Assignment3_answers

Hair Parra

April 16, 2020

Hair ALbeiro Parra Barrera

260738619

MATH 545: ASSG3

Assignment 3

Libraries

```r
library(tidyverse)
library(tidyquant)
library(gridExtra)
library(tibbletime)
library(forecast)
library(itsmr)
library(here)
library(bbmle)
library(tseries)
library(fpp2)
knitr::opts_chunk$set(comment=NA,tidy=FALSE)
#library(future) Not needed yet
#library(doFuture) Not needed yet
#library(rbenchmark) Not needed yet
```

Question 3.9

(a) Calculate the autocovarinace function $\gamma(.)$ of the stationary time series:

$$Y_t = \mu_t + Z_t + \theta_1 Z_1 + \theta_{12} Z_{12}, \quad \{Z_t\} \sim WN(0, \sigma^2)$$

Solution

In the hand-written part , we got that

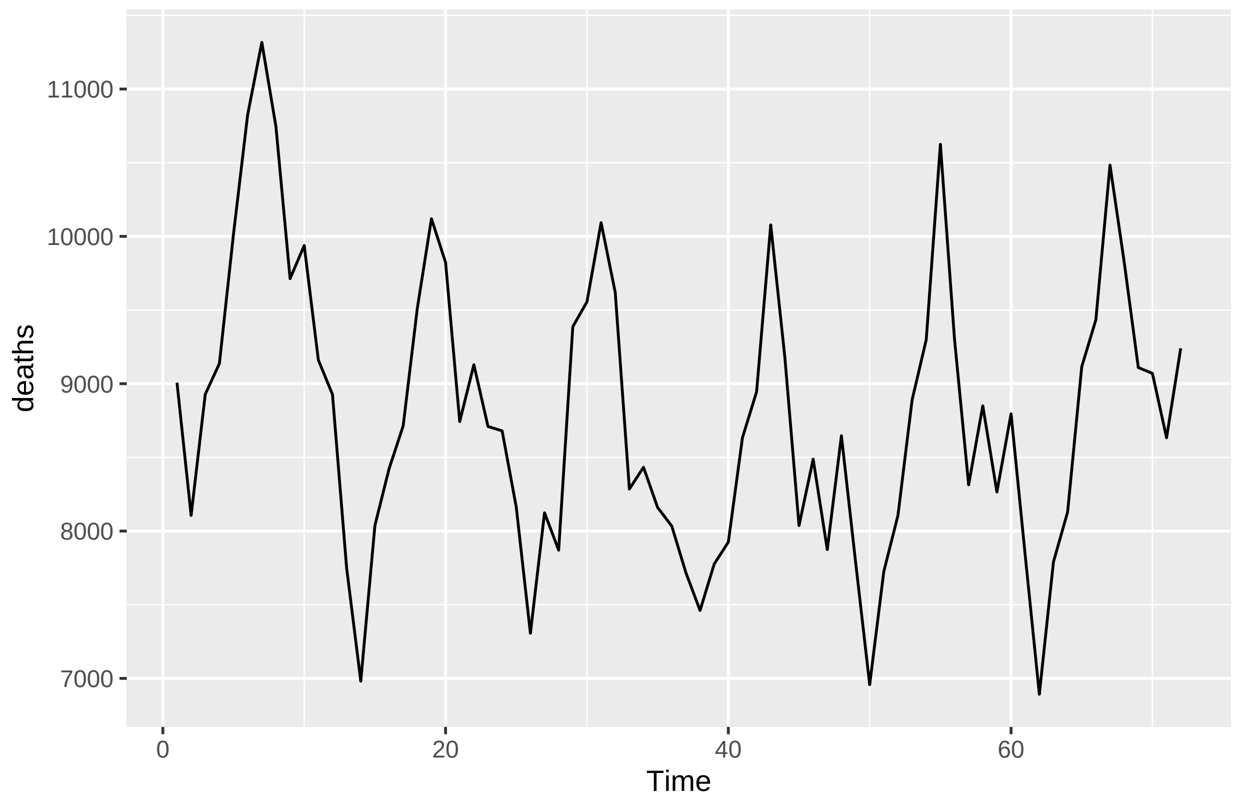
$$\gamma_Y(h) = \begin{cases} \sigma^2(1+\theta_1^2+\theta_{12}^2), & h=0 \\ \sigma^2\theta_1, & h=1 \\ \sigma^2\theta_1\theta_{12}, & h=11 \\ \sigma^2\theta_{12}, & h=12 \end{cases}$$

(b) Compte the sample mean and sample autocovariances $\hat{\gamma}(h), 0, \leq h \leq 20$ of $\{\nabla\nabla_{12}X_t\}$, where $\{X_t, t = 1, \ldots, 72\}$ is the accidental deaths of the sereis DEATHS

```
library(itsmr)
deaths <- ts(deaths) # convert into a time series
deaths
```

```
Time Series:
Start = 1
End = 72
Frequency = 1
 [1]  9007  8106  8928  9137 10017 10826 11317 10744  9713  9938  9161  8927
[13]  7750  6981  8038  8422  8714  9512 10120  9823  8743  9129  8710  8680
[25]  8162  7306  8124  7870  9387  9556 10093  9620  8285  8433  8160  8034
[37]  7717  7461  7776  7925  8634  8945 10078  9179  8037  8488  7874  8647
[49]  7792  6957  7726  8106  8890  9299 10625  9302  8314  8850  8265  8796
[61]  7836  6892  7791  8129  9115  9434 10484  9827  9110  9070  8633  9240
```

```
# Plot the data for visualization
autoplot(deaths)
```



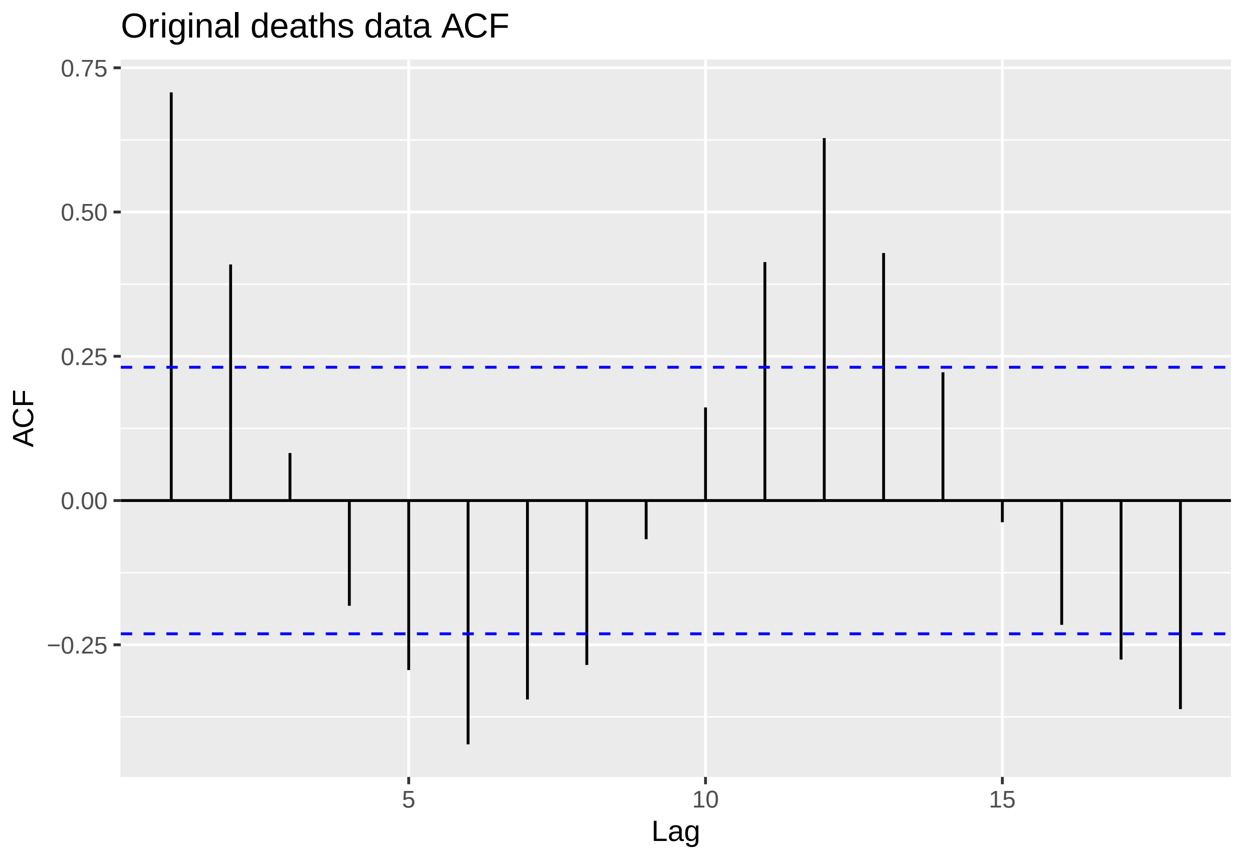
We can calculate the sample mean before differencing:

```
x_bar <- mean(deaths)
x_bar
```

```
[1] 8787.736
```

So $\hat{mu} = \bar{x}_{72} = 8787.736$.

```
# Original data ACF
deaths_acf <- ggAcf(deaths) + ggtitle("Original deaths data ACF")
deaths_acf
```



```
orig_sample_ACF_vals <- deaths_acf$data  # extract the data object
orig_sample_ACF_vals <- c(1,orig_sample_ACF_vals$Freq) # extract autocorrelations
orig_sample_ACF_vals <- c(orig_sample_ACF_vals, orig_sample_ACF_vals[1:2])  # repeat some of these
orig_sample_ACF_vals
```

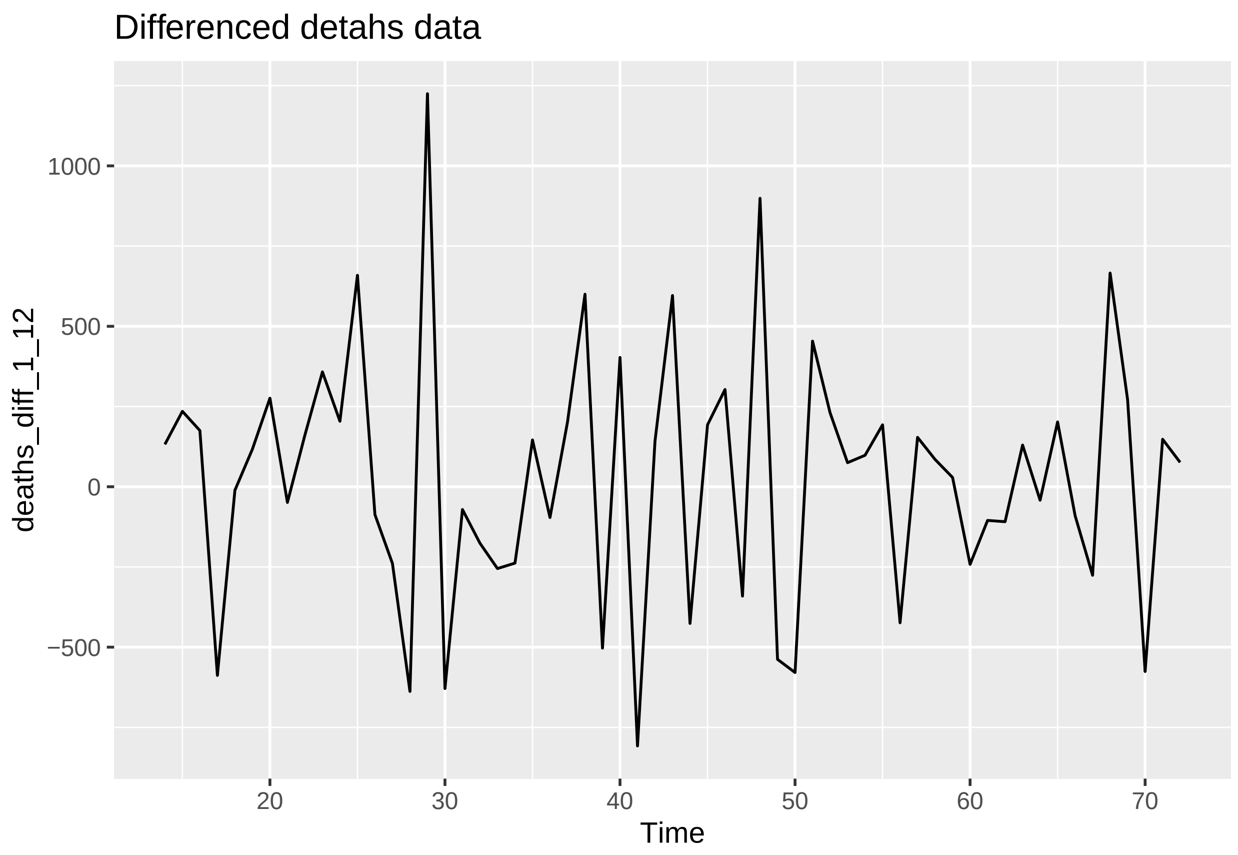
```
 [1]  1.00000000  0.70739498  0.40914118  0.08232642 -0.18240432 -0.29389209
 [7] -0.42256255 -0.34478440 -0.28495416 -0.06704366  0.16137269  0.41325445
[13]  0.62817012  0.42898283  0.22227129 -0.03760829 -0.21542724 -0.27571757
[19] -0.36154679  1.00000000  0.70739498
```

We will first difference to obtain the series of interest $\{\nabla\nabla_{12}X_t\}$:

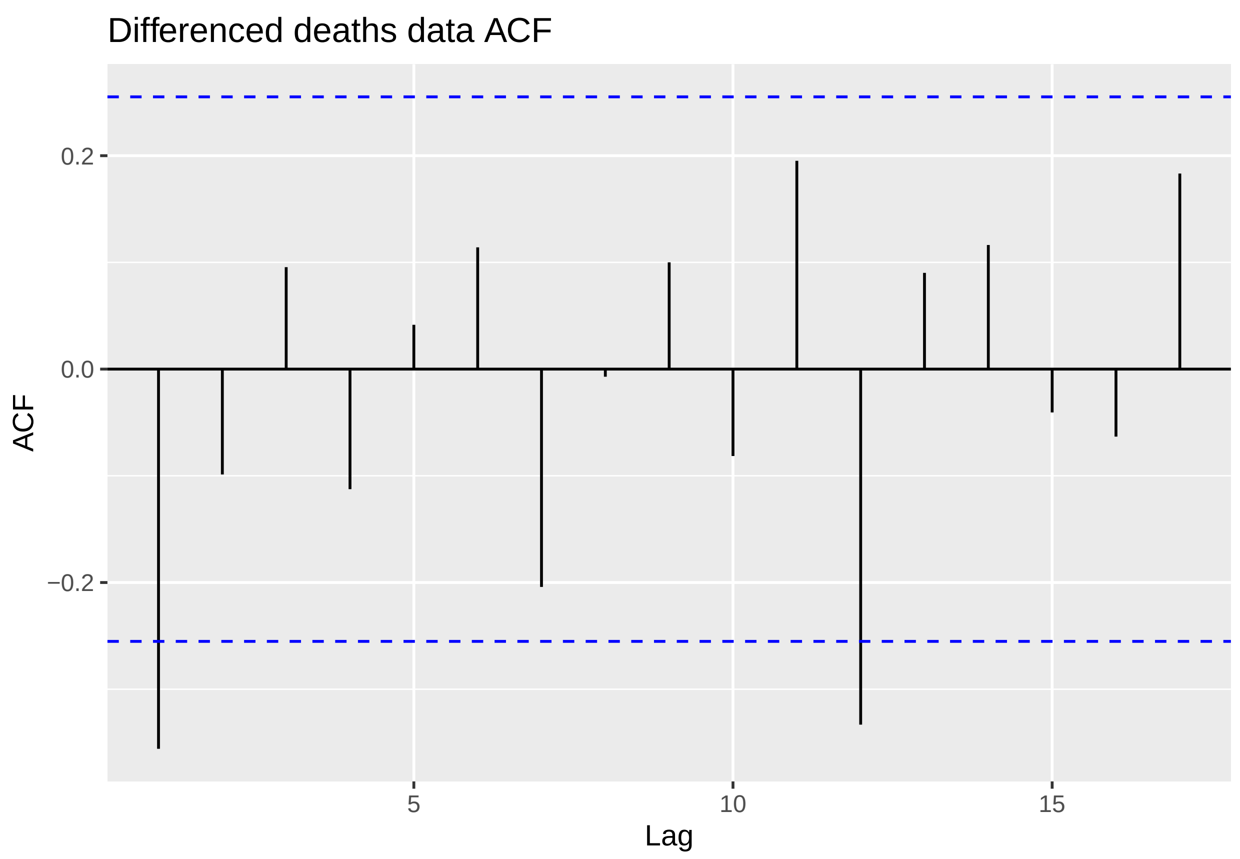
```
deaths_diff_12 <- diff(deaths, 12) # difference of order 12
deaths_diff_1_12 <- diff(deaths_diff_12, 1) # diff again
deaths_diff_1_12 # Note we lost some data
```

```
Time Series:
Start = 14
End = 72
Frequency = 1
 [1]  132  235  175 -588  -11  117  276  -49  161  358  204  659  -87 -239 -638
[16] 1225 -629  -71 -176 -255 -238  146  -96  201  600 -503  403 -808  142  596
[31] -426  193  303 -341  899 -538 -579  454  231   75   98  193 -424  154   85
[46]   29 -242 -105 -109  130  -42  202  -90 -276  666  271 -576  148   76
```

```
autoplot(deaths_diff_1_12) + ggtitle("Differenced detahs data")
```



```
deaths_diff_1_12_ACF = ggAcf(deaths_diff_1_12) + ggtitle("Differenced deaths data ACF")
deaths_diff_1_12_ACF
```



We can then extract sample autocovariances $\hat{\gamma}(h), 0, \leq h \leq 20$

```
sample_ACF_vals <- deaths_acf$data  # extract the data object
sample_ACF_vals <- c(sample_ACF_vals$Freq) # extract autocorrelations
sample_ACF_vals <- c(sample_ACF_vals,  sample_ACF_vals)  # repeat some of these
sample_ACF_vals
```

```
 [1]  0.70739498  0.40914118  0.08232642 -0.18240432 -0.29389209 -0.42256255
 [7] -0.34478440 -0.28495416 -0.06704366  0.16137269  0.41325445  0.62817012
[13]  0.42898283  0.22227129 -0.03760829 -0.21542724 -0.27571757 -0.36154679
[19]  0.70739498  0.40914118  0.08232642 -0.18240432 -0.29389209 -0.42256255
[25] -0.34478440 -0.28495416 -0.06704366  0.16137269  0.41325445  0.62817012
[31]  0.42898283  0.22227129 -0.03760829 -0.21542724 -0.27571757 -0.36154679
```

Note that due to the differencing, we can only obtain $\hat{\gamma}(h), 0, \leq h \leq 18$, after which if the series is stationary, means that $\gamma(h) = \gamma(h, h+t)$, so that it simply gets repeated.

