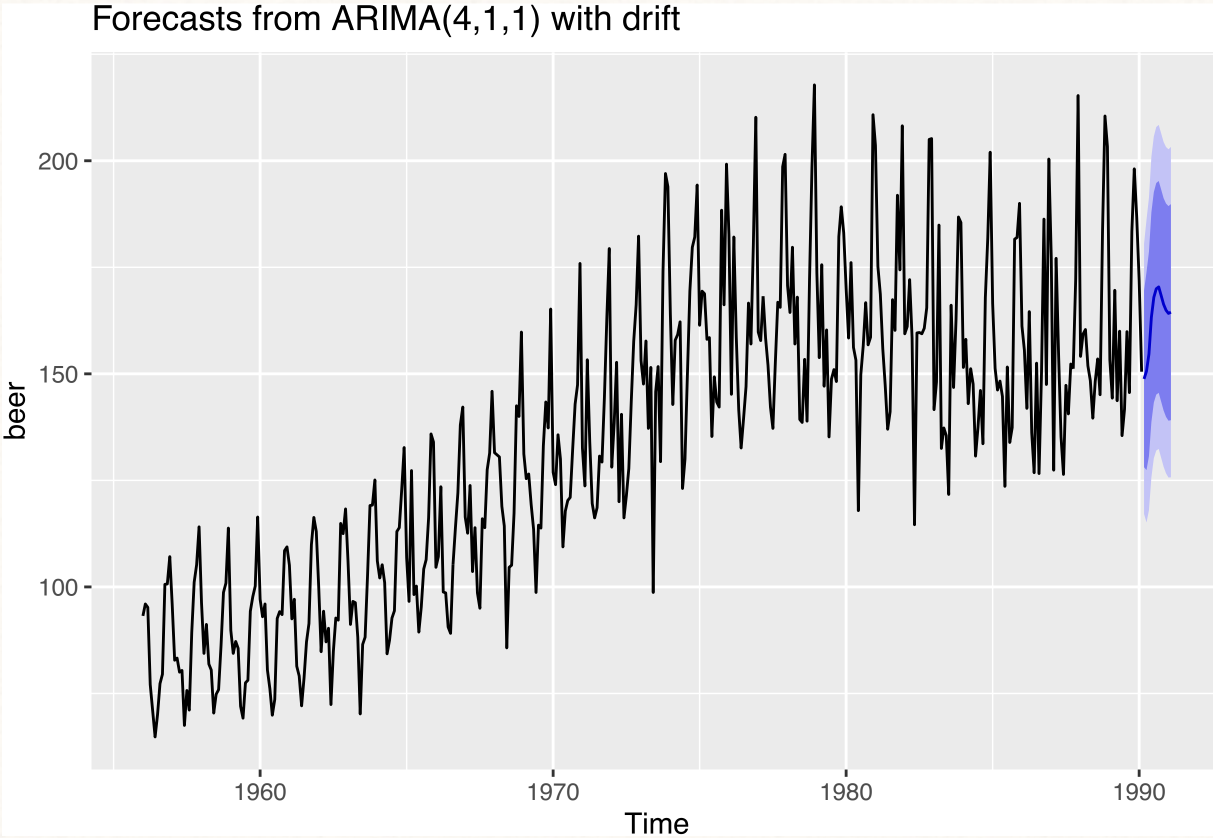
Construct the 95% confidence interval. Showing the bounds for component θ and ϕ

```r
confint(beer_ari)
```

```
##             2.5 %     97.5 %
## ar1    0.35276931  0.5396158
## ar2   -0.09861421  0.1043077
## ar3   -0.02492603  0.1775627
## ar4   -0.41008956 -0.2239423
## ma1   -0.96429447 -0.9146189
## drift  0.07376610  0.3216664
```

Now forecast on the 12 removed data

```r
beer_forecast<-forecast(beer_ari,h=12)
autoplot(beer_forecast)
```



Here's the result of prediction presented in table:

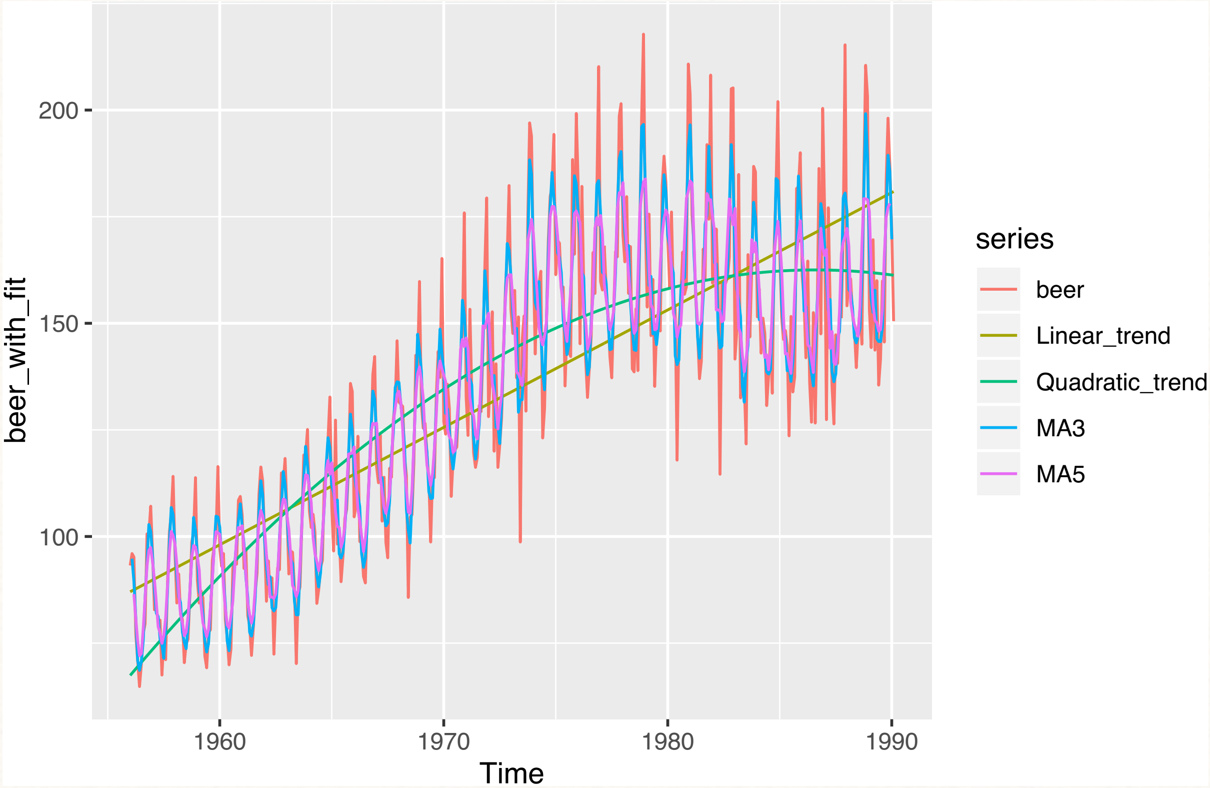
```
bt<-log(tail(beer_dat,12))
error<-bt-beer_forecast$mean
data<-data.frame(error, beer_forecast$lower, beer_forecast$upper,beer_forecast$mean)
colnames(data)<-c('error', '2.5%','97.5%','predict')
data
```

```
##        error    2.5%    97.5%  predict       NA       NA
## 1  -143.7117 128.0945 117.1307 169.5165 180.4803 148.8055
## 2  -145.5951 127.4111 115.1201 173.8479 186.1389 150.6295
## 3  -149.6245 130.6737 117.9794 178.6337 191.3279 154.6537
## 4  -158.3575 138.6542 125.6256 187.8773 200.9059 163.2658
## 5  -162.9536 143.2583 130.1858 192.6473 205.7197 167.9528
## 6  -164.9548 145.1331 131.9937 194.7749 207.9143 169.9540
## 7  -165.5043 145.5410 132.3823 195.2557 208.4145 170.3984
## 8  -163.1195 143.4968 130.3211 193.2759 206.4516 168.3864
## 9  -160.9738 141.3979 128.2084 191.2287 204.4182 166.3133
## 10 -159.6554 139.9032 126.6503 189.9740 203.2269 164.9386
## 11 -159.0816 139.0031 125.6744 189.3598 202.6884 164.1814
## 12 -159.4786 139.1259 125.7065 189.8257 203.2451 164.4758
```

##(c) First deseasonalizing

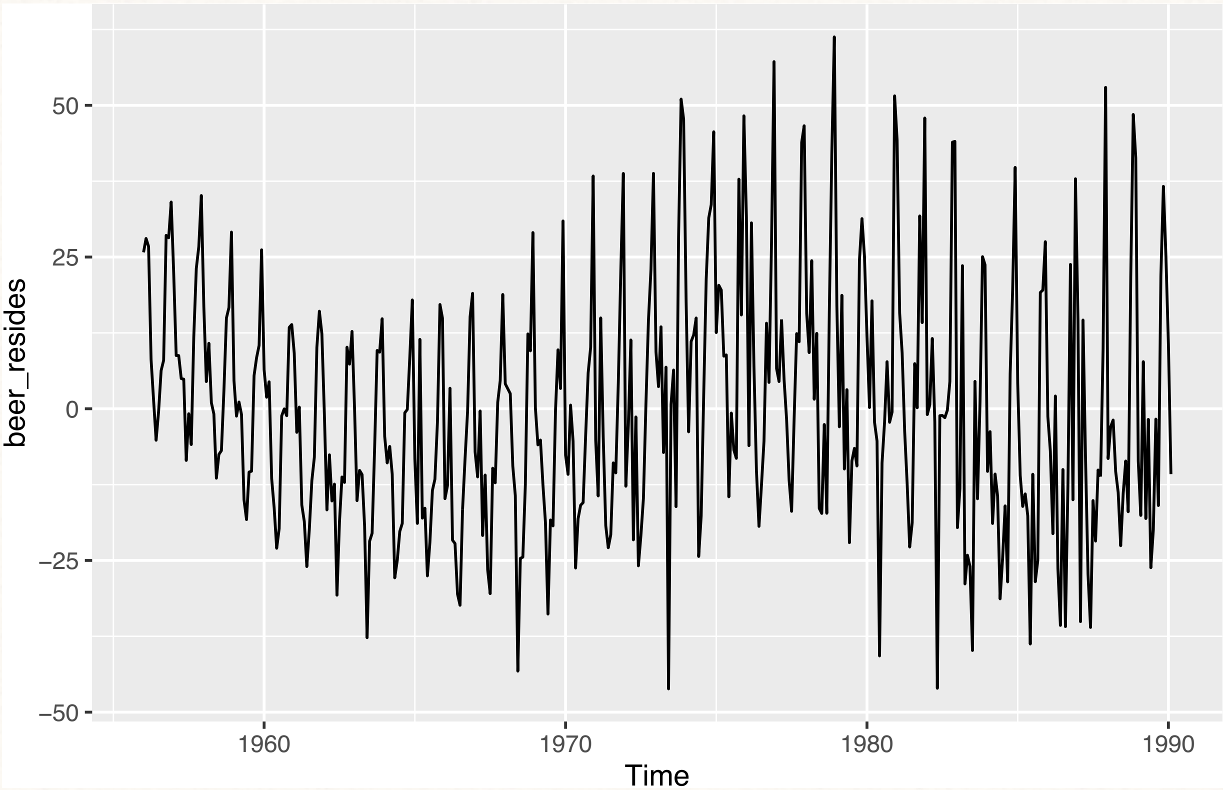
```
beer_linear<- tslm(beer~trend)
beer_quadratic<- tslm(beer~trend+I(trend^2))
beer_ma3<-ma(beer,order=3)
beer_ma5<-ma(beer,order=5)
#plot the fitted grapth
```

```
beer_with_fit<-cbind(beer,Linear_trend=fitted(beer_linear),Quadratic_trend=fitted(beer_quadratic),MA3=be
autoplot(beer_with_fit)
```