(c) By equating $\hat{\gamma}(1), \hat{\gamma}(11)$ and $\hat{\gamma}(12)$ from part 3 with $\gamma(1), \gamma(2)$ and $\gamma(12)$ respectively from part (a), find a model of the form defined in (a) to represent $\{\nabla\nabla_{12}X_t\}$

- **NOTE:** Here I really wasn't sure which autocovariances I am supposed to take. From the assg, question 3.9, (a) we are asked to find the theoretical form of the autocovariances. Then, in (c) we are asked to compare these to the ACF of the **differenced** series, which doesn't make much sense? Instead, what makes more sense to me is to use the estimated ACF from the **original series** and compare these to the theoretical ACFs in part (a). Then, solve for θ_1, θ_{12}, and use these in the series $\nabla\nabla_{12}X_t$. So this is what I've done here. If it's not what was supposed, then I should have used the values from the ACF of the latter series (again, doesn't make much sense to me). If that's the case, I should still get full points as this question was not clear.

So now we will extract the values of interest, $\hat{\gamma}(1), \hat{\gamma}(11)$ and $\hat{\gamma}(12)$ from the original series.

```
gamma1 <- sample_ACF_vals[1]
gamma11 <- sample_ACF_vals[11]
gamma12 <- sample_ACF_vals[12]
gammas_hat <- c(gamma1,gamma11,gamma12)
names(gammas_hat) <- c("gamma1","gamma11","gamma12")
gammas_hat
```

```
   gamma1   gamma11   gamma12
0.7073950 0.4132545 0.6281701
```

Now, from (a), we have get the system of equations

$$\begin{cases} \sigma^2\theta_1 & = \hat{\gamma}_Y(1) \\ \sigma^2\theta_1\theta_{12} & = \hat{\gamma}_Y(11) \\ \sigma^2\theta_{12} & = \hat{\gamma}_Y(12) \end{cases}$$

So solving for the parameters we obtain

$$\begin{cases} \hat{\theta}_1 & = \dfrac{\hat{\gamma}_Y(1)}{\hat{\sigma}^2} = 0.6578705 \\ \hat{\theta}_{12} & = \dfrac{\hat{\gamma}_Y(12)}{\hat{\sigma}^2} = 0.5841921 \\ \hat{\sigma}^2 & = \dfrac{\hat{\gamma}_Y(1)\hat{\gamma}_Y(12)}{\hat{\gamma}_Y(11)} = 1.07528 \end{cases}$$

So the differenced series (after some calculations) has the form

$$\nabla_{12}X_{12} = \hat{\theta}_1(Z_{t-1} - Z_{t-13}) + \hat{\theta}_{12}(Z_{t-12} - Z_{t-24})$$
$$\nabla\nabla_{12}X_{12} = \hat{\theta}_1(Z_{t-1} - Z_{t-2} - Z_{t-13} + Z_{t-14}) + \hat{\theta}_{12}(Z_{t-12} - Z_{t-13} - Z_{t-24} + Z_{t-25})$$

Final question

(a) Remove the last 12 values from the `beer` data

(b) Find an ARIMA model for the *logarithms* of the beerdata. Your analysis should include:

i) A logical explanation of the steps taken to choose the final model.
ii) Appropriate 95% bounds for the componenets of ϕ and θ.
iii) AN examination of the residuals to check for similarity of a white noise process.
iv) A graph of the series showing forecasts for the removed 12 values and 95% prediction bounds.
v) A table of the actual forecast errors, i.e. observed - predicted, for the removed 12 values

(c) Repeat the steps in part (b), but instead use a classical decomposition approach by deseasonalizing, subtracting a quadratic trend, and then fitting and ARMA model to the reisudals. Then compare your forecast errors to those in part (b).

Preprocessing the data

We will first load the `beer` dataset

```r
# Load the data into a ts object
beer_dat = dget("beer.Rput")
str(beer_dat)
```

```
 Time-Series [1:422] from 1956 to 1991: 93.2 96 95.2 77.1 70.9 ...
```

Now we will consider the **logarithm** transformation of the data, which will be the focus of the analysis, then split the data into training and testing datasets:

```r
# Preprocess data
beer_dat <-log(beer_dat) # logarithm transformation
beer_dat_train <- window(beer_dat, end=1990.1) # train dataset
beer_dat_test <- window(beer_dat, start=1990.1) # test dataset
str(beer_dat_train)
```

```
 Time-Series [1:410] from 1956 to 1990: 4.53 4.56 4.56 4.35 4.26 ...
```

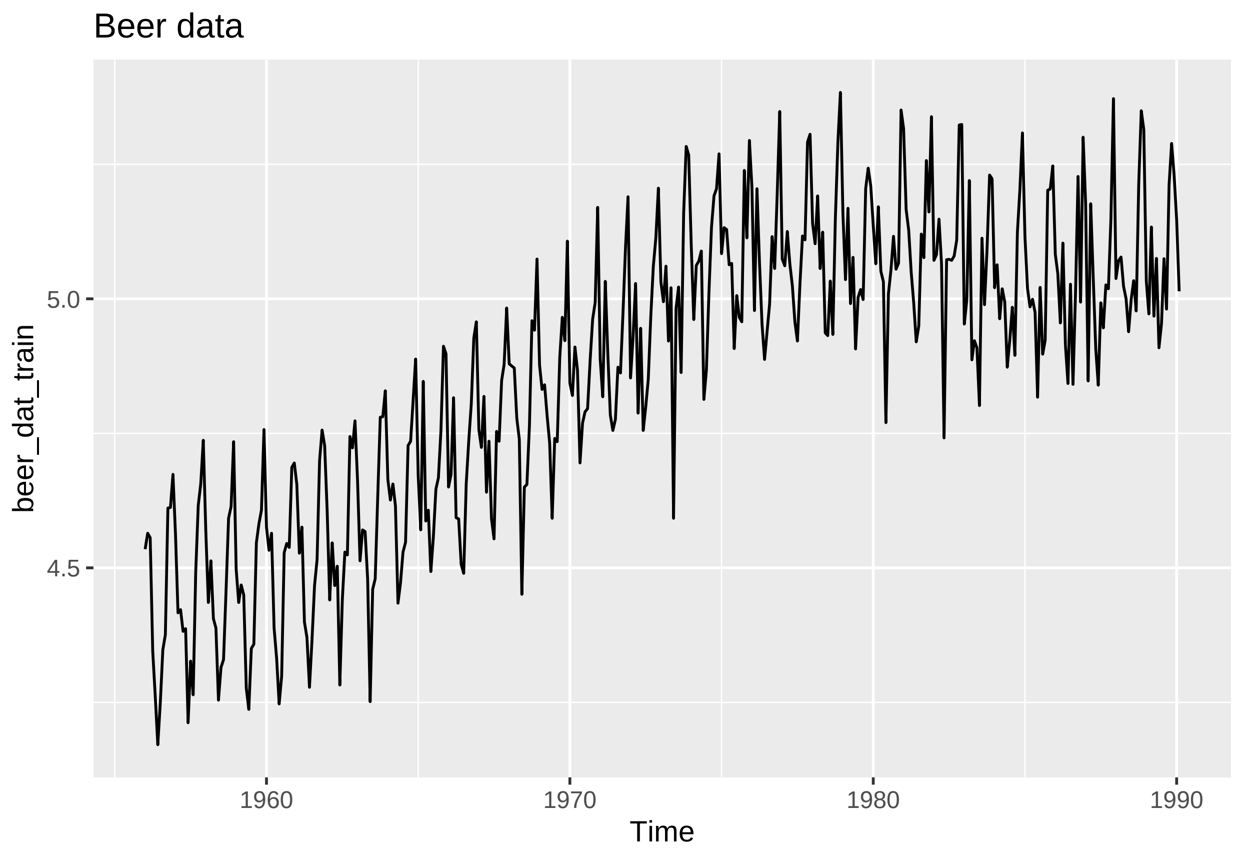
```r
str(beer_dat_test)
```

```
 Time-Series [1:12] from 1990 to 1991: 5.09 5.03 5.03 4.91 5 ...
```

Inspecting the data

Let's first take a look at the data

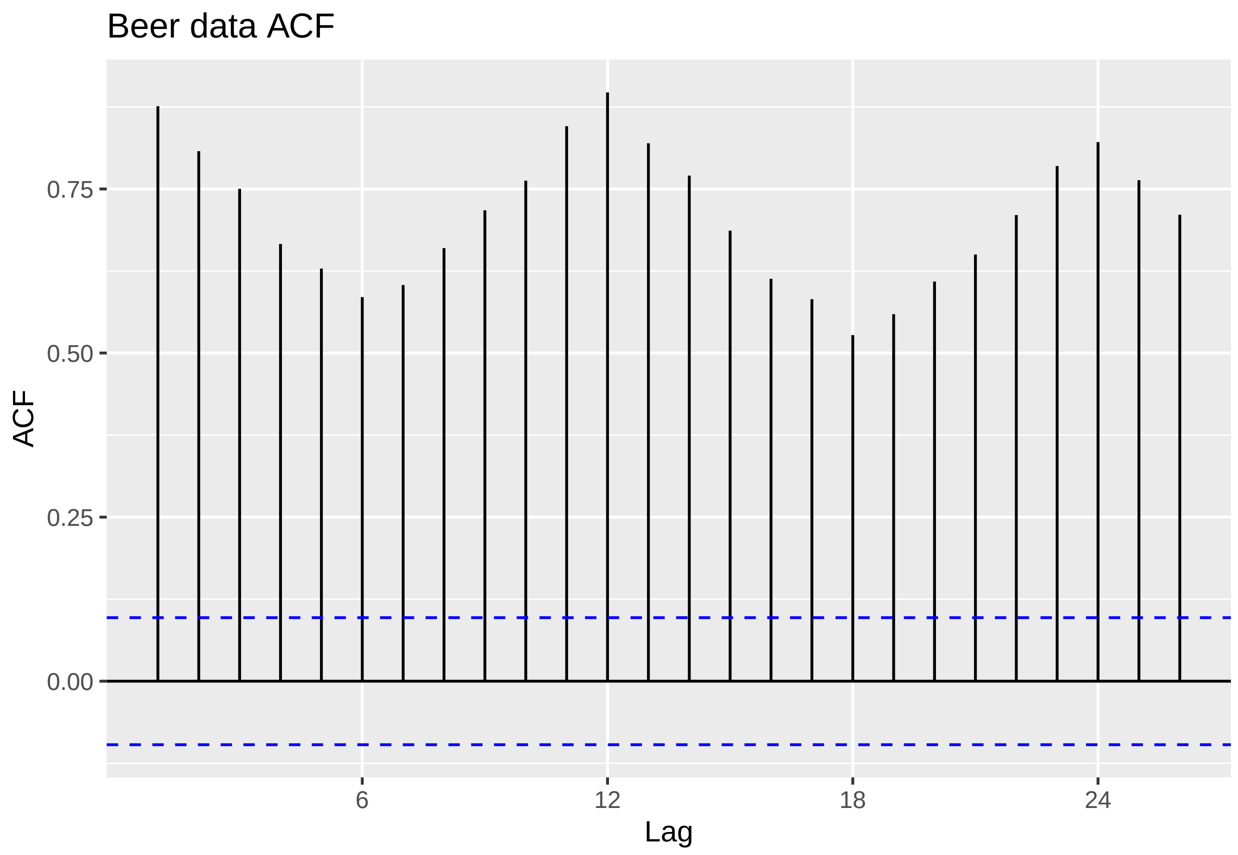
```r
# Plot series
autoplot(beer_dat_train) + ggtitle("Beer data")
```



The data definitely does not look stationary. and some periodicity is also visible.

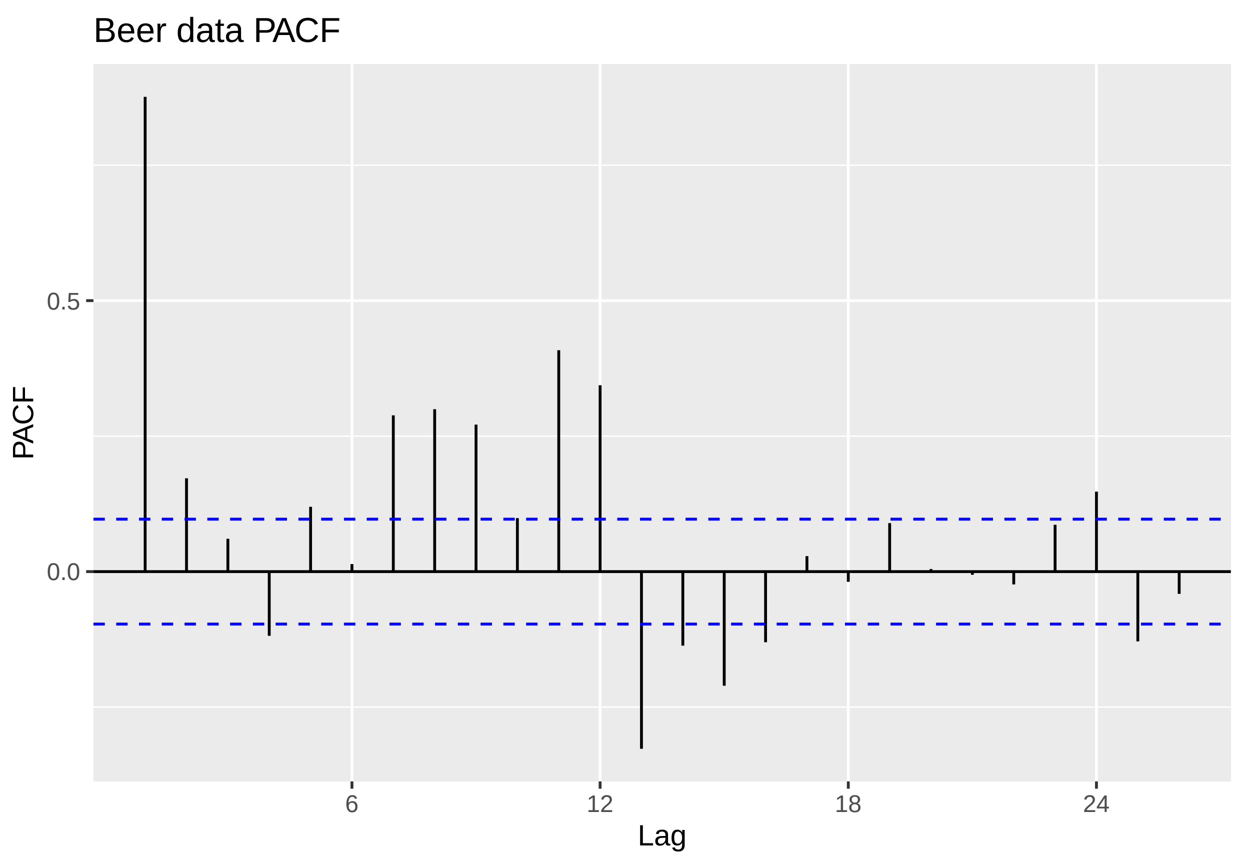
Let’s take a looka the autocorrelation function:

```r
# Plot ACF
ggAcf(beer_dat_train) + ggtitle("Beer data ACF")
```



Indeed, points at all lags are terribly correlated, and seasonality can be seen. This clearly tells this is definitely not a white noise $WN(0, \sigma^2)$ process.

```
# Plot PACF
ggPacf(beer_dat_train)  + ggtitle("Beer data PACF")
```



However from the PACF plot, we observe that the correlation seems to decrease at further lags.

Fitting the ARIMA model

- Let $d \in \mathbb{Z}$. $\{X_t\}$ is said to be ARIMA(p,d,q) iff $Y_t := (1-B)^d X_t$ is a causal ARMA(p,q) process.

We will now try to fit the best ARIMA model we can find to the data, using the `arima.sim` function (i.e. no ic selected)

```r
# Fit an auto-arima model
beerdat_arima = auto.arima(beer_dat_train,
                           seasonal=FALSE,
                           stepwise=FALSE,
                           ic = c("aicc", "aic", "bic") ,
                           approximation=FALSE,
                           trace=TRUE)
```

```
 ARIMA(0,1,0)                               : -472.2122
 ARIMA(0,1,0)            with drift         : -470.223
 ARIMA(0,1,1)                               : -495.6891
 ARIMA(0,1,1)            with drift         : -493.7302
 ARIMA(0,1,2)                               : -534.0083
 ARIMA(0,1,2)            with drift         : -539.507
 ARIMA(0,1,3)                               : -544.2853
 ARIMA(0,1,3)            with drift         : -549.5625
 ARIMA(0,1,4)                               : -570.0043
 ARIMA(0,1,4)            with drift         : -574.6963
 ARIMA(0,1,5)                               : -569.6033
 ARIMA(0,1,5)            with drift         : -573.9958
 ARIMA(1,1,0)                               : -492.1827
 ARIMA(1,1,0)            with drift         : -490.2058
 ARIMA(1,1,1)                               : -555.8088
 ARIMA(1,1,1)            with drift         : -560.3982
 ARIMA(1,1,2)                               : -553.7811
 ARIMA(1,1,2)            with drift         : -558.3878
 ARIMA(1,1,3)                               : -554.0459
 ARIMA(1,1,3)            with drift         : -558.6354
 ARIMA(1,1,4)                               : -570.0013
 ARIMA(1,1,4)            with drift         : -574.3103
 ARIMA(2,1,0)                               : -494.7938
 ARIMA(2,1,0)            with drift         : -492.8235
 ARIMA(2,1,1)                               : -503.6188
 ARIMA(2,1,1)            with drift         : -501.6311
 ARIMA(2,1,2)                               : -556.337
 ARIMA(2,1,2)            with drift         : -561.0268
 ARIMA(2,1,3)                               : -554.9761
 ARIMA(2,1,3)            with drift         : -559.5321
 ARIMA(3,1,0)                               : -494.544
 ARIMA(3,1,0)            with drift         : -492.5492
 ARIMA(3,1,1)                               : -502.1042
 ARIMA(3,1,1)            with drift         : -500.1003
 ARIMA(3,1,2)                               : -555.9742
 ARIMA(3,1,2)            with drift         : -560.3795
 ARIMA(4,1,0)                               : -504.1889
 ARIMA(4,1,0)            with drift         : -502.2292
 ARIMA(4,1,1)                               : -603.5335
 ARIMA(4,1,1)            with drift         : -610.4952
 ARIMA(5,1,0)                               : -503.2939
 ARIMA(5,1,0)            with drift         : -501.3409



 Best model: ARIMA(4,1,1)            with drift
```

The fitting method employed is conditional-sum-of-squares to find starting values, then maximum likelihood. In this case, as the `ic` (information criterion) was not specified, the algorithm performed model selection according to all of the three criteria: AIC, AICC and BIC. The values displayed on the right-most column are $-2\ell(x;\underline{\hat{\phi}},\underline{\hat{\theta}},\hat{\sigma^2})$ for fitting the different models. Whenever we obtained `Inf`. It means thar the model was so terrible that it could not find appropriate initial values, and it diverged.

Inspecting the chosen ARIMA(4,1,1) model

Now let's inspect our model:

```
beerdat_arima_411 <- beerdat_arima
beerdat_arima_411
```

```
Series: beer_dat_train
ARIMA(4,1,1) with drift

Coefficients:
         ar1     ar2     ar3      ar4      ma1   drift
      0.4655  0.0537  0.0512  -0.3486  -0.9360  0.0017
s.e.  0.0470  0.0517  0.0516   0.0469   0.0126  0.0005

sigma^2 estimated as 0.01282:  log likelihood=312.39
AIC=-610.77   AICc=-610.5   BIC=-582.68
```

This says that we have a lag 1 difference giving an ARMA(3,1) model.From this, we see all the estimated parameters $\hat{\phi}_1, \hat{\phi}_2, \hat{\phi}_3, \hat{\phi}_4, \hat{\theta}_1$ and $\hat{\sigma}^2$. (Note how little variance there is!)

Parameters

We can construct 95% confidence intervals for these:

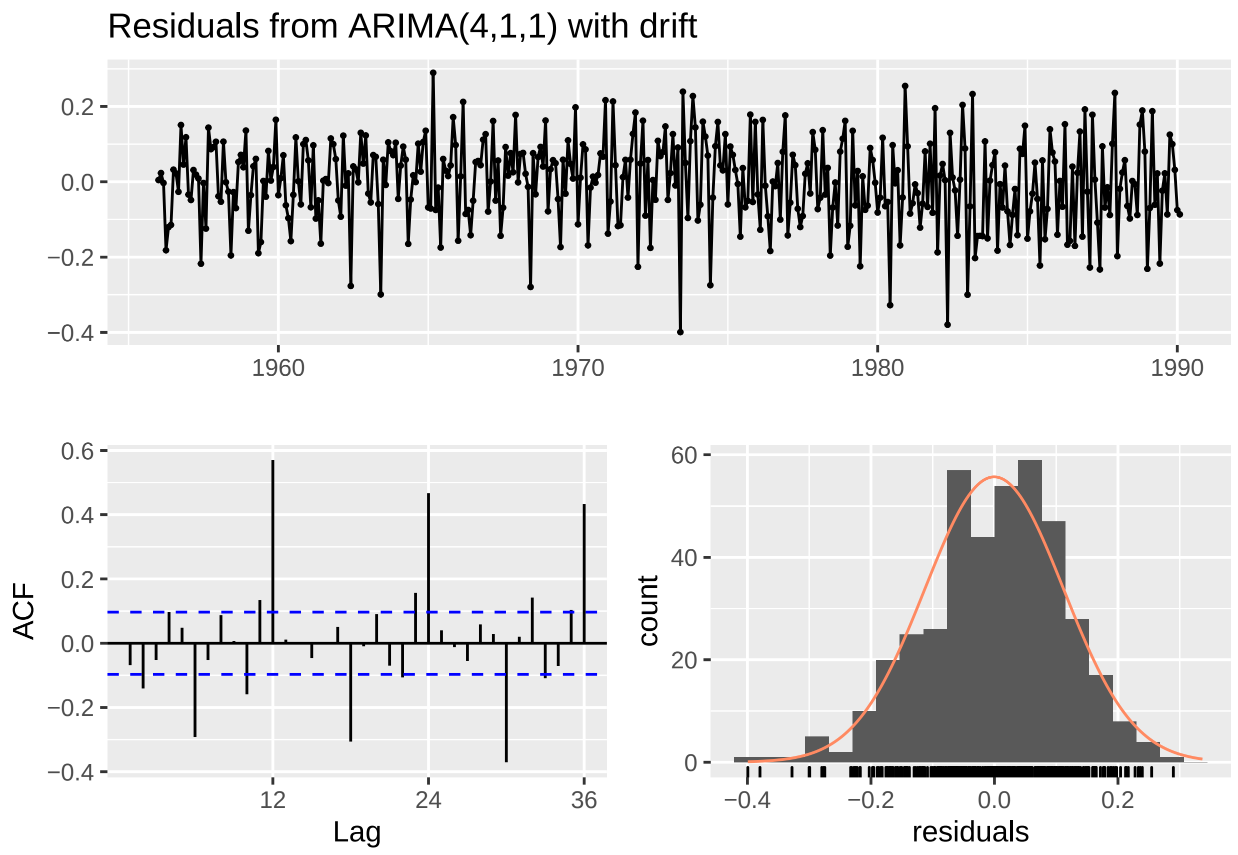
```
## Calculate the waltd confidence interval
params_arima <- beerdat_arima_411$coef # extract parameters
se_arima <- sqrt(diag(beerdat_arima_411$var.coef)) # extract s.e. for each
z <- qnorm(0.975) # this is the alpha/2 quantile
c.upper <- params_arima + z*se_arima # upper bound
c.lower <- params_arima - z*se_arima # lower bound
CI_arima_411 <- cbind(params_arima, se_arima, c.lower,c.upper)
colnames(CI_arima_411)<-c("parameter","s.d.", "2.5 %","97.5 %")
CI_arima_411
```

```
         parameter         s.d.          2.5 %       97.5 %
ar1    0.465458908 0.0470214978  0.3732984661  0.557619350
ar2    0.053715988 0.0517065296 -0.0476269473  0.155058924
ar3    0.051233576 0.0516342440 -0.0499676828  0.152434834
ar4   -0.348570052 0.0468793510 -0.4404518912 -0.256688212
ma1   -0.935950001 0.0126007088 -0.9606469365 -0.911253066
drift  0.001658102 0.0004784955  0.0007202684  0.002595936
```

Residuals inspection and diagnostics

Let's observe the residuals from the resulting model:

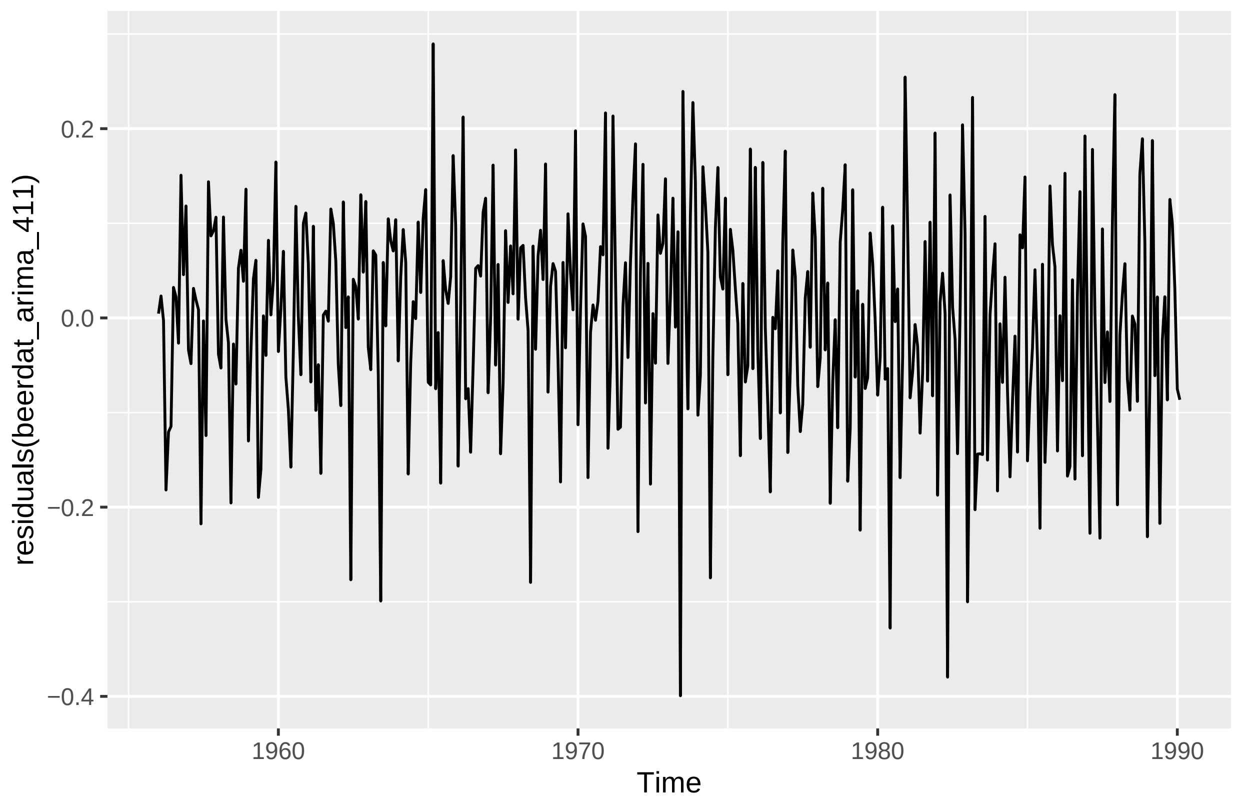
```
checkresiduals(beerdat_arima_411)
```



```
    Ljung-Box test

data:  Residuals from ARIMA(4,1,1) with drift
Q* = 371.97, df = 18, p-value < 2.2e-16

Model df: 6.   Total lags used: 24
```

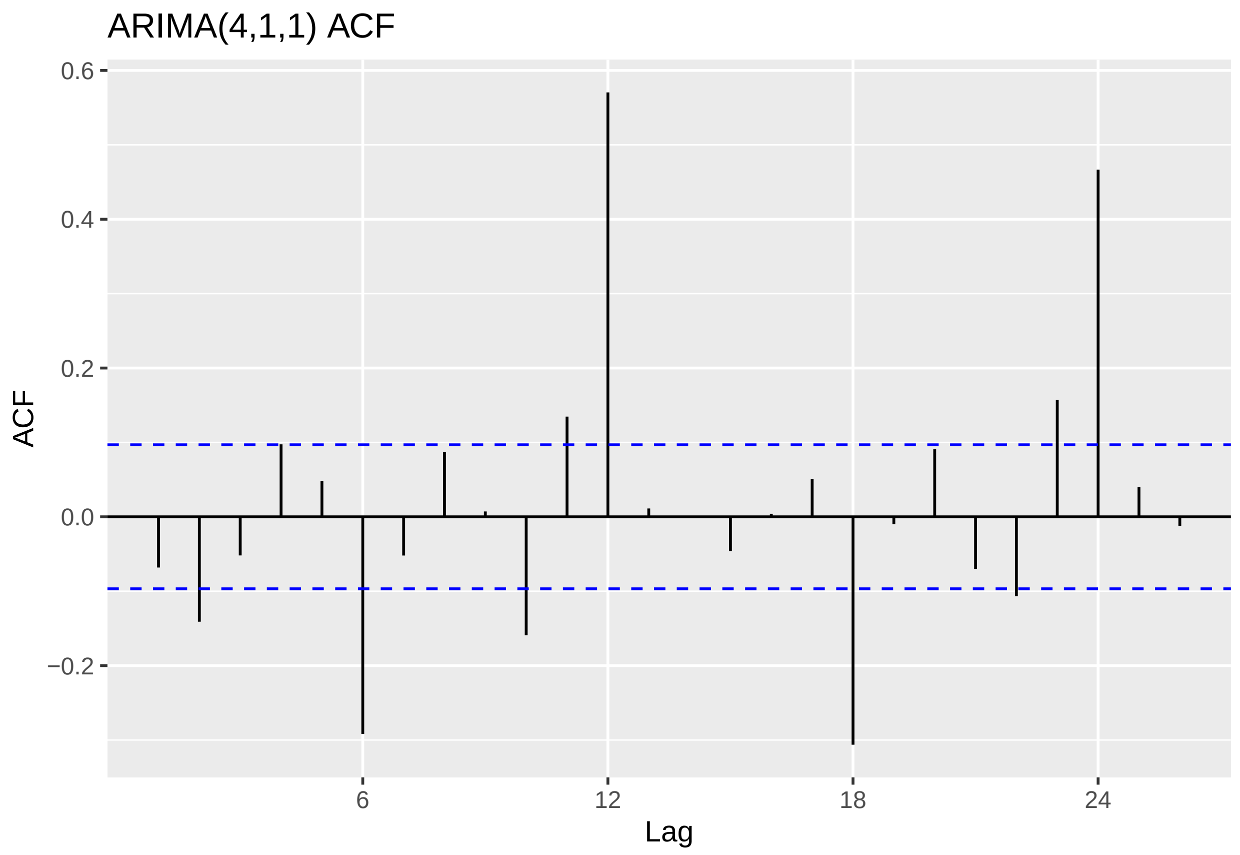
At first sight, it would seem that the residuals look somehow stationary and zero centered, as well as approximatedly normal. However, it would seem that there is still seasonality floating around! Indeed, from the Ljung-Box test, we see that we strongly reject the hypothesis of stationarity. HOWEVER! This is the wrong approach for testing stationarity for an ARIMA(p,d,q) model, so we wil come back to this.

```
# Plot the resulting model
autoplot(residuals(beerdat_arima_411))
```

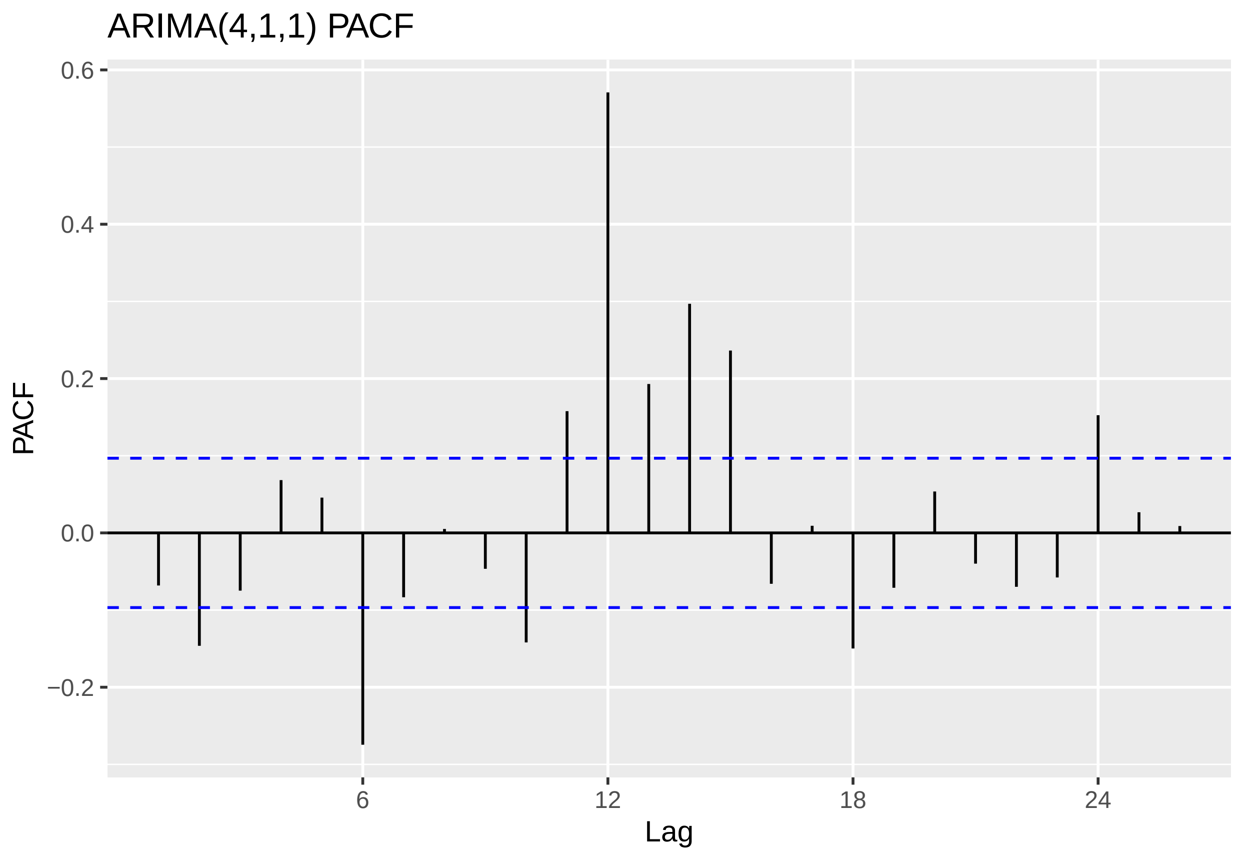


We observe that indeed the model seems to be a lot more centered towards 0, and seems a little bit more like white noise. Let us check this with the ACF and PACF plots:

```r
# Plot ACF
ggAcf(residuals(beerdat_arima_411)) + ggtitle("ARIMA(4,1,1) ACF")
```



```
# Plot PACF
ggPacf(residuals(beerdat_arima_411)) + ggtitle("ARIMA(4,1,1) PACF")
```



Inspecting the roots

Indeed, there is too much of a seasonal component flaoting around.

All of this seems to indicate the model is not appropriate, however, recall the definitions of stationarity, invertibility and causality for an ARMA(p,q) process:

- **Staionarity:** A solution to an ARMA(p,q) exists and is **stationary** iff

$$\boxed{\Phi(z) = 1 - \sum_{j=1}^{p} \phi_j z^j \neq 0}, \quad \forall |z| = 1 \text{ (for z on the unit circle)}$$

  i.e. we want **no roots** of $\Phi(z)$ on the unit circle.

- **Causality:** An ARMA(p,q) is **causal** if $\exists \ \{\psi_j\}$ such that

$$\boxed{\sum_{j=1}^{\infty} |\psi_j| < \infty} \quad \text{and} \quad \boxed{X_t = \sum_{j=1}^{\infty} \psi_j Z_{t-j}} \quad \text{(i.e. } X_t \text{ only depends on the past)}$$

  equivalently , iff

$$\boxed{\Phi(z) = 1 - \sum_{j=1}^{p} \phi_j z^j \neq 0}, \quad \forall |z| \leq 1 \text{ (for z on and inside the unit circle)}$$

- **Invertibility**: An ARMA(p,q) process $\{X_t\}$ is <u>invertible</u> if $\exists$ coefficients $\{\pi_j\}$ such that

$$\boxed{\sum_{j=1}^{\infty} |\pi_j| < \infty} \quad \text{and} \quad \boxed{Z_t = \sum_{j=1}^{\infty} \pi_j X_{t-j}}$$

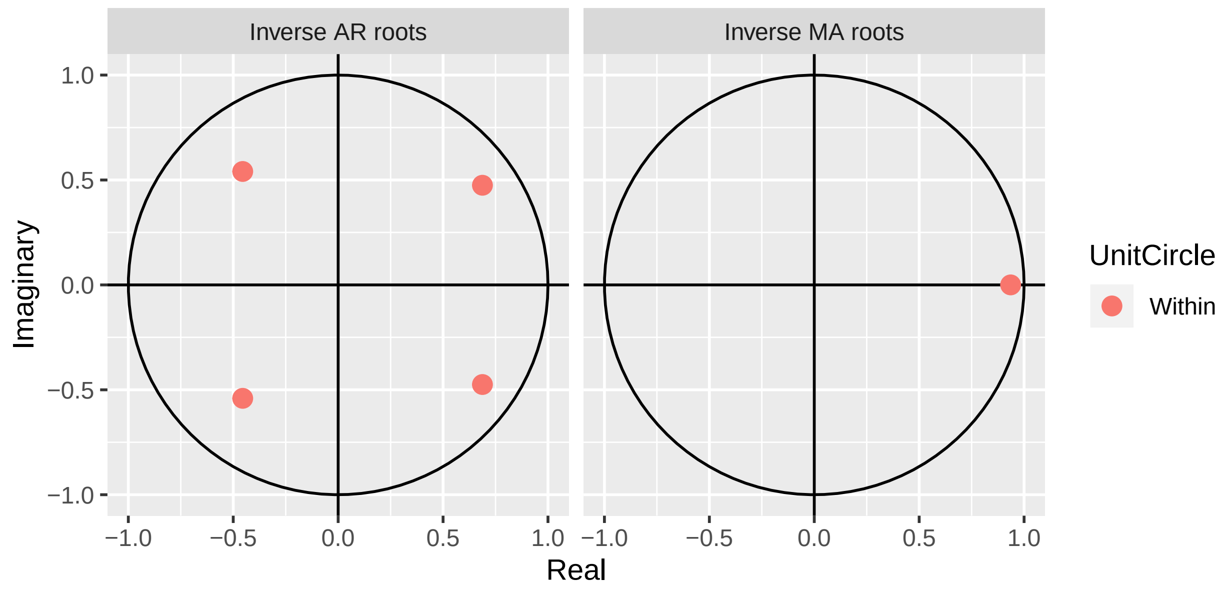
  equivalently, we want

$$\boxed{\Theta(z) = 1 + \sum_{j=1}^{q} \theta_k z^j \neq 0}, \quad \forall |z| \leq 1$$

$\equiv$ no roots of $\Theta(z)$ <u>on</u> or <u>inside</u> the unit circle.