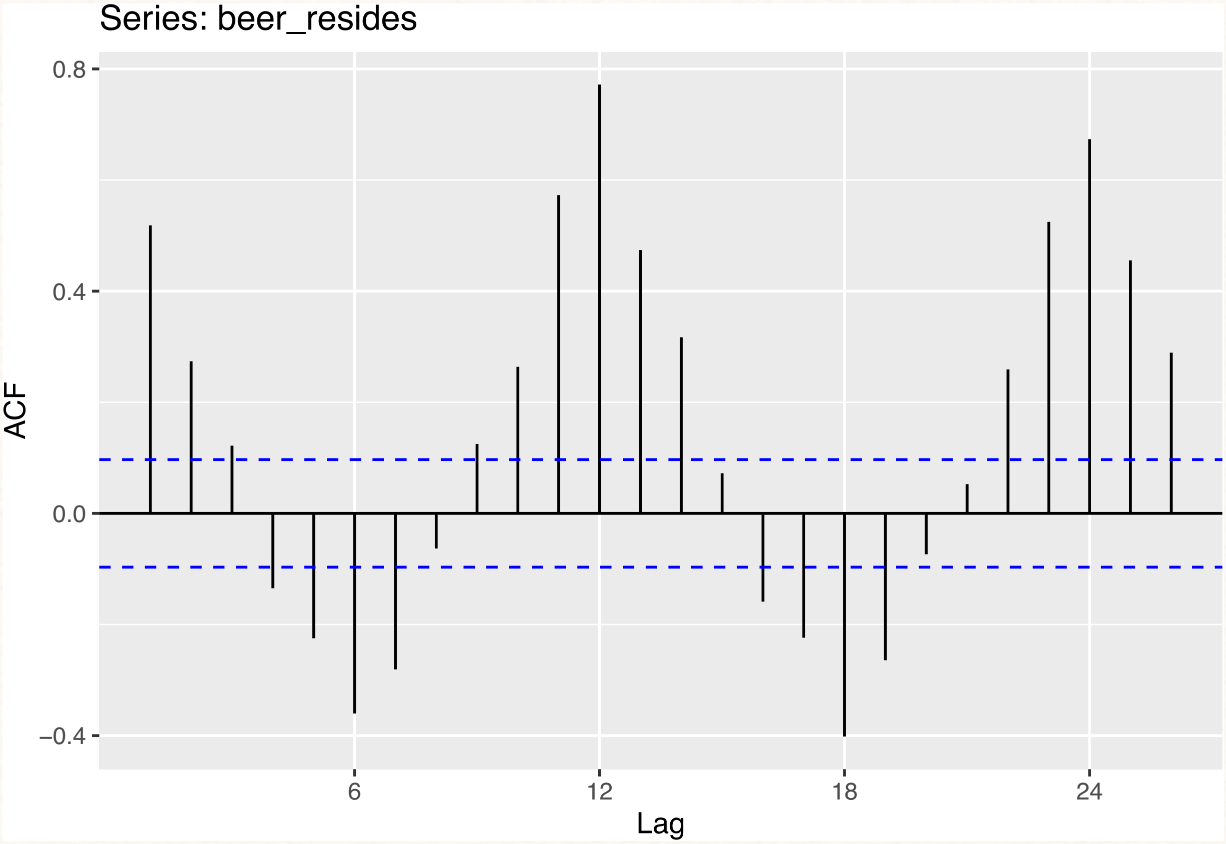


The residual plots by removing these the quadratic trends respectively from the dataset :

```
beer_resides<-cbind(
                    res_quad=beer-fitted(beer_quadratic))

autoplot(beer_resides,facet=TRUE)

## Warning: Ignoring unknown parameters: facet
```