So let's inspect the roots of what would be the AR(4) AND MA(1) polynomials:

```r
# Plot the toods
autoplot(beerdat_arima_411)
```



We observe that all the inverse AR and MA roots lie inside the unit circle respectively, which indicates that our ARMA(4,1) process we would obtain by differencing once the ARMA(4,1,1) is a stationary, causal and invertible processs.

Testing the ARIMA(4,1,1) stationarity

Since we **cannot** use likelihood-based approaches for the ARIMA, instead, we will perform ADP and KPSS tests before, and after taking one difference of the ARMA(4,1,1) process. Recall the respective hypotheses for each of the tests:

- **ADP**

$$\begin{cases} H_0: & \text{The process is non-stationary (i.e. at least one unit root)} \\ H_a: & \text{The process is stationary} \end{cases}$$

- **KSPP**

$$\begin{cases} H_0: & \text{The process is stationary (i.e. no unit roots)} \\ H_a: & \text{The process is non-stationary} \end{cases}$$

```
# Test with a bunch of different k's ? (bigger augmented versions)
adf.test(residuals(beerdat_arima_411))
```

```
Warning in adf.test(residuals(beerdat_arima_411)): p-value smaller than printed
p-value


    Augmented Dickey-Fuller Test

data:  residuals(beerdat_arima_411)
Dickey-Fuller = -10.12, Lag order = 7, p-value = 0.01
alternative hypothesis: stationary
```

```
kpss.test(residuals(beerdat_arima_411))
```

```
Warning in kpss.test(residuals(beerdat_arima_411)): p-value smaller than printed
p-value


    KPSS Test for Level Stationarity

data:  residuals(beerdat_arima_411)
KPSS Level = 0.74841, Truncation lag parameter = 5, p-value = 0.01
```

So in this case the ADF test tells us that the process is stationary, while the second one tells us it is not. Therefore we should try differencing to see whether this solves the issue.

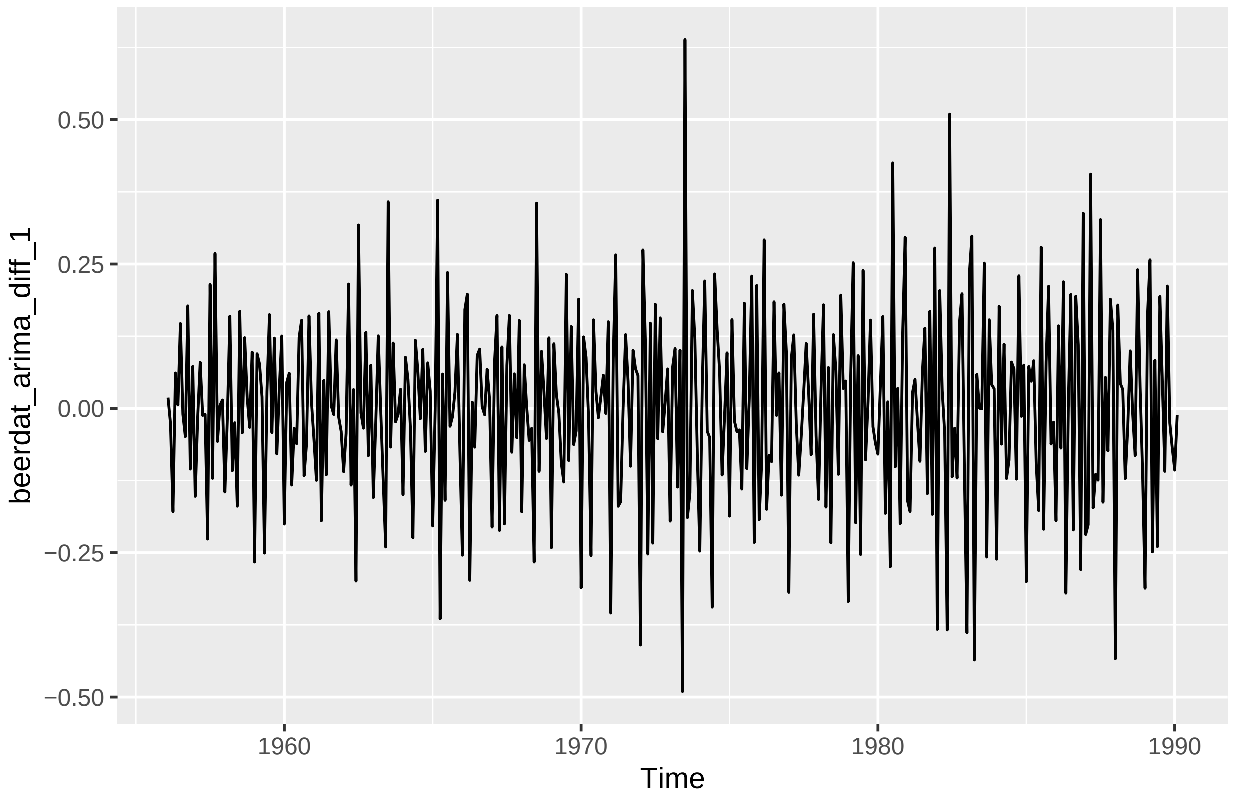
Differencing the ARIMA(4,1,1)

The new process is

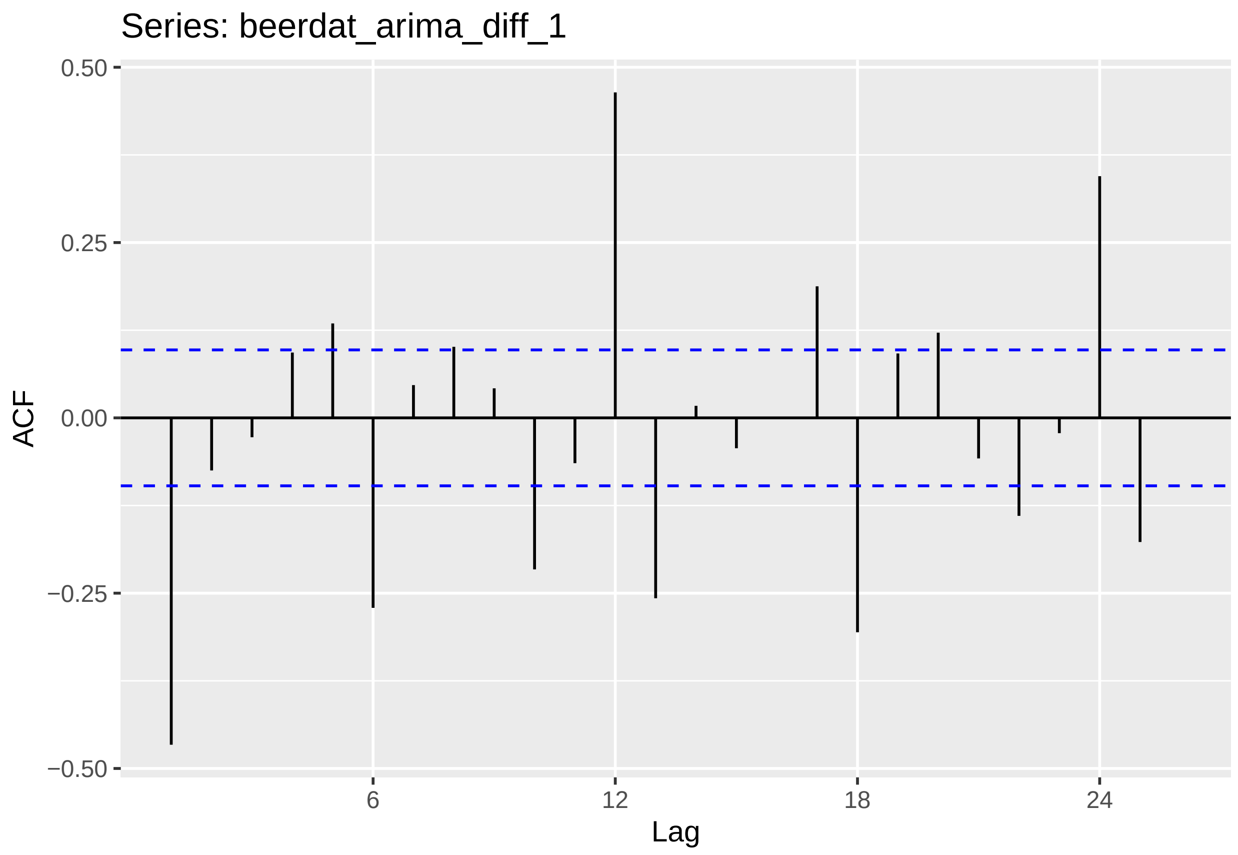
```
# differentiate the residuals form before
beerdat_arima_diff_1 <- diff(residuals(beerdat_arima_411), 1)
```

Let's inspect the plot, ACF and PACF

```r
# Plot the seris
autoplot(beerdat_arima_diff_1)
```

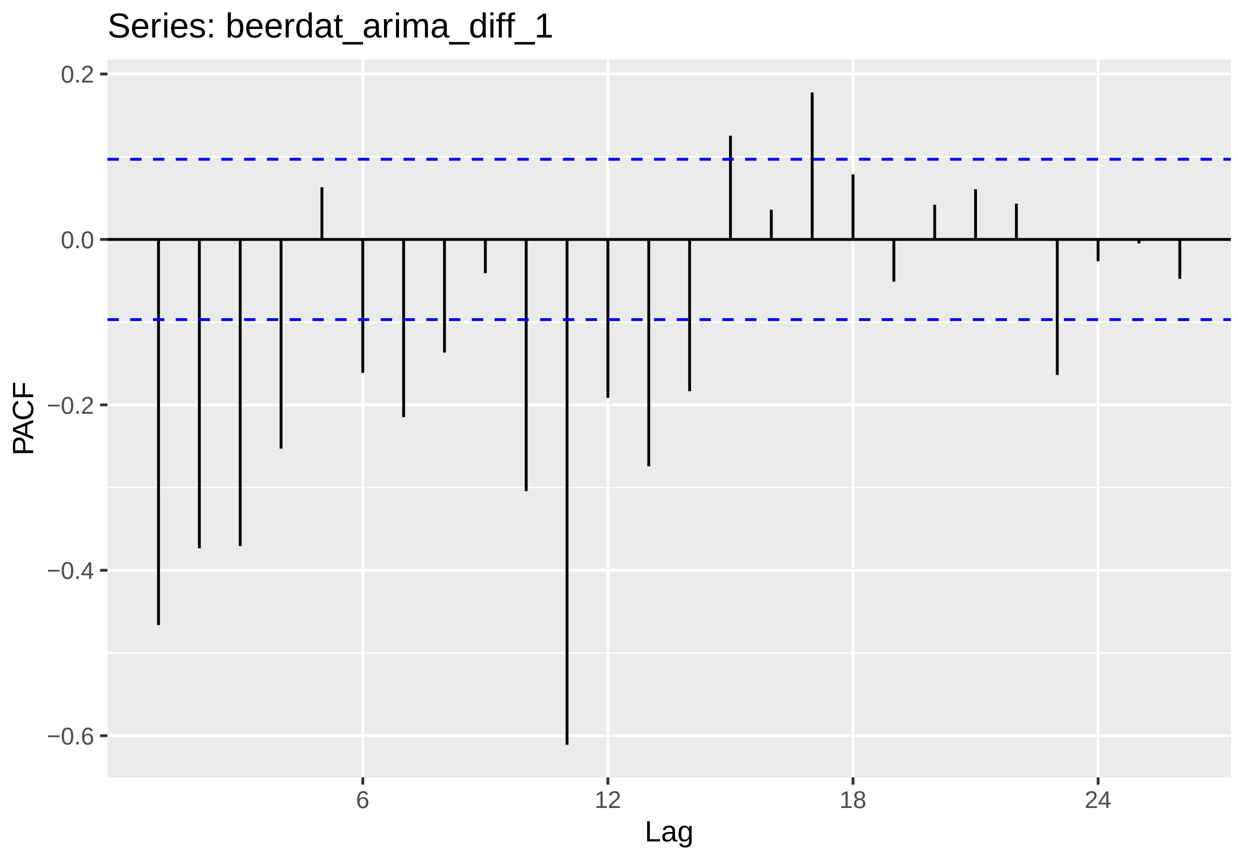


```r
# Plot ACF
ggAcf(beerdat_arima_diff_1)
```



This is still not good! :(

```
# Plot ACF
ggPacf(beerdat_arima_diff_1)
```



In fact it looks like we made things worse. That PACF doesn't look healthy at all. This model is not good.

Attempting to remove seasonality from the ARIMA(1,1,1)

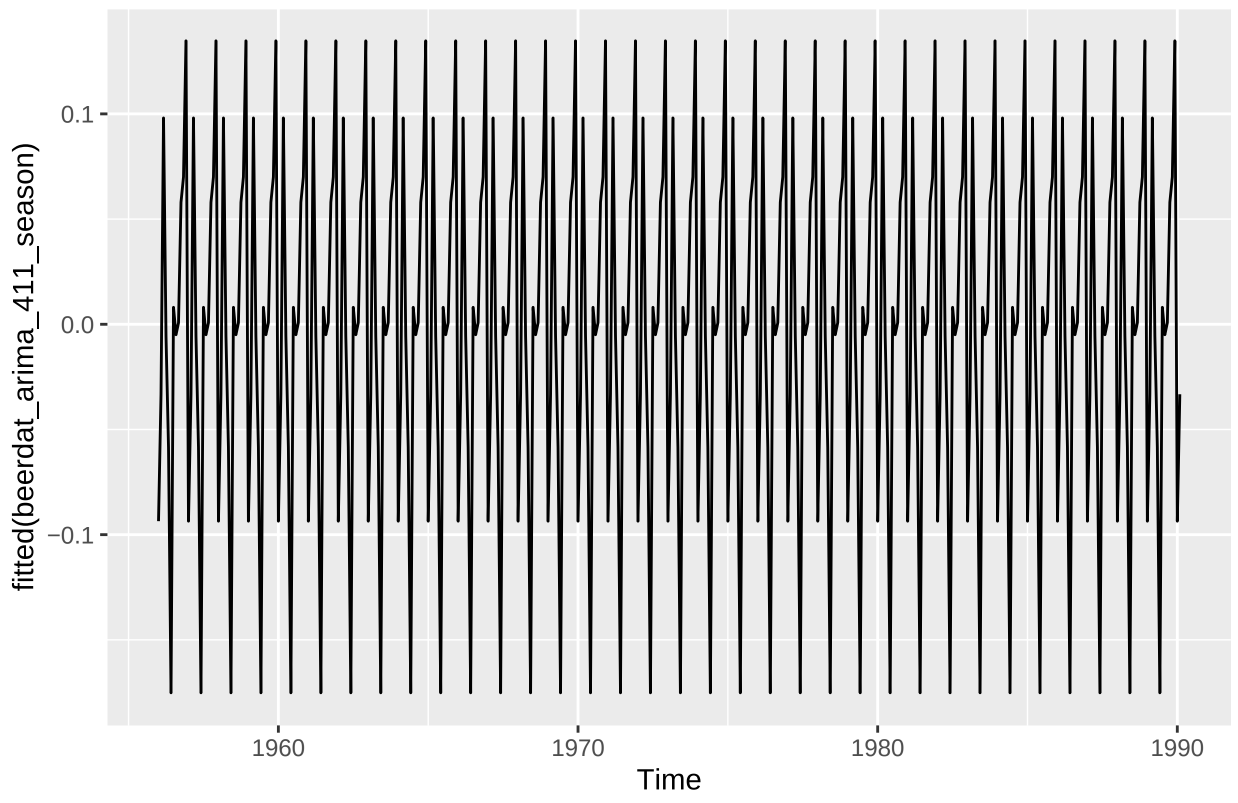
Let's try removing seasonality from the ARIMA(4,1,1) **before** differencing to see if this helps.

```r
# frequency obtains the lag?
frequency(residuals(beerdat_arima_411))
```

```
[1] 12
```

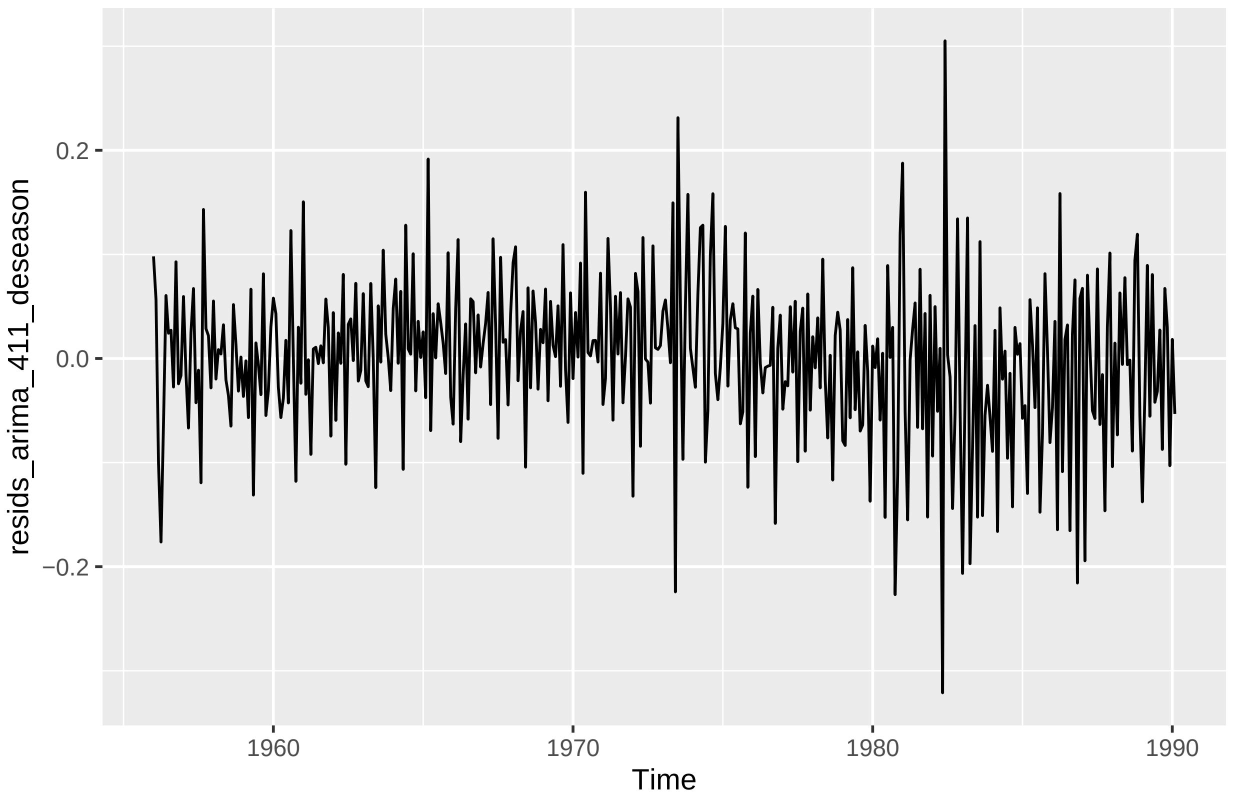
This is an indication that we have a lag every 12 months (indeed, every year!)

```r
# attempt to calculate the de-seasoned data
beerdat_arima_411_season <- tslm(residuals(beerdat_arima_411) ~ season)
autoplot(fitted(beerdat_arima_411_season))
```



Now le't s remove this from the original series

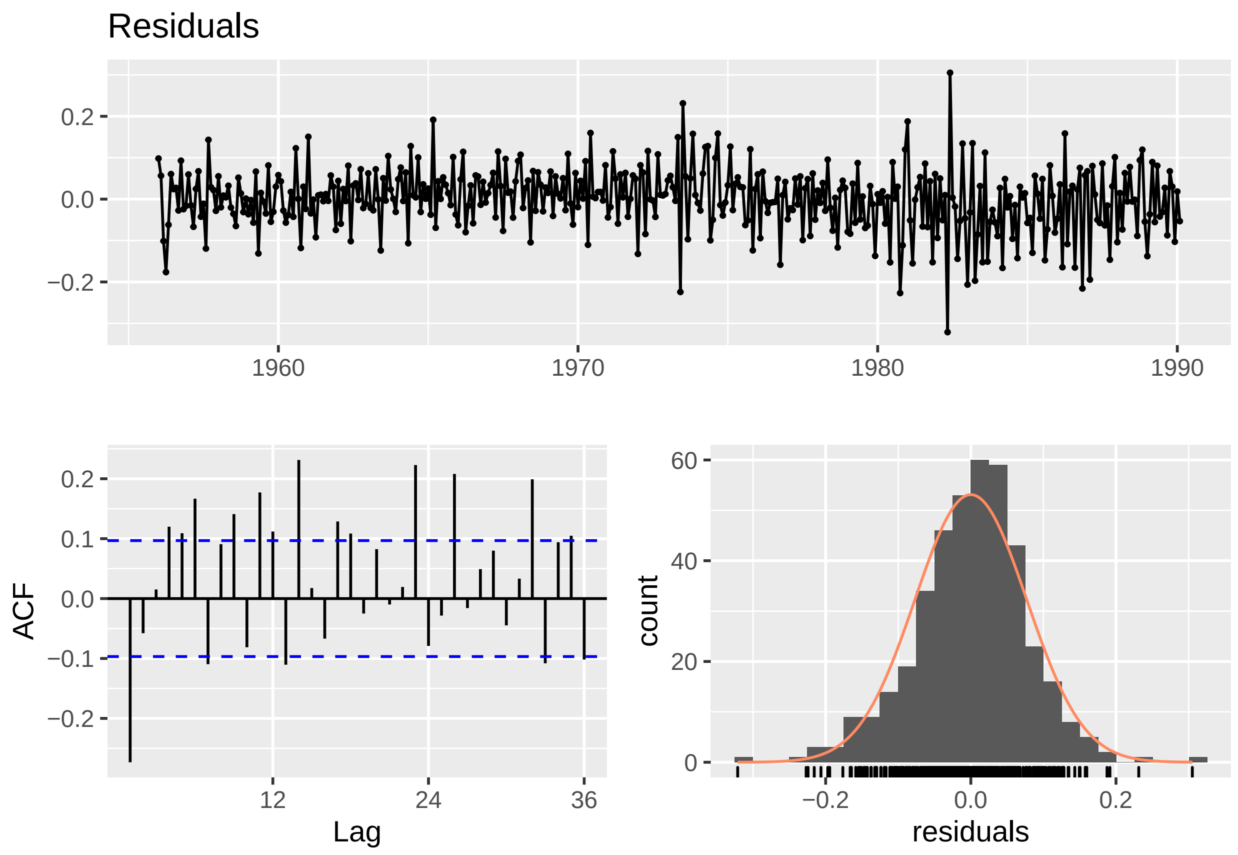
```
# Remove seasonal component
resids_arima_411_deseason <- ( residuals(beerdat_arima_411) - fitted(beerdat_arima_411_season) )
autoplot(resids_arima_411_deseason)
```



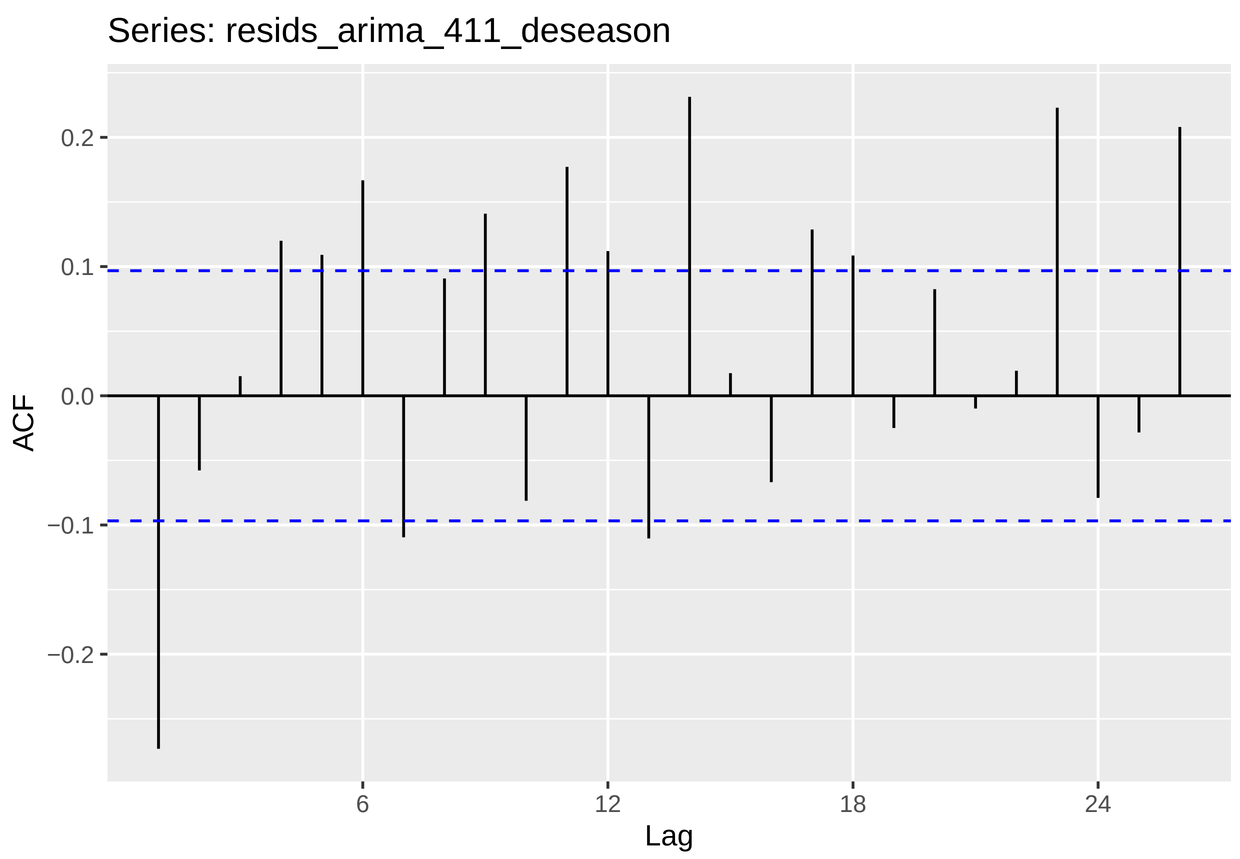
Now let's inspect the residuals

```
checkresiduals(resids_arima_411_deseason)
```

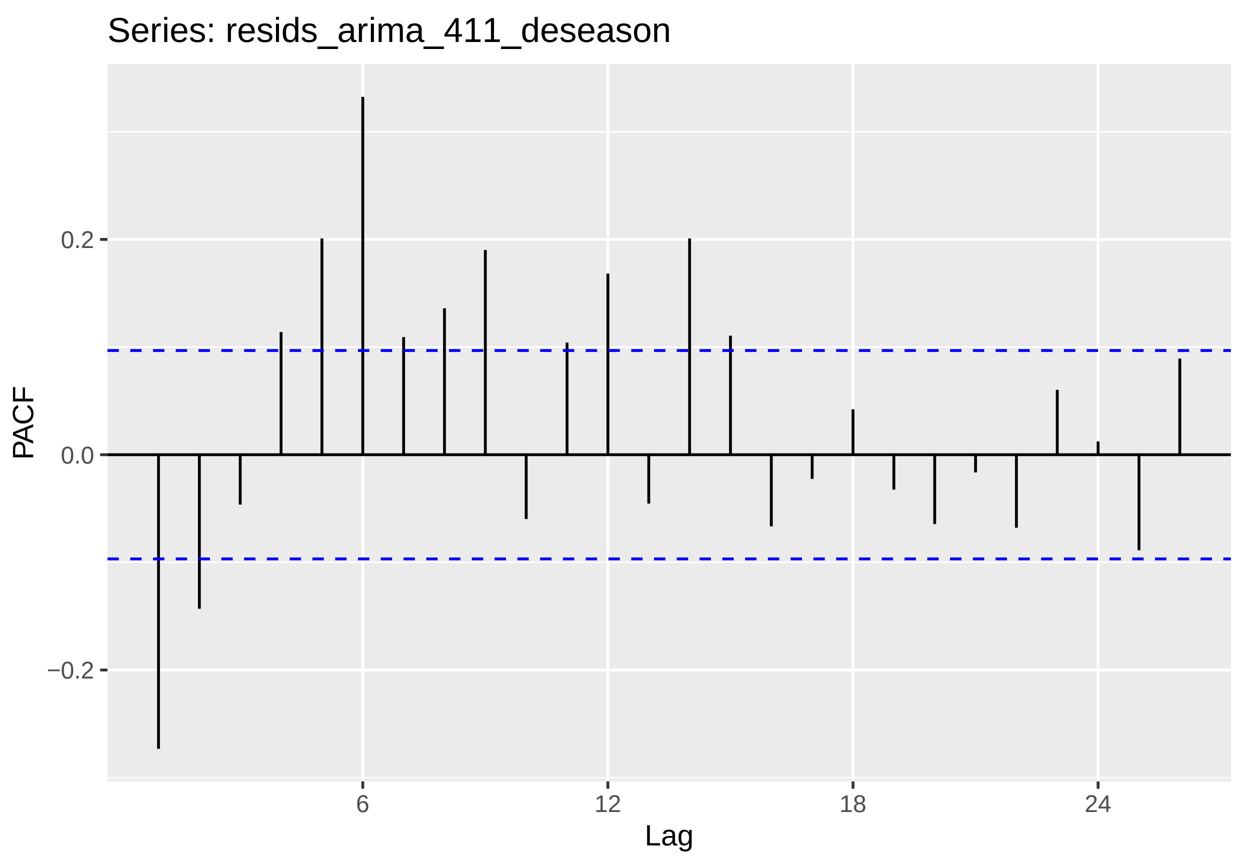
```
Warning in modeldf.default(object): Could not find appropriate degrees of
freedom for this model.
```



```
# Plot ACF
ggAcf(resids_arima_411_deseason)
```



```r
# Plot PACF
ggPacf(resids_arima_411_deseason)
```



Although we now have a series that looks sltightly a bit more like white noise, the ACF plot still doens't look that healthy. It appears that now a new trend has surged, which indicates that perhaps we should have remmoved it from the beginning instead. For now, we will simply see the forecasted values of the ARIMA(4,1,1) model.

Forecasting

Just for clarity, let's re-fit the ARIMA(4,1,1) model with the trianing data and predict on the test data:

```r
# Fit the ARIMA model on trian data
beerdat_arima_411 = auto.arima(beer_dat_train,
                        seasonal=FALSE,
                        stepwise=FALSE,
                        ic = c("aicc", "aic", "bic") ,
                        approximation=FALSE,
                        trace=FALSE)

# Forecast 12 observations
beerdat_arima_411_forecasts = forecast::forecast(beerdat_arima_411,h=12)
```

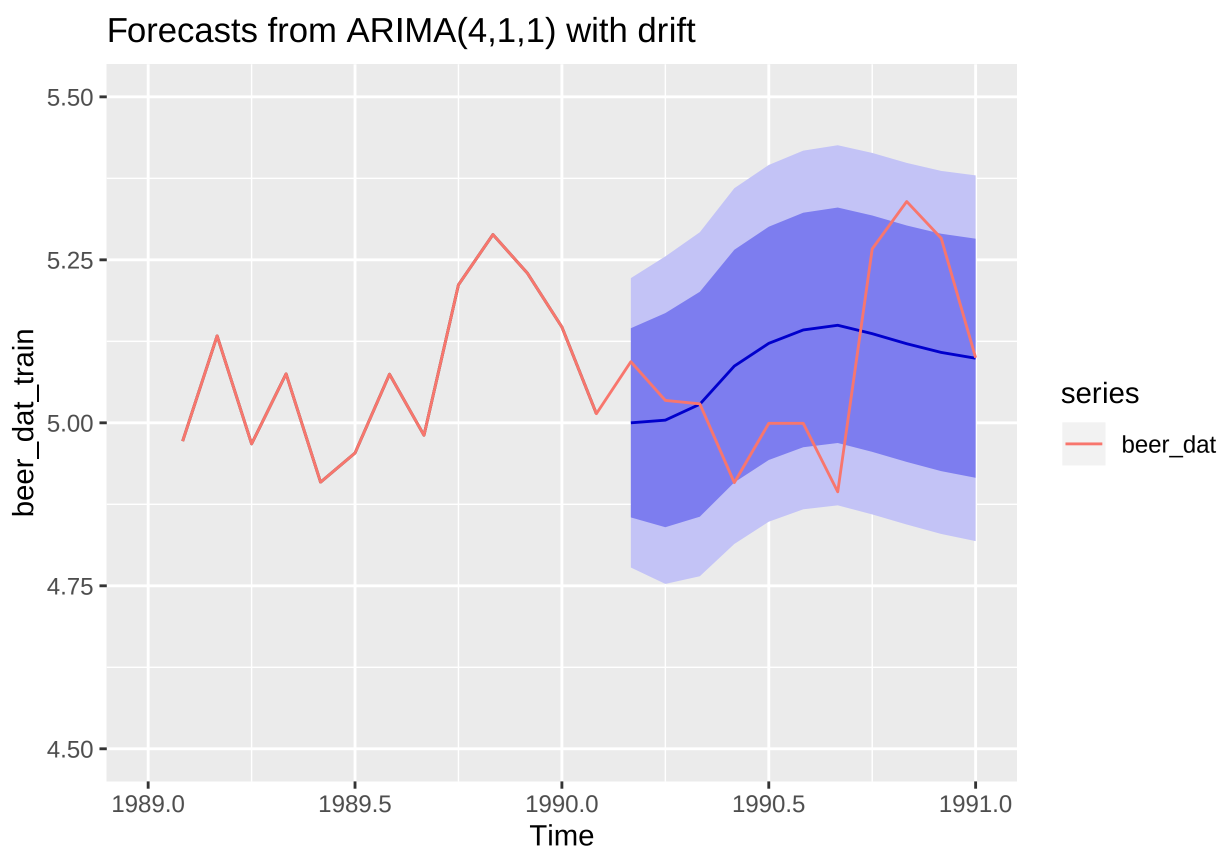
We can also produce the values into a nice table containing the forecasting points with confidence intervals

```r
# Print the forecasts along with the boundaries
forecast_table<-print(beerdat_arima_411_forecasts) %>%
  mutate(observed=beer_dat_test,
         errors=`Point Forecast`-observed)
         Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
Mar 1990       5.000004 4.854874 5.145133 4.778047 5.221960
Apr 1990       5.004083 4.839863 5.168302 4.752931 5.255235
May 1990       5.028499 4.855981 5.201016 4.764656 5.292342
Jun 1990       5.087000 4.908593 5.265408 4.814150 5.359851
Jul 1990       5.121907 4.943000 5.300813 4.848293 5.395521
Aug 1990       5.142416 4.962550 5.322282 4.867334 5.417498
Sep 1990       5.149614 4.969024 5.330204 4.873425 5.425803
Oct 1990       5.136753 4.955477 5.318029 4.859515 5.413991
Nov 1990       5.121327 4.939987 5.302667 4.843992 5.398662
Dec 1990       5.107966 4.925893 5.290039 4.829510 5.386423
Jan 1991       5.099041 4.915647 5.282435 4.818564 5.379518
Feb 1991       5.099152 4.913904 5.284399 4.815840 5.382464
```

We see that although the model diagnostics were rather bad, all of the forecasted points still fall inside their respective confidence intervals

```r
# Plot the predictions
autoplot(beerdat_arima_411_forecasts) +
  autolayer(beer_dat) + xlim(c(1989,1991)) + ylim(c(4.5,5.5))
```

```
Scale for 'x' is already present. Adding another scale for 'x', which will
replace the existing scale.

Warning: Removed 397 rows containing missing values (geom_path).

Warning: Removed 398 rows containing missing values (geom_path).
```



Above: - The red is the original simulated data - Then the other is the best linear predictor for the data (blue line), along with condifence bounds (80% and 95%) - Most of the points are within the 80$ confidence intervals

Evaluating predictions

We will now compare how many of these were "right", by comparing them to the test data:

```
# Check how many points are inside the 95% confidence intervals
forecast_table %>%
  mutate(outside95=I(observed<`Lo 95` | observed>`Hi 95`)) %>%
  count(outside95)
```

```
# A tibble: 1 x 2
  outside95      n
  <I<lgl>>   <int>
1 FALSE         12
```

```
forecast_table %>%
  mutate(outside80=I(observed<`Lo 80` | observed>`Hi 80`)) %>%
  count(outside80)
```

```
# A tibble: 2 x 2
  outside80      n
  <I<lgl>>  <int>
1 FALSE         9
2 TRUE          3
```

We observe that there are 3 points that fall outside the 80% confidence bounds, , whereas all of them fall within the 95% ones. This implies arounf 75% prediction accuracy.