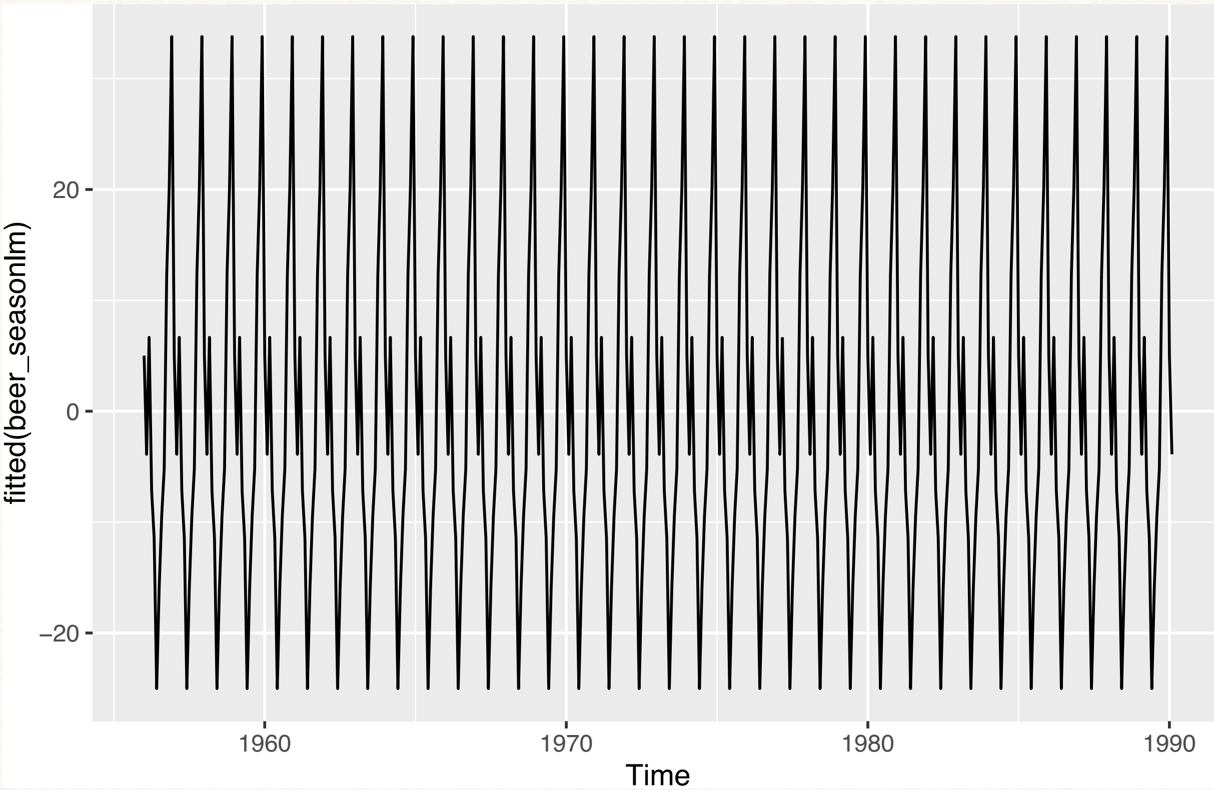

```
#autoplot(sales_resides,facet=TRUE)
ggAcf(beer_resides)
```



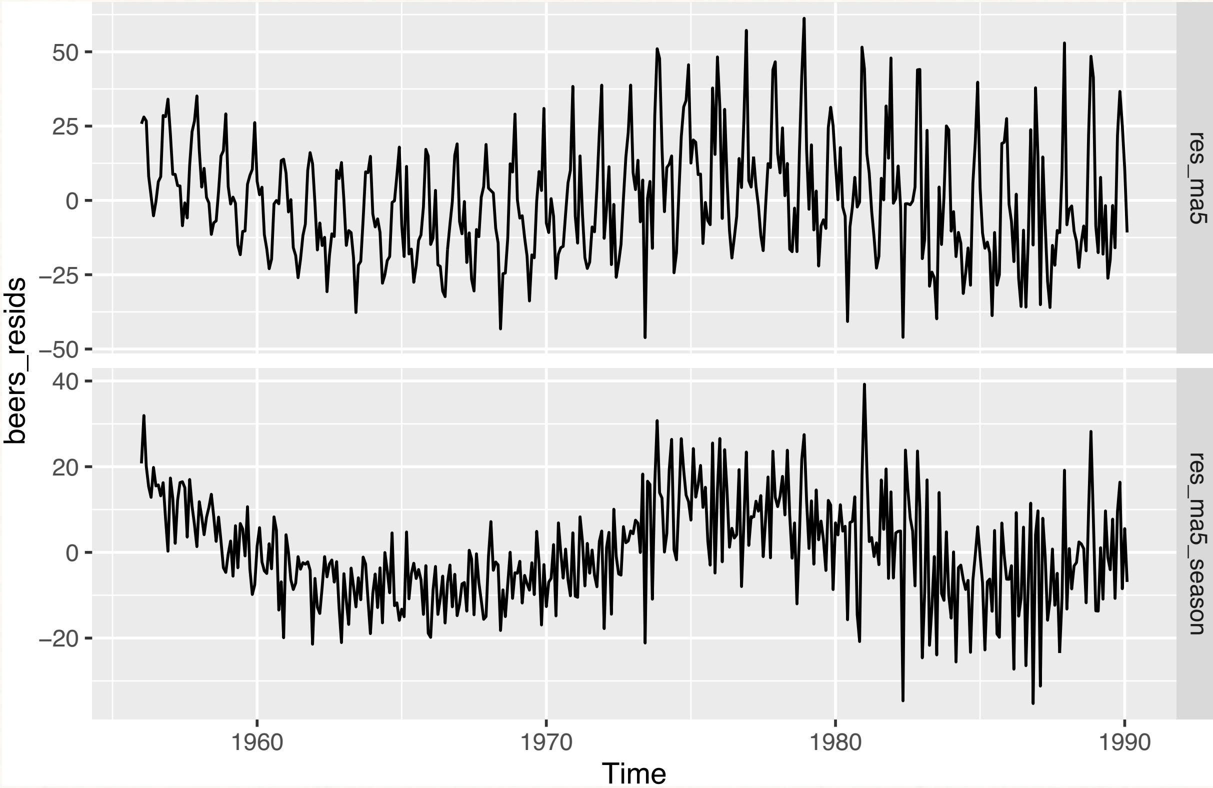
```r
frequency(beer_resides)
```

```
## [1] 12
```

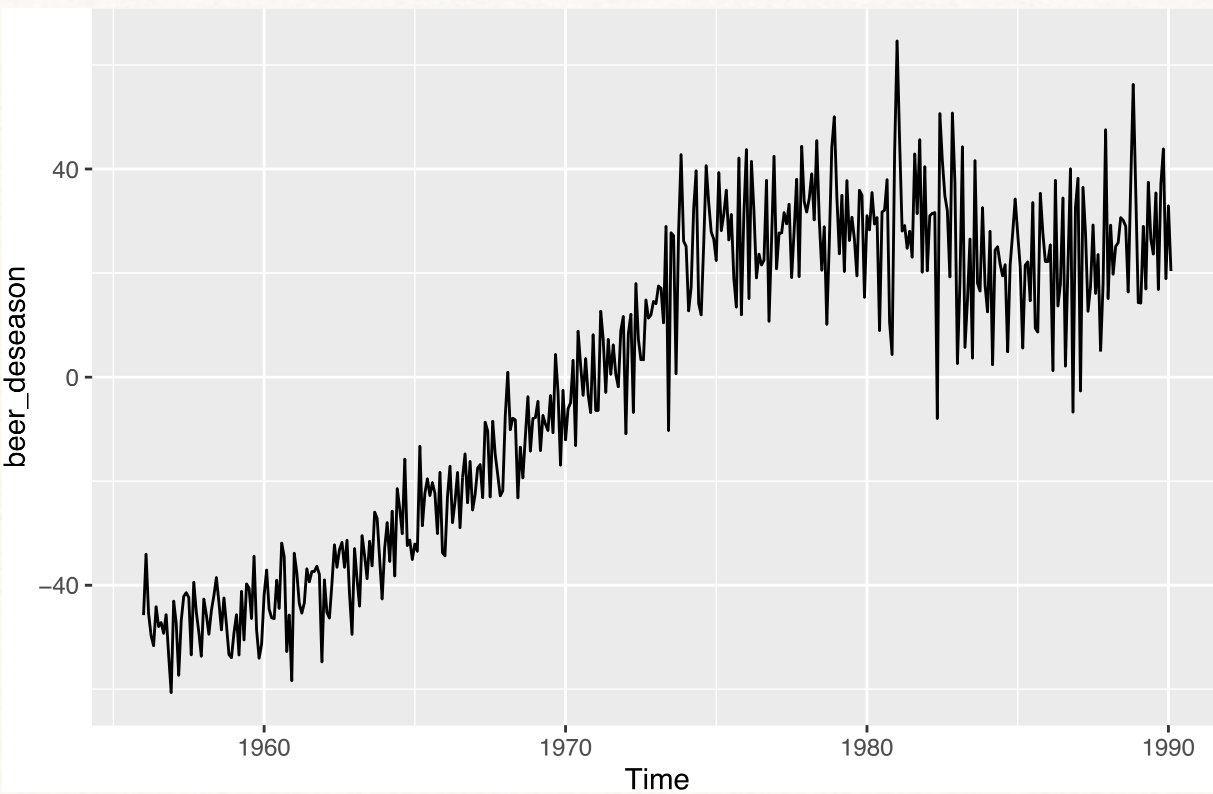
```r
beer_seasonlm<-tslm(beer_resides~season)
autoplot(fitted(beer_seasonlm))
```



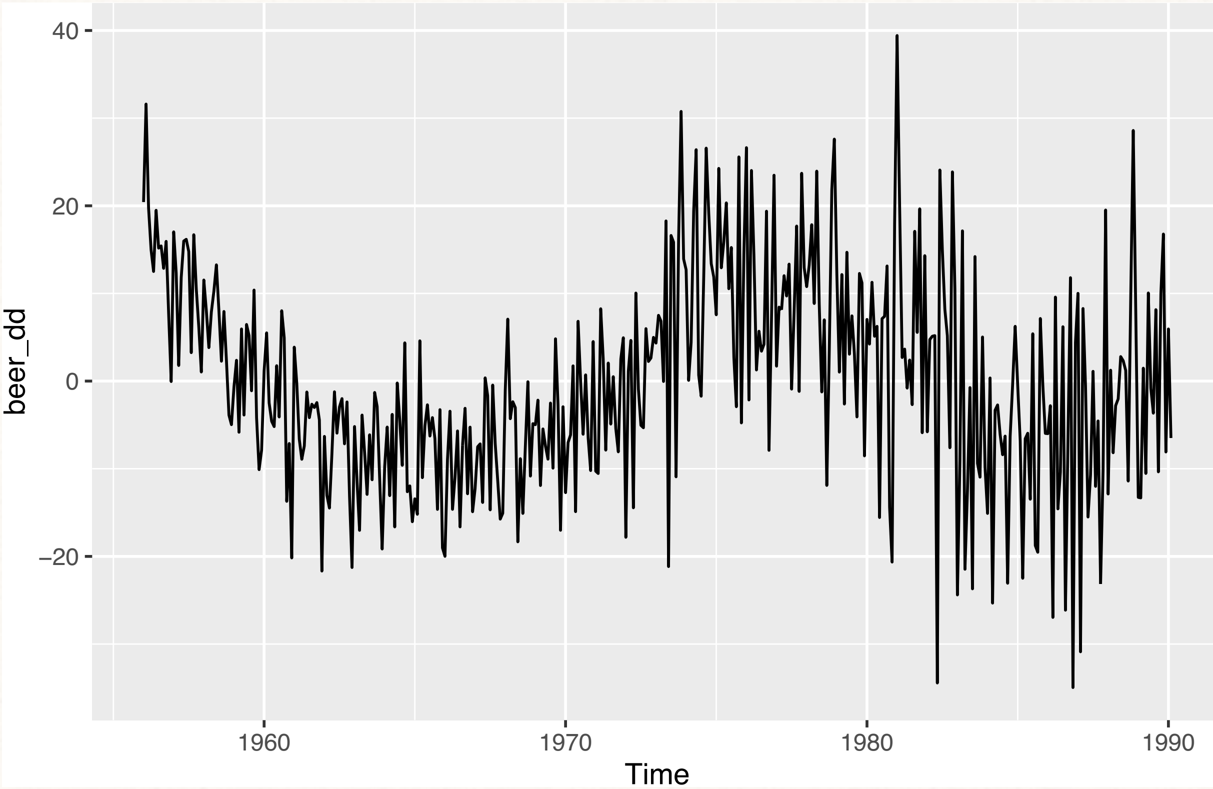

```r
beers_resids<-cbind(res_ma5=beer_resides,
                    res_ma5_season=beer_resides-fitted(beer_seasonlm))
autoplot(beers_resids,facet=TRUE)
```