Beer Data: Classical Decomposition approach

Background: Classical Decomposition

We saw that we had some problems with correclty estimating trend and seasonality for the `beer` data. We will now try the **classical decomposition** approach before fitting the series, i.e. we assume that for a time series $\{X_t\}$

$$X_t = \underbrace{m_t}_{trend} + \underbrace{s_t}_{season} + \underbrace{Y_t}_{noise}$$

where

$$\mathbb{E}[Y_t] = 0 \qquad s_{t+d} = s_t,\ \forall t \qquad \sum_{j=1}^{d} s_j = 0$$

So we will follow the following approach:

1) Estimate the trend $\hat{m}_t$, and remove it, i.e $\tilde{X}_t = X_t - \hat{m}_t \approx s_t + Y_t$ (by linear regression, moving avg, etc.)
2) Estimate the seasonal component, say $\hat{s}_t$, and obtain deseasonalized data $d_t = X_t - \hat{s}_t$, where

$$\hat{s}_k = w_k - \frac{1}{d}\sum_{i=1}^{d} w_i \qquad w_k = \frac{1}{(t/d)}\sum_{j=0}^{t/d-1} (X_{k+jd} - \hat{m}_{k+jd})$$

3) Re-esimate the trend from d_t , say $\tilde{m}_t$, and calculate

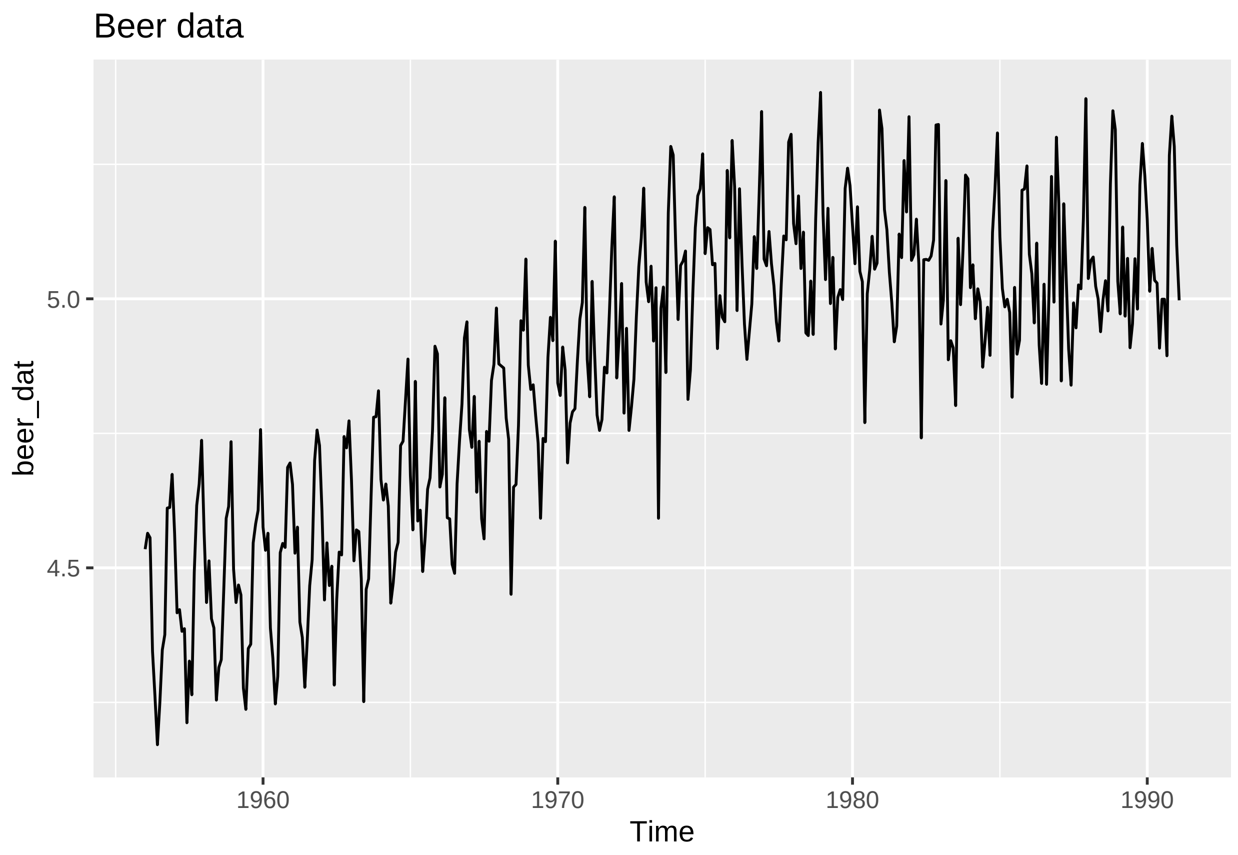
$$\tilde{Y}_t = X_t - \hat{s}_t - \tilde{m}_t$$

4) Finally, we will fit an ARMA model and make predictions on that! (then re-add the season and trend, and voilà)

Re-inspecting the data

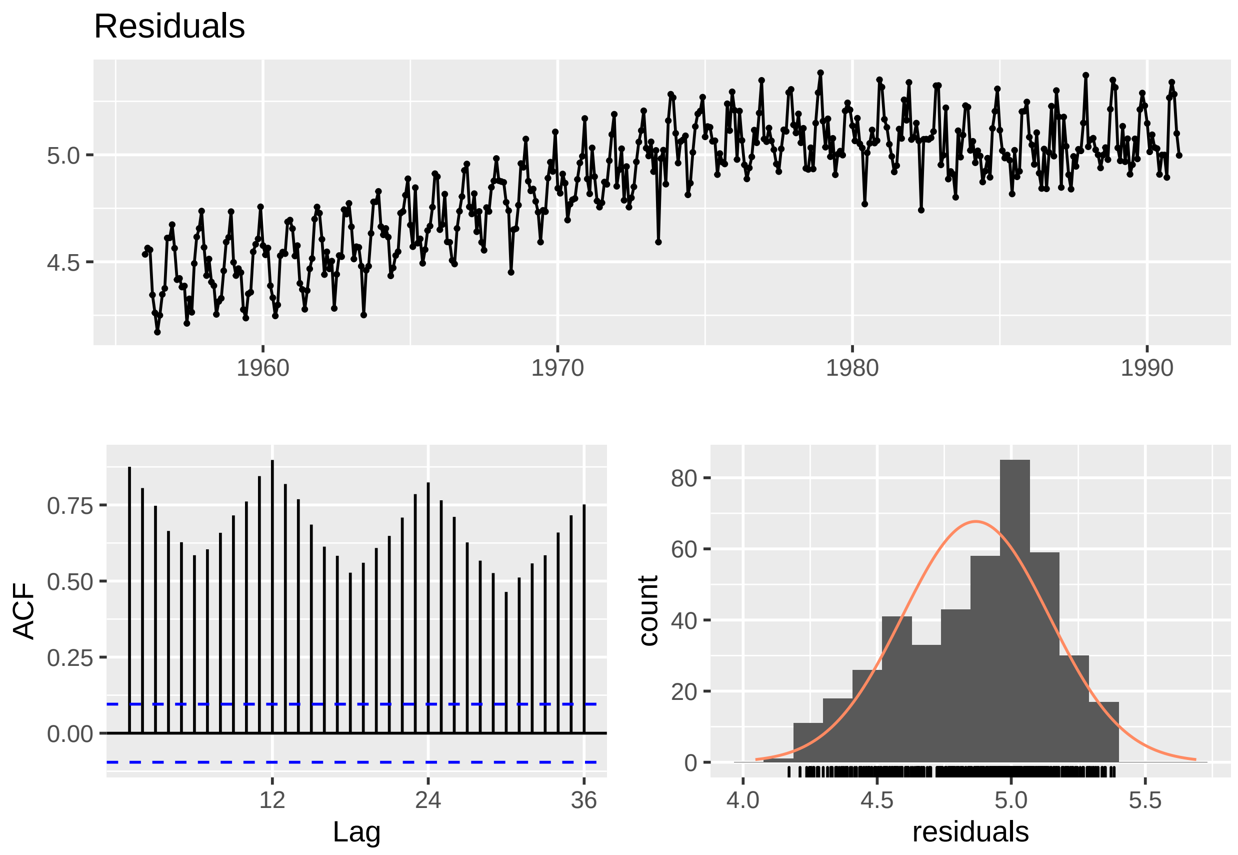
Let's take a look at our data once more:

```r
# Plot series
autoplot(beer_dat) + ggtitle("Beer data")
```



```
checkresiduals(beer_dat)
```

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

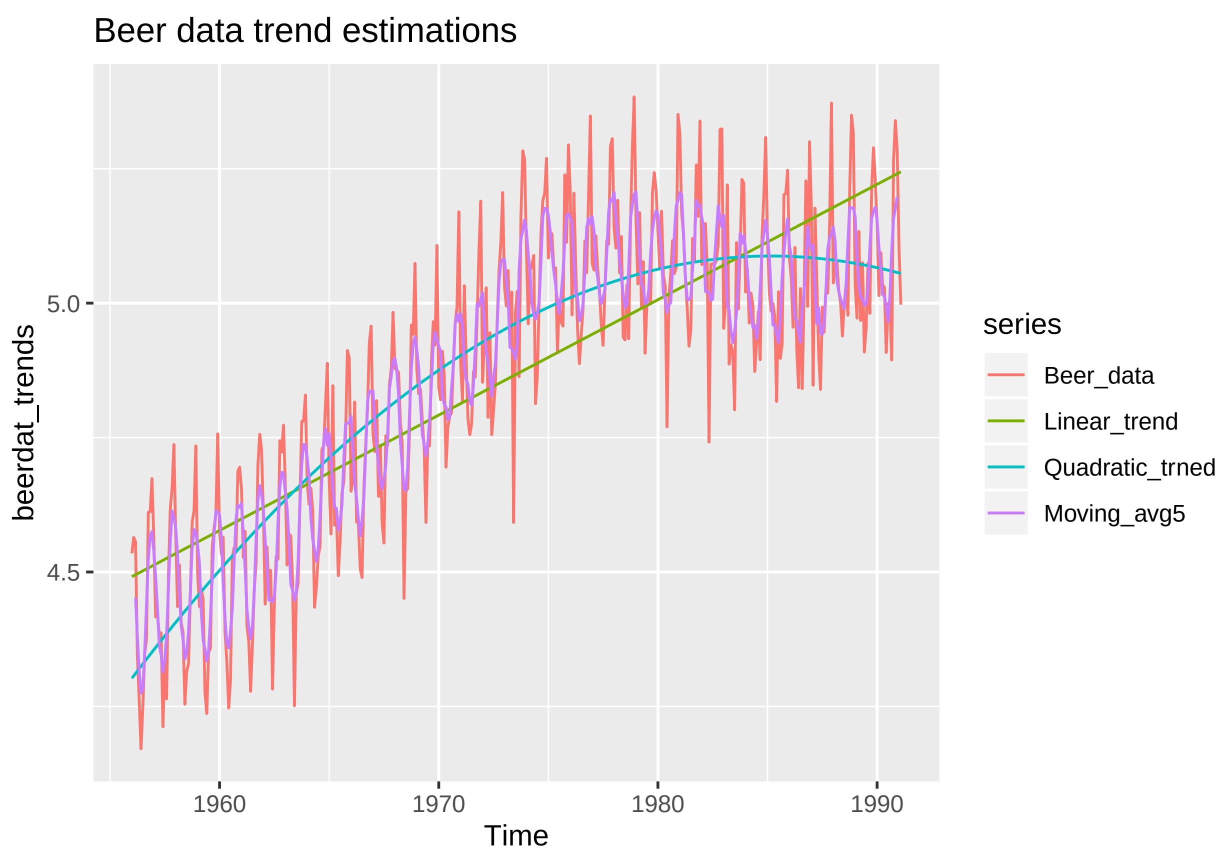


We can observe once again there is a clear trend in the data, as well as a seasonal component present. The data is definitely not white noise.

Estimating and removing trend

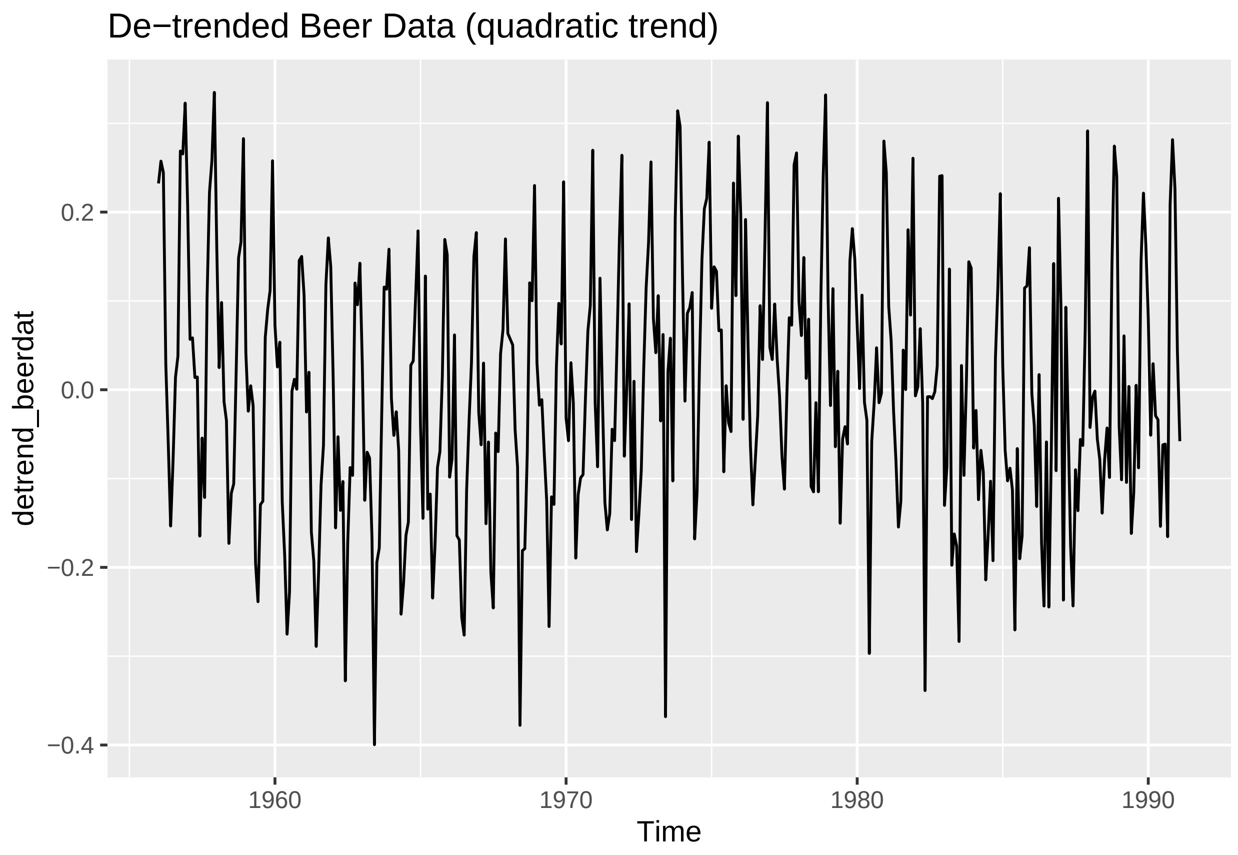
We will try estimating various trends to the data, being aware that if we over-estimate the trend we might lose predictive power later:

```r
# Estimate various trends
beerdat_trend_linear <- tslm(beer_dat~trend)
beerdat_trend_quad <- tslm(beer_dat~trend + I(trend^2))
beerdat_trend_ma5 <- ma(beer_dat, order=5)
beerdat_trends <- cbind(Beer_data=beer_dat,
                        Linear_trend=fitted(beerdat_trend_linear),
                        Quadratic_trned=fitted(beerdat_trend_quad),
                        Moving_avg5 = beerdat_trend_ma5
                        )

# Plot all the trends together
autoplot(beerdat_trends) + ggtitle("Beer data trend estimations")
```



We will opt to remove the quadratic trend as the moving average may be a bit of an overkill :)

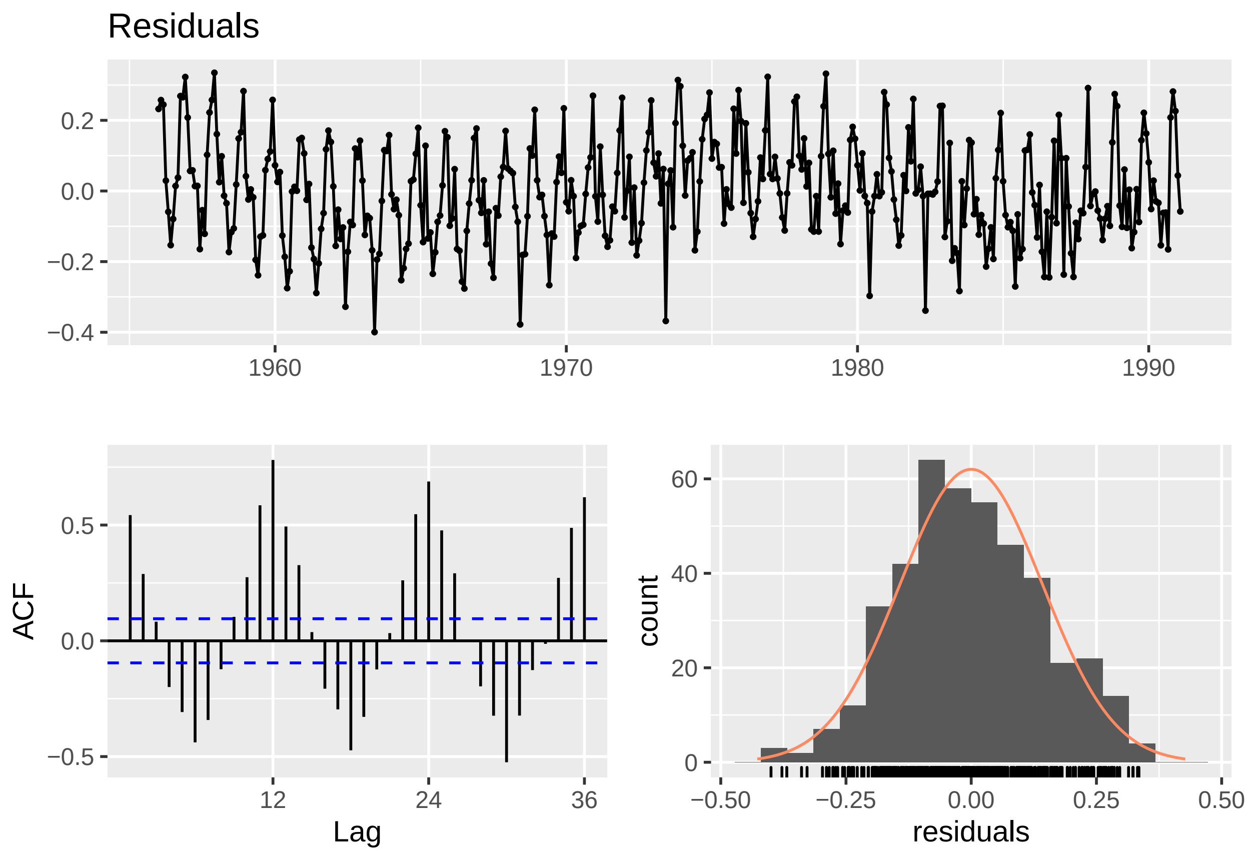
```
# Detrend and show the de-trended series
detrend_beerdat <- beer_dat - fitted(beerdat_trend_quad)
autoplot(detrend_beerdat) + ggtitle("De-trended Beer Data (quadratic trend)")
```