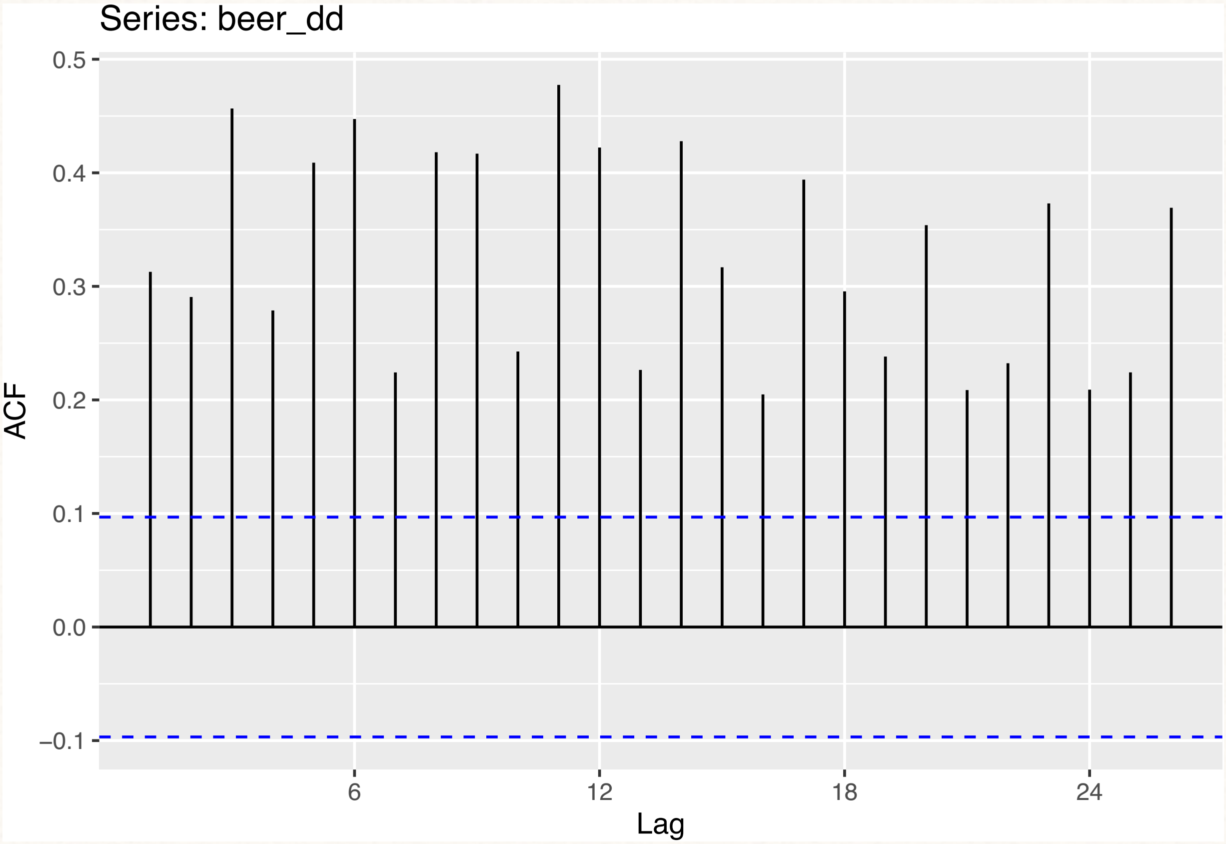

```r
beer_deseason<-beer-mean(beer)-fitted(beer_seasonlm)
autoplot(beer_deseason)
```



```r
beer_qua<- tslm(beer_deseason~trend+I(trend^2))
beer_dd<-beer_deseason-fitted(beer_qua)
autoplot(beer_dd)
```



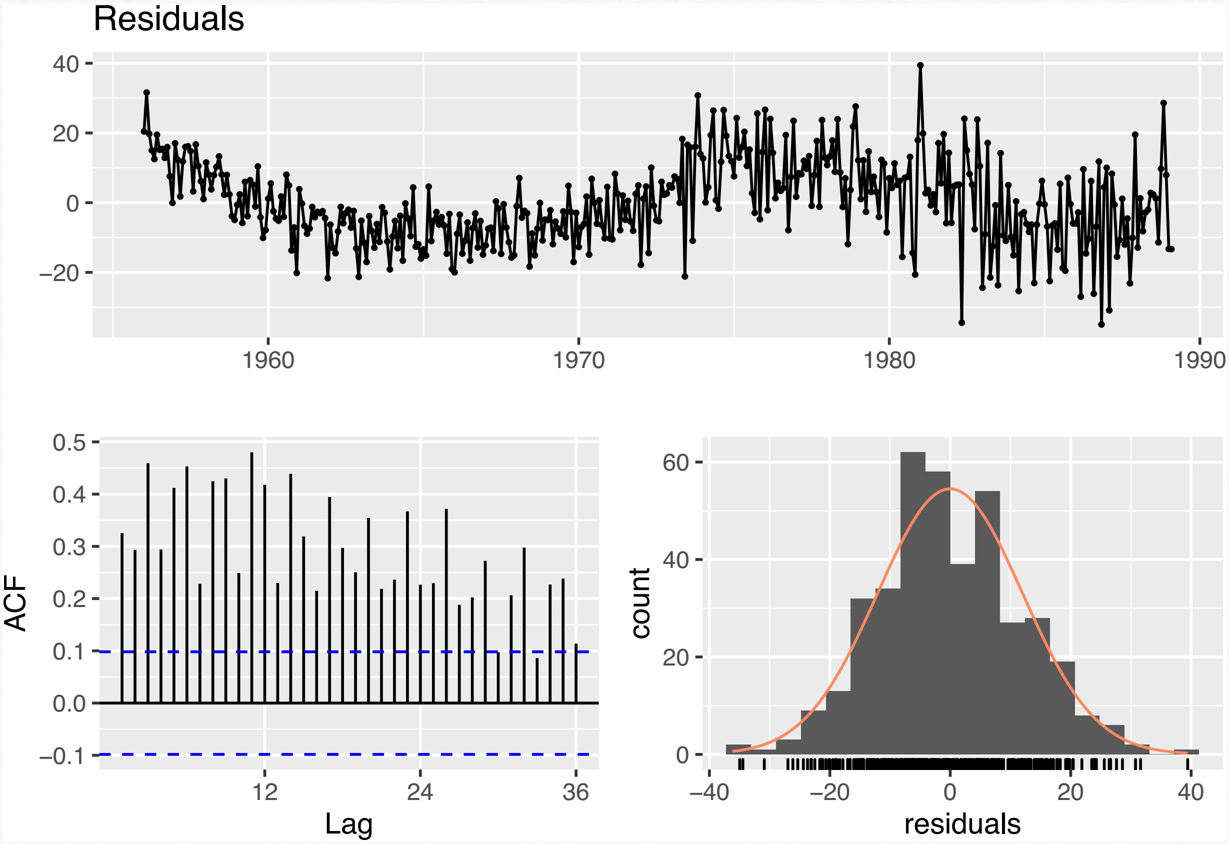

```r
ggAcf(beer_dd)
```



By the autoplot and residual plot of the data, it shows that the data may possibly be a white noise process, with low correlation.

```
beer_ddd<-head(beer_dd,-12)
checkresiduals(beer_ddd)
```

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



Try to do the ARIMA model again:

```
pro<-auto.arima(beer_ddd,stepwise=FALSE, seasonal=FALSE, ic='aic',trace=TRUE)
```

```
##
##  Fitting models using approximations to speed things up...
##
##  ARIMA(0,1,0)                          : 3215.129
##  ARIMA(0,1,0)            with drift    : 3217.114
##  ARIMA(0,1,1)                          : 2946.174
##  ARIMA(0,1,1)            with drift    : 2947.363
##  ARIMA(0,1,2)                          : 2939.357
##  ARIMA(0,1,2)            with drift    : 2940.609
##  ARIMA(0,1,3)                          : 2924.046
##  ARIMA(0,1,3)            with drift    : 2925.451
##  ARIMA(0,1,4)                          : 2925.712
##  ARIMA(0,1,4)            with drift    : 2927.091
##  ARIMA(0,1,5)                          : 2915.388
##  ARIMA(0,1,5)            with drift    : 2916.907
##  ARIMA(1,1,0)                          : 3111.811
##  ARIMA(1,1,0)            with drift    : 3113.748
##  ARIMA(1,1,1)                          : 2952.66
##  ARIMA(1,1,1)            with drift    : 2953.785
##  ARIMA(1,1,2)                          : 2953.807
##  ARIMA(1,1,2)            with drift    : 2954.976
##  ARIMA(1,1,3)                          : 2933.006
##  ARIMA(1,1,3)            with drift    : 2934.259
##  ARIMA(1,1,4)                          : 2933.831
```

```
##  ARIMA(1,1,4)            with drift         : 2935.139
##  ARIMA(2,1,0)                               : 3005.483
##  ARIMA(2,1,0)            with drift         : 3007.368
##  ARIMA(2,1,1)                               : 2929.203
##  ARIMA(2,1,1)            with drift         : 2930.578
##  ARIMA(2,1,2)                               : 2928.436
##  ARIMA(2,1,2)            with drift         : 2929.725
##  ARIMA(2,1,3)                               : 2910.104
##  ARIMA(2,1,3)            with drift         : 2912.099
##  ARIMA(3,1,0)                               : 2998.079
##  ARIMA(3,1,0)            with drift         : 2999.973
##  ARIMA(3,1,1)                               : 2928.147
##  ARIMA(3,1,1)            with drift         : 2929.609
##  ARIMA(3,1,2)                               : 2927.022
##  ARIMA(3,1,2)            with drift         : 2928.434
##  ARIMA(4,1,0)                               : 2959.86
##  ARIMA(4,1,0)            with drift         : 2961.755
##  ARIMA(4,1,1)                               : 2915.703
##  ARIMA(4,1,1)            with drift         : 2917.263
##  ARIMA(5,1,0)                               : 2934.629
##  ARIMA(5,1,0)            with drift         : 2936.461
##
##  Now re-fitting the best model(s) without approximations...
##
##
##
##
##  Best model: ARIMA(2,1,3)
```

```r
summary(pro)
```