Let's quicly inspect the residuals

```r
checkresiduals(detrend_beerdat)
```

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

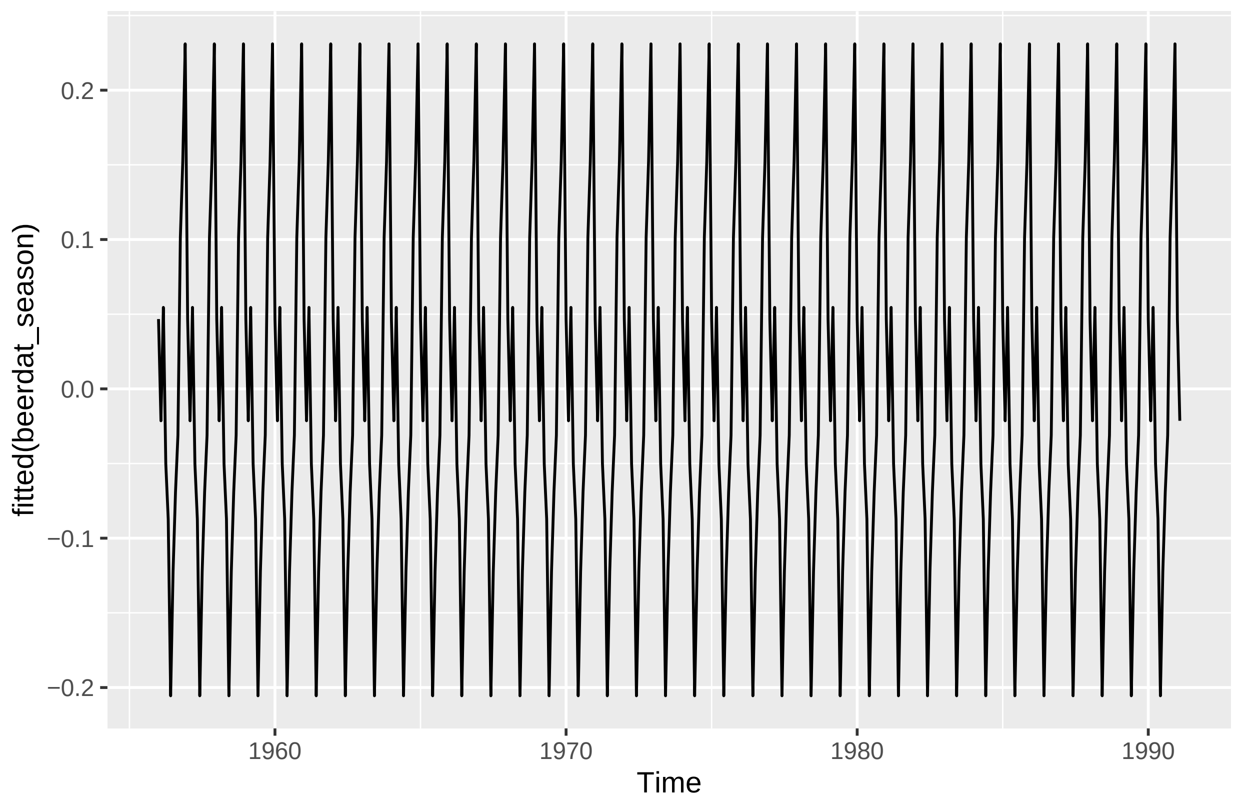


The residuals definitely seem a lot more centered towards 0, however there still seems to be a trend present (NOTE: it looks a bit like a sin/cos wave?) Let's however proceed with removing the seasonal component.

Removing seasonality

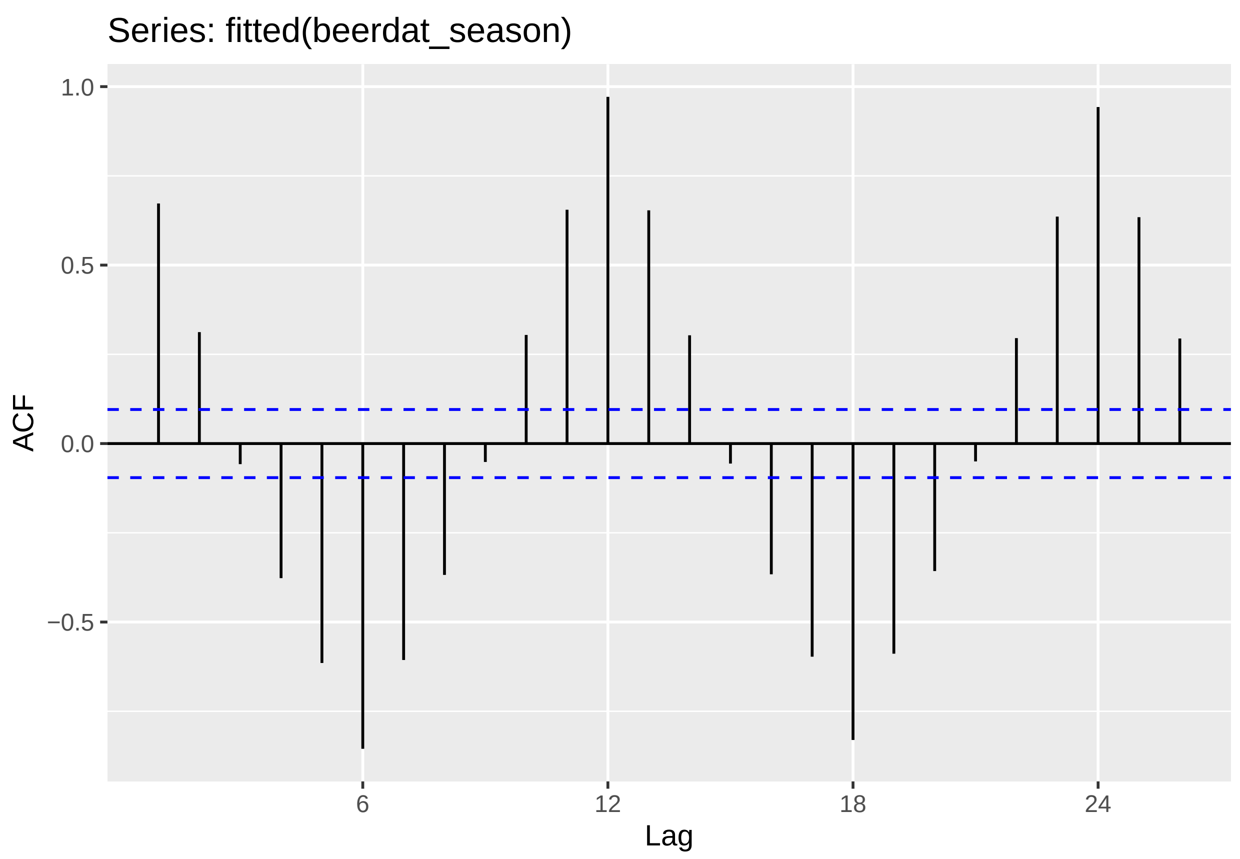
We will now estimate and remove the seasonal component.

```r
# Estimate seasonality
beerdat_season <- tslm(detrend_beerdat ~ season)
autoplot(fitted(beerdat_season))
```



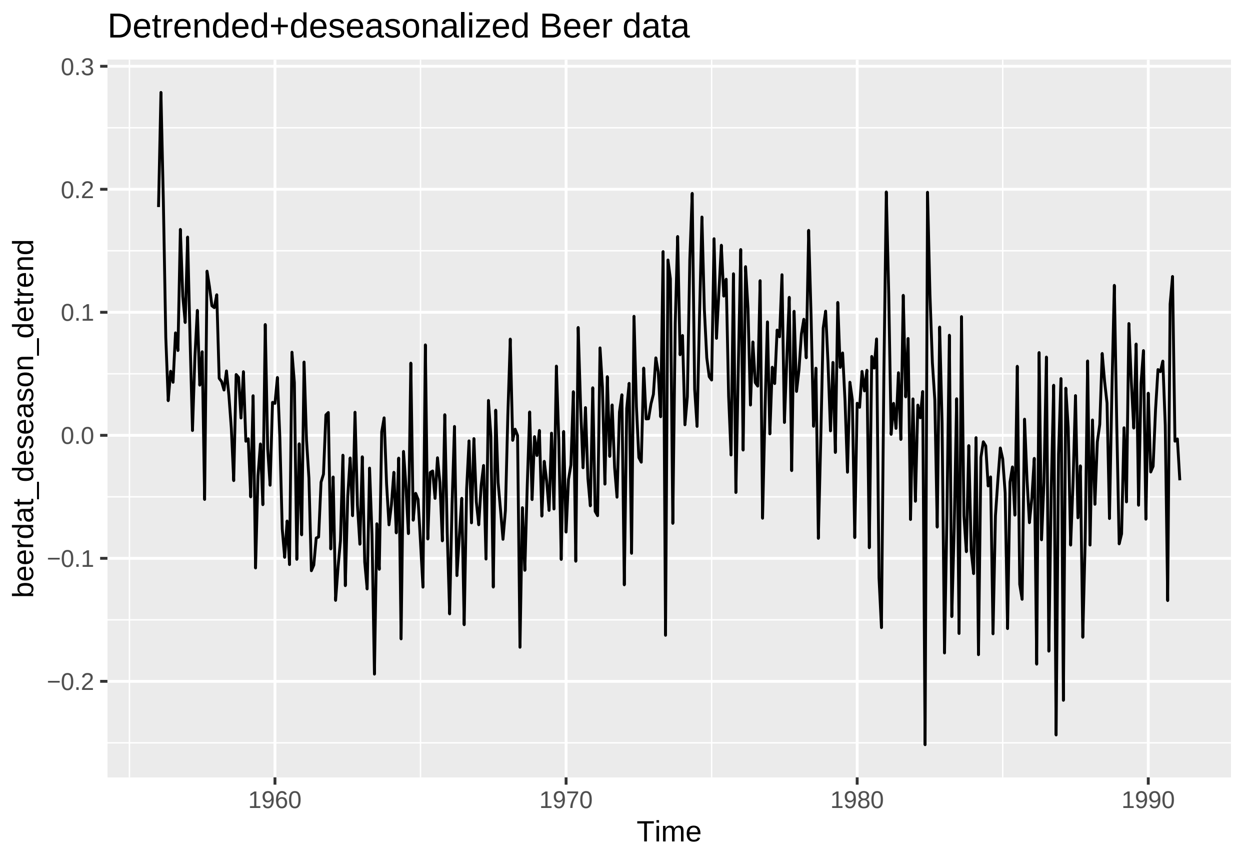
Great! Let's now take a look at the ACF

```
# ACF
ggAcf(fitted(beerdat_season))
```



It seems that we have correctly estimated the seasonality, as the ACF clearly shows the seasonal dependency each 6 lags. Now, let's remove it from our data:

```r
# Remove seasonal component and plot
beerdat_deseason_detrend <- (detrend_beerdat - fitted(beerdat_season))
autoplot(beerdat_deseason_detrend) + ggtitle("Detrended+deseasonalized Beer data")
```

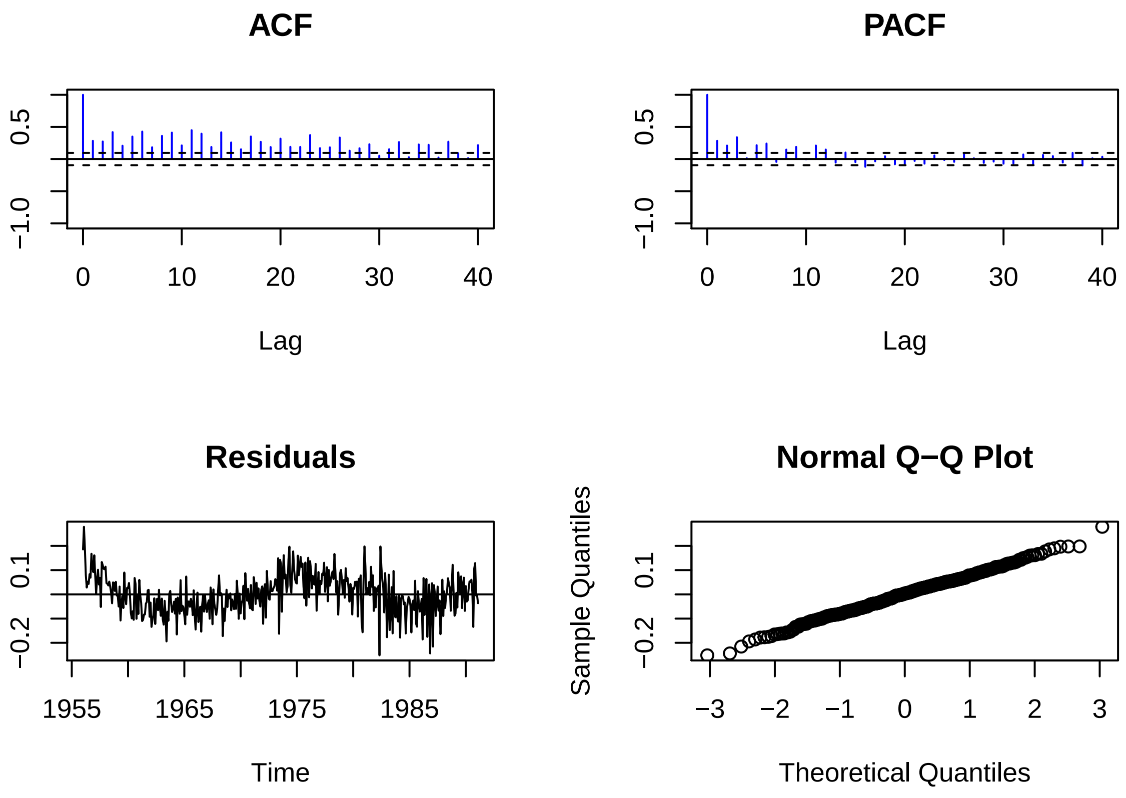


OH boy, that doesn't look like white noise…

Testing the detrended + deseasonalized data

```
test(beerdat_deseason_detrend)
```

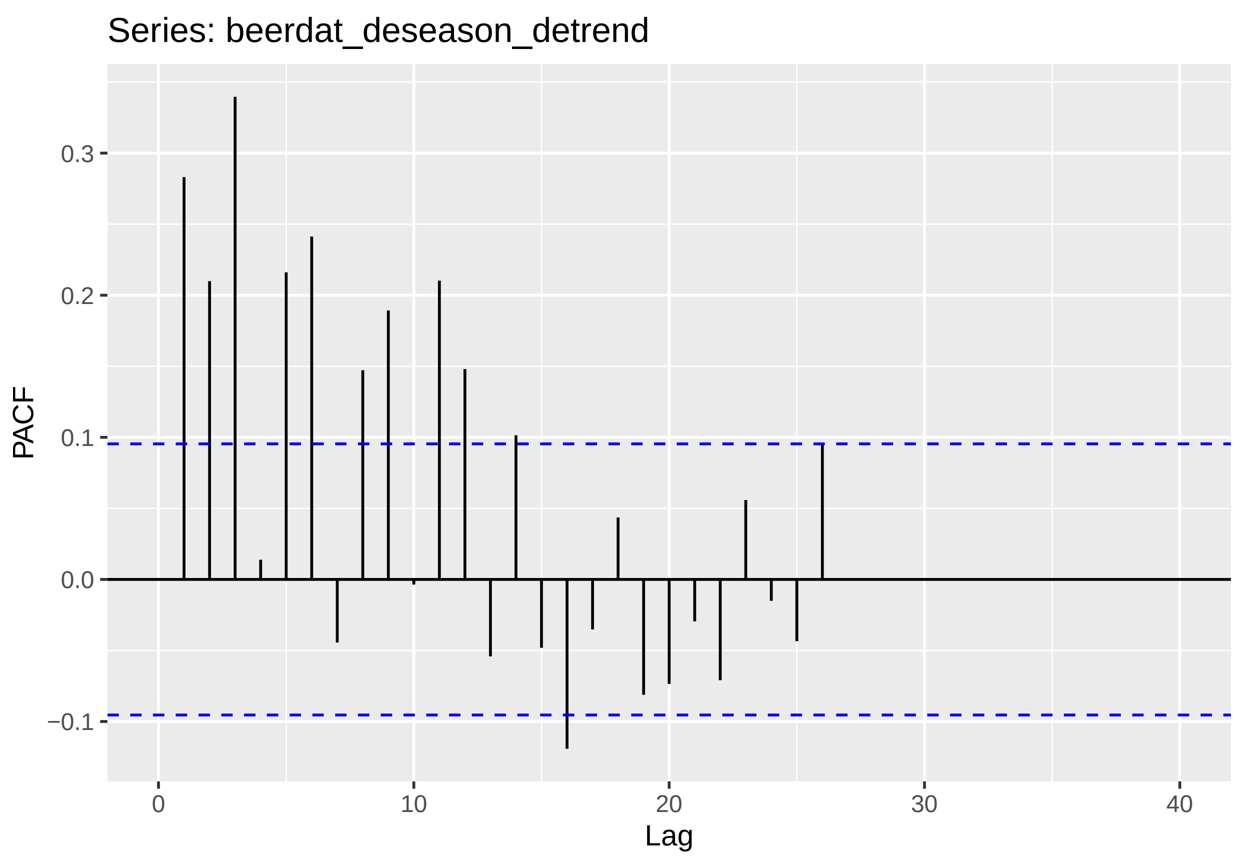
```
Null hypothesis: Residuals are iid noise.
Test                        Distribution Statistic   p-value
Ljung-Box Q                Q ~ chisq(20)     887.9         0 *
McLeod-Li Q                Q ~ chisq(20)     72.55         0 *
Turning points T    (T-280)/8.6 ~ N(0,1)       287     0.418
Diff signs S      (S-210.5)/5.9 ~ N(0,1)       205    0.3543
Rank P       (P-44415.5)/1447.4 ~ N(0,1)     45222    0.5774
```



FUnnily enough, although the residuals definitely don't look like white noise , and the series seems not to be stationary according to three out of the five tests (hoewer note that the Ljung-Box and the McLod-Li seem to indicate the opposite), wecan see from the PACF that in fact most of the correlation can be explained by the first ~12 lags, which is indicative of the seasonal component:

```r
# Plot PACF for residuals
ggPacf(beerdat_deseason_detrend) + xlim(c(0,40))
```

```
Scale for 'x' is already present. Adding another scale for 'x', which will
replace the existing scale.
```



Fitting an ARMA model

We will still try to fit an ARMA model to these series, and see if they perform better than the ARIMA estimation we performed before. However, we will have to separate the data into train and test sets since we detrended and deseasonalized the whole original data.

```
# Preprocess data
beer_dat_train_dd <- window(beerdat_deseason_detrend, end=1990.1) # train dataset
beer_dat_test_dd <- window(beerdat_deseason_detrend, start=1990.1) # test dataset
str(beer_dat_train_dd)
```

```
 Time-Series [1:410] from 1956 to 1990: 0.1855 0.2788 0.19 0.0795 0.0281 ...
```

```
str(beer_dat_test_dd)
```

```
 Time-Series [1:12] from 1990 to 1991: -0.0252 0.0209 0.0535 0.0517 0.0603 ...
```

```
# Fit the ARIMA model on trian data
beerdat_arma = auto.arima(beer_dat_train_dd ,
                       seasonal=FALSE,
                       stepwise=FALSE,
                       max.d= 0, # don't allow ARIMA
                       ic = c("aicc", "aic", "bic") ,
                       approximation=FALSE,
                       trace=TRUE)
```

```
 ARIMA(0,0,0)            with zero mean     : -896.8996
 ARIMA(0,0,0)            with non-zero mean : -894.8991
 ARIMA(0,0,1)            with zero mean     : -921.0555
 ARIMA(0,0,1)            with non-zero mean : -919.037
 ARIMA(0,0,2)            with zero mean     : -928.4903
 ARIMA(0,0,2)            with non-zero mean : -926.455
 ARIMA(0,0,3)            with zero mean     : -965.251
 ARIMA(0,0,3)            with non-zero mean : -963.2019
 ARIMA(0,0,4)            with zero mean     : -964.4386
 ARIMA(0,0,4)            with non-zero mean : -962.3789
 ARIMA(0,0,5)            with zero mean     : -979.6284
 ARIMA(0,0,5)            with non-zero mean : -977.5614
 ARIMA(1,0,0)            with zero mean     : -930.5653
 ARIMA(1,0,0)            with non-zero mean : -928.5411
 ARIMA(1,0,1)            with zero mean     : Inf
 ARIMA(1,0,1)            with non-zero mean : Inf
 ARIMA(1,0,2)            with zero mean     : Inf
 ARIMA(1,0,2)            with non-zero mean : Inf
 ARIMA(1,0,3)            with zero mean     : Inf
 ARIMA(1,0,3)            with non-zero mean : Inf
 ARIMA(1,0,4)            with zero mean     : Inf
 ARIMA(1,0,4)            with non-zero mean : Inf
 ARIMA(2,0,0)            with zero mean     : -951.3139
 ARIMA(2,0,0)            with non-zero mean : -949.2745
 ARIMA(2,0,1)            with zero mean     : Inf
 ARIMA(2,0,1)            with non-zero mean : Inf
 ARIMA(2,0,2)            with zero mean     : Inf
 ARIMA(2,0,2)            with non-zero mean : Inf
 ARIMA(2,0,3)            with zero mean     : Inf
 ARIMA(2,0,3)            with non-zero mean : Inf
 ARIMA(3,0,0)            with zero mean     : -1006.679
 ARIMA(3,0,0)            with non-zero mean : -1004.657
 ARIMA(3,0,1)            with zero mean     : Inf
 ARIMA(3,0,1)            with non-zero mean : Inf
 ARIMA(3,0,2)            with zero mean     : Inf
 ARIMA(3,0,2)            with non-zero mean : Inf
 ARIMA(4,0,0)            with zero mean     : -1004.848
 ARIMA(4,0,0)            with non-zero mean : -1002.82
 ARIMA(4,0,1)            with zero mean     : Inf
 ARIMA(4,0,1)            with non-zero mean : Inf
 ARIMA(5,0,0)            with zero mean     : -1021.291
 ARIMA(5,0,0)            with non-zero mean : -1019.29



 Best model: ARIMA(5,0,0)            with zero mean
```

So according to the auto.arima, our best model is an AR(5) model. Let's inpect it.