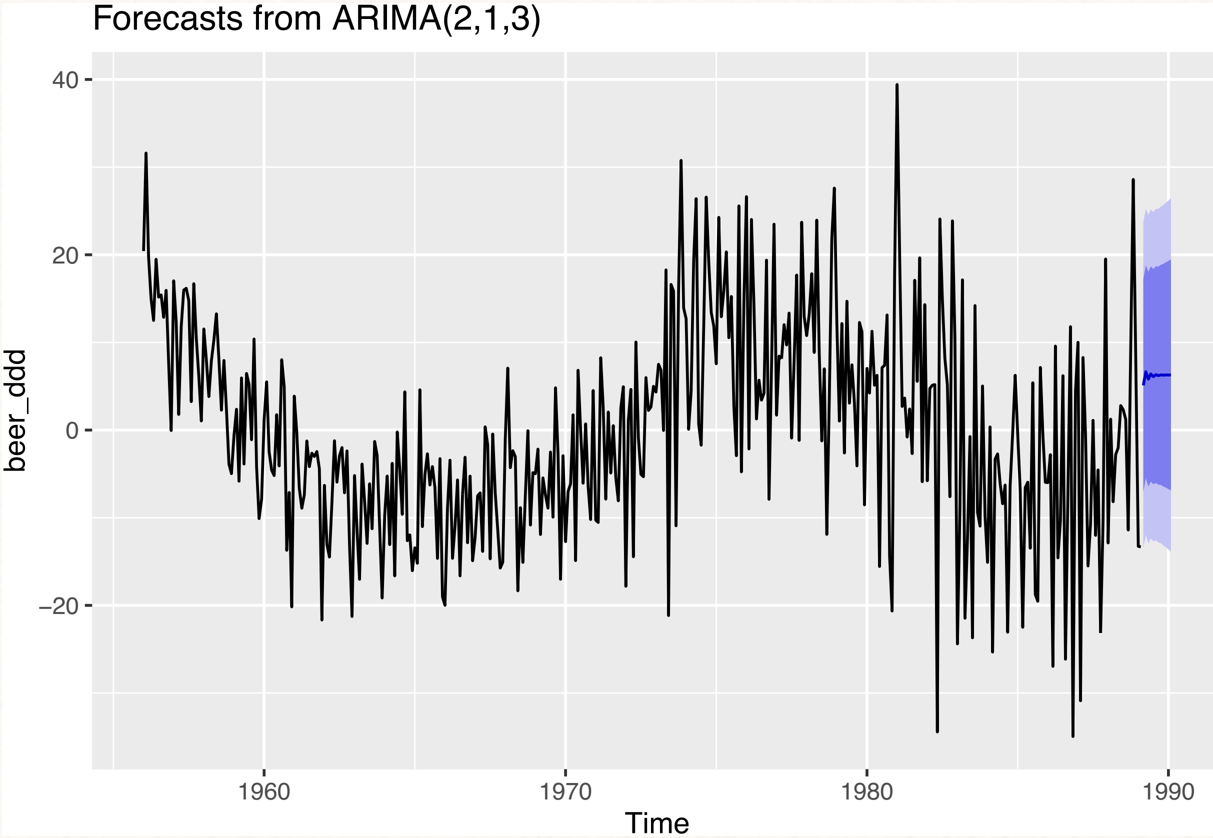
```
## Series: beer_ddd
## ARIMA(2,1,3)
##
## Coefficients:
##          ar1     ar2      ma1      ma2     ma3
##       0.0996  0.4361  -1.1228  -0.4402  0.6487
## s.e.  0.1378  0.1212   0.1208   0.2098  0.1093
##
## sigma^2 estimated as 89.82:  log likelihood=-1455.05
## AIC=2922.11   AICc=2922.33   BIC=2946.01
##
## Training set error measures:
##                     ME     RMSE     MAE      MPE    MAPE      MASE        ACF1
## Training set -0.202667 9.405641 7.28973 50.03155 318.183 0.7778955 -0.03251424
```

```r
confint(pro)
```

```
##          2.5 %      97.5 %
## ar1 -0.1705255  0.36979818
## ar2  0.1985049  0.67371573
## ma1 -1.3595752 -0.88611656
## ma2 -0.8514258 -0.02906865
## ma3  0.4344729  0.86291610
```

Here the 95% confidence interval shows the bounds od parameter.

```r
# forcast the 12-step data
foo<-forecast(pro,h=12)
autoplot(foo)
```



Now show the result by table

```r
#b1<-log(tail(beer_dd,12))
b1<-tail(beer_dd,12)
e1<-b1-foo$mean
d1<- data_frame(e1,foo$lower,foo$upper,foo$mean)
```

```
## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```

```r
colnames(d1)<-c('error','2.5%','97.5%','prediction')
d1
```

```
## # A tibble: 12 x 4
##      error `2.5%`[,"80%"] [,"95%"] `97.5%`[,"80%"] [,"95%"] prediction
##      <dbl>          <dbl>    <dbl>           <dbl>    <dbl>      <dbl>
##  1  -3.62           -7.05    -13.5            17.2     23.7       5.10
##  2 -17.2            -5.49    -11.9            18.8     25.2       6.65
##  3   4.24           -6.44    -12.9            18.1     24.5       5.81
##  4  -7.21           -5.87    -12.4            18.7     25.2       6.40
##  5  -9.76           -6.19    -12.7            18.4     24.9       6.09
##  6   1.83           -6.04    -12.6            18.7     25.2       6.32
##  7 -16.6            -6.23    -12.8            18.6     25.2       6.21
##  8   3.69           -6.27    -12.9            18.9     25.5       6.30
##  9  10.5            -6.44    -13.2            19.0     25.7       6.26
```

```
## 10 -14.4              -6.56    -13.4             19.1     26.0        6.29
## 11  -0.328            -6.73    -13.6             19.3     26.2        6.28
## 12 -12.8              -6.89    -13.9             19.5     26.5        6.29
```

It is obvious that comparing with part b, the error od prediction become much smaller.