Inspecting the chosen AR(5) model

Now let's inspect our model:

```r
beerdat_ar5 <- beerdat_arma
beerdat_ar5
```

```
Series: beer_dat_train_dd
ARIMA(5,0,0) with zero mean

Coefficients:
         ar1     ar2     ar3     ar4     ar5
      0.1284  0.0809  0.3327  0.0008  0.2143
s.e.  0.0483  0.0493  0.0466  0.0494  0.0492

sigma^2 estimated as 0.004755:  log likelihood=516.75
AIC=-1021.5   AICc=-1021.29   BIC=-997.4
```

This says that we have a lag 1 difference giving an ARMA(3,1) model.From this, we see all the estimated parameters $\hat{\phi}_1, \hat{\phi}_2, \hat{\phi}_3, \hat{\phi}_4, \hat{\phi}_5$ and $\hat{\sigma}^2$. However we can see that the coefficient $\hat{\phi}_4$ is pretty much zero, so we could assume this is in fact an AR(4) model.

Parameters

We can construct 95% confidence intervals for these:

```r
## Calculate the waltd confidence interval
params_ar5 <- beerdat_ar5$coef # extract parameters
se_ar5 <- sqrt(diag(beerdat_ar5$var.coef)) # extract s.e. for each
z <- qnorm(0.975) # this is the alpha/2 quantile
c.upper <- params_ar5 + z*se_ar5 # upper bound
c.lower <- params_ar5 - z*se_ar5 # lower bound
CI_ar5 <- cbind(params_ar5, se_ar5, c.lower,c.upper)
colnames(CI_ar5)<-c("parameter","s.d.", "2.5 %","97.5 %")
CI_ar5
```

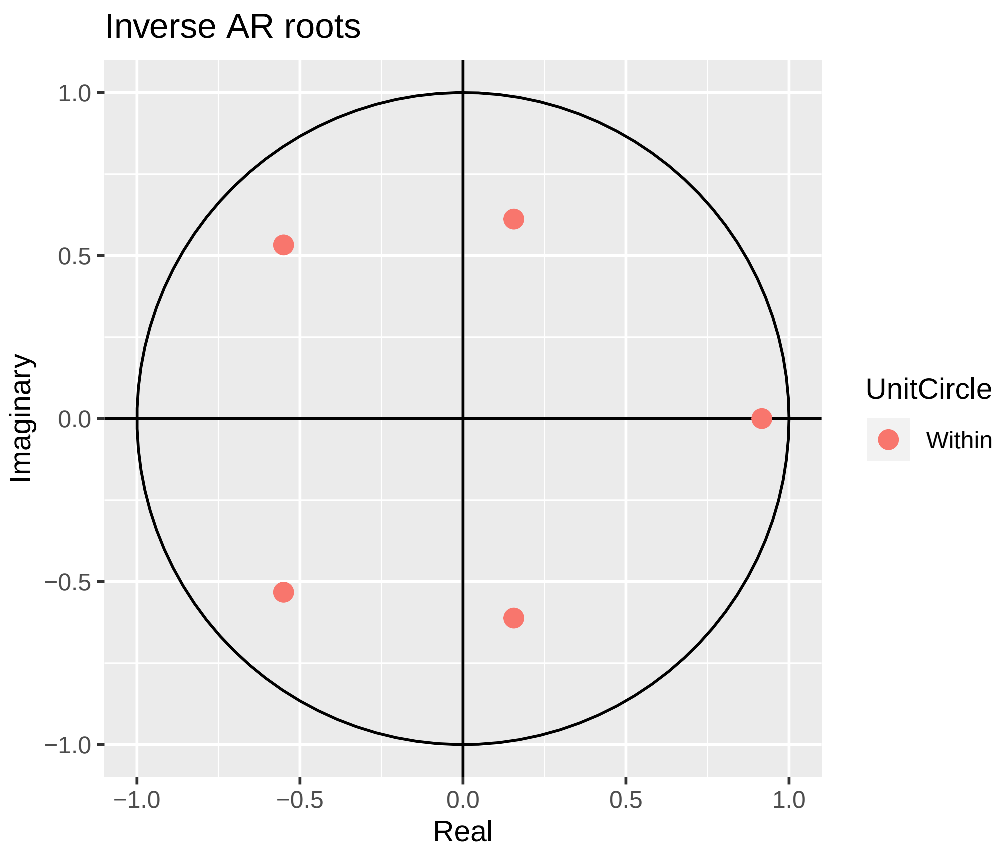
```
       parameter       s.d.       2.5 %    97.5 %
ar1 0.1284392295 0.04829561  0.03378157 0.2230969
ar2 0.0808822273 0.04925638 -0.01565850 0.1774229
ar3 0.3326715559 0.04659216  0.24135259 0.4239905
ar4 0.0007815309 0.04938875 -0.09601864 0.0975817
ar5 0.2142980228 0.04920953  0.11784912 0.3107469
```

Again, note that the ar4 coefficient is estimated to be zero.

Residuals inspection and diagnostics

Let’s first verify the roots of the selected AR(5) process:

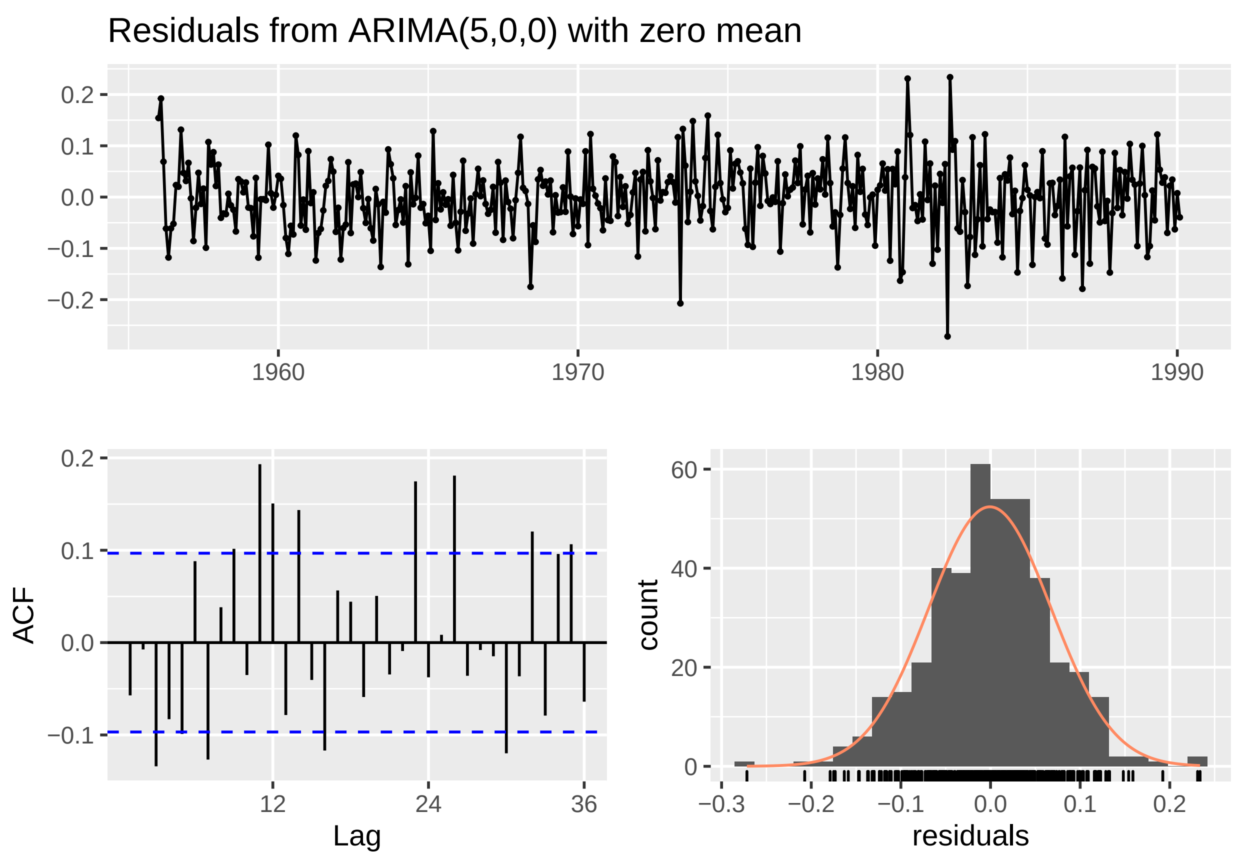
```r
# Plot the toods
autoplot(beerdat_ar5)
```



We see that they are all between, so for sure the process is both stationary and causal. The root that is very close to boundary seems to be the one for the ar4 coefficient.

Let's observe the residuals from the resulting model:

```r
checkresiduals(beerdat_ar5)
```



```
    Ljung-Box test

data:  Residuals from ARIMA(5,0,0) with zero mean
Q* = 93.909, df = 19, p-value = 6.682e-12

Model df: 5.   Total lags used: 24
```

According to the Ljung-Box test, we reject the hupothesis of sationarity, as we would expect. Howerver note that the roots tells us otherwise!

```
# Test with a bunch of different k's ? (bigger augmented versions)
adf.test(residuals(beerdat_ar5))
```

```
Warning in adf.test(residuals(beerdat_ar5)): p-value smaller than printed p-
value
```

```
    Augmented Dickey-Fuller Test

data:  residuals(beerdat_ar5)
Dickey-Fuller = -9.1761, Lag order = 7, p-value = 0.01
alternative hypothesis: stationary
```

```r
kpss.test(residuals(beerdat_ar5))
```

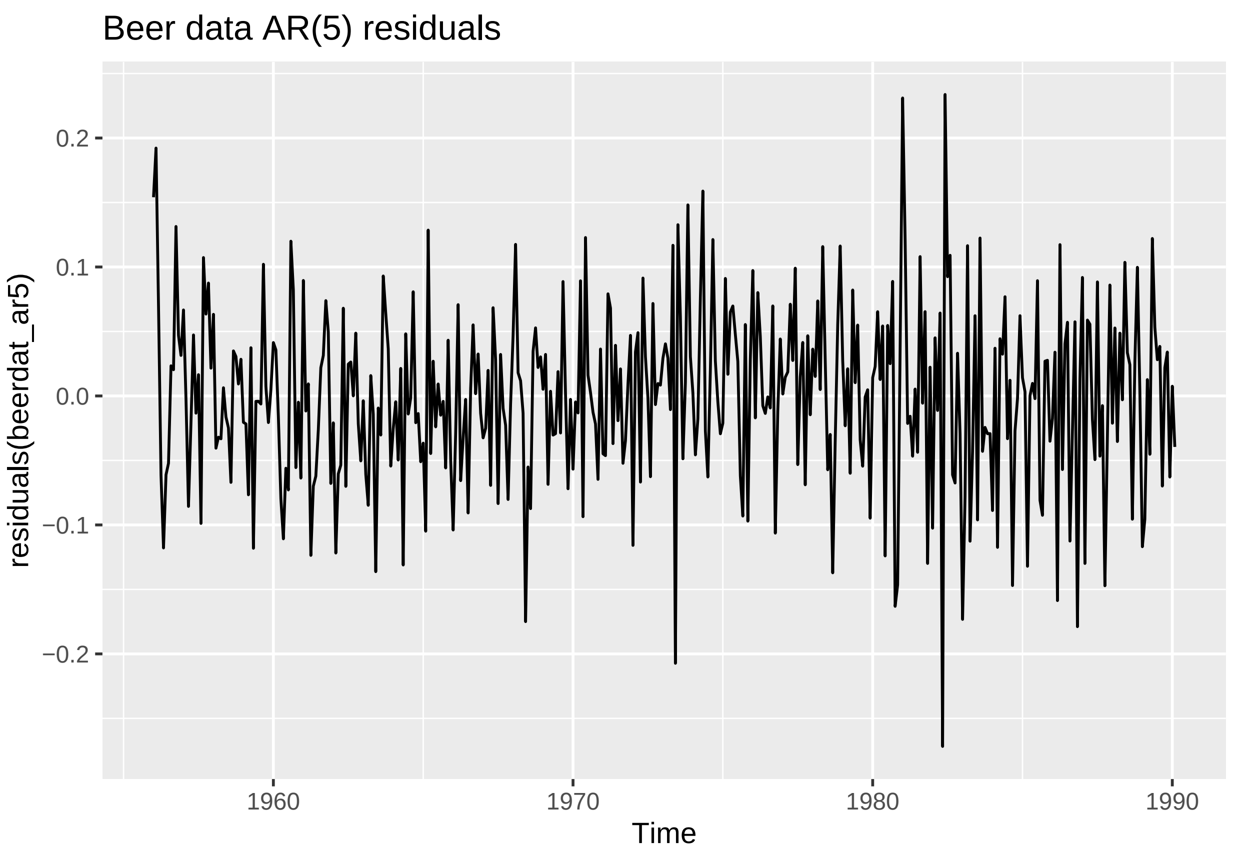
```
Warning in kpss.test(residuals(beerdat_ar5)): p-value greater than printed p-
value
```

```

    KPSS Test for Level Stationarity

data:  residuals(beerdat_ar5)
KPSS Level = 0.19311, Truncation lag parameter = 5, p-value = 0.1
```

- The adf test indicates a stationary process
- The kpss test also indicates a stationary process, as we fail to reject the null.

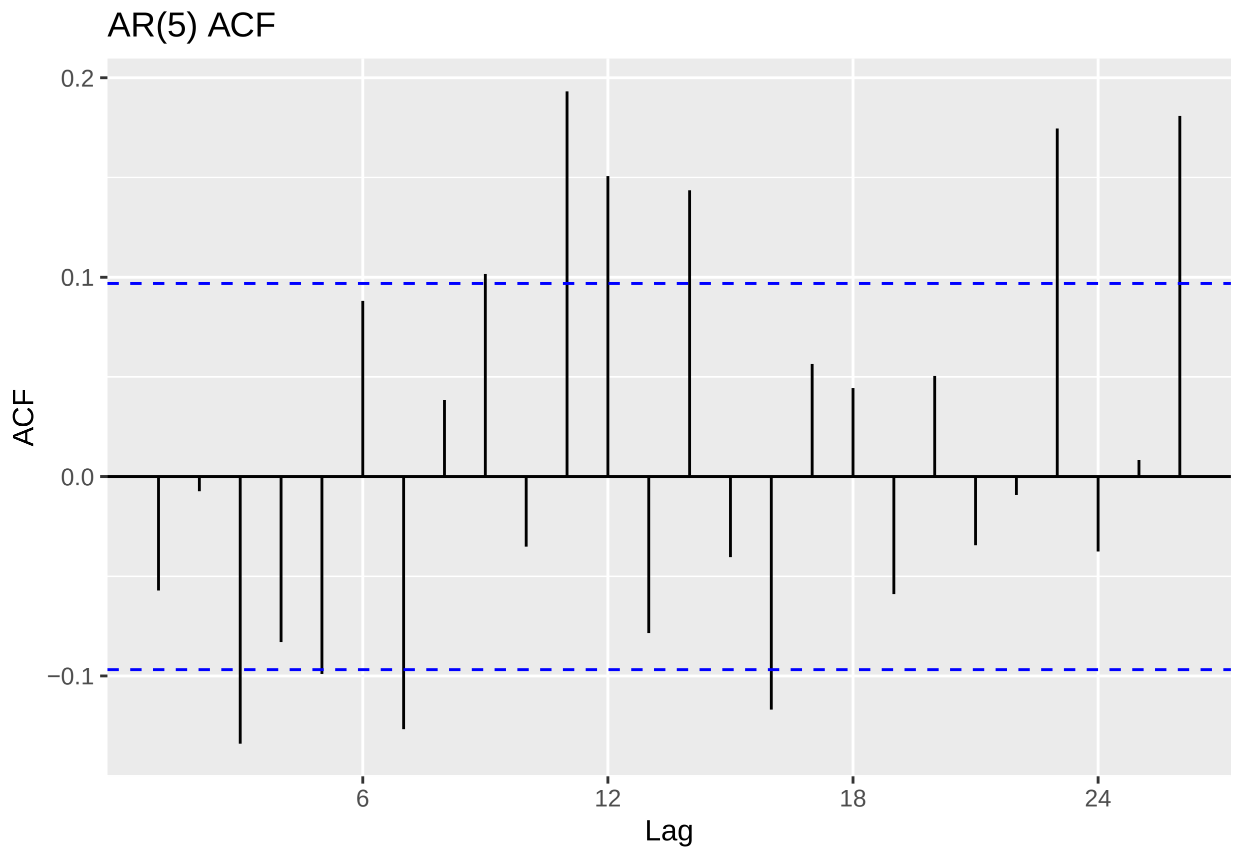
Eureka! It seems that this process is indeed stationary.

```r
# Plot the resulting model
autoplot(residuals(beerdat_ar5)) + ggtitle("Beer data AR(5) residuals")
```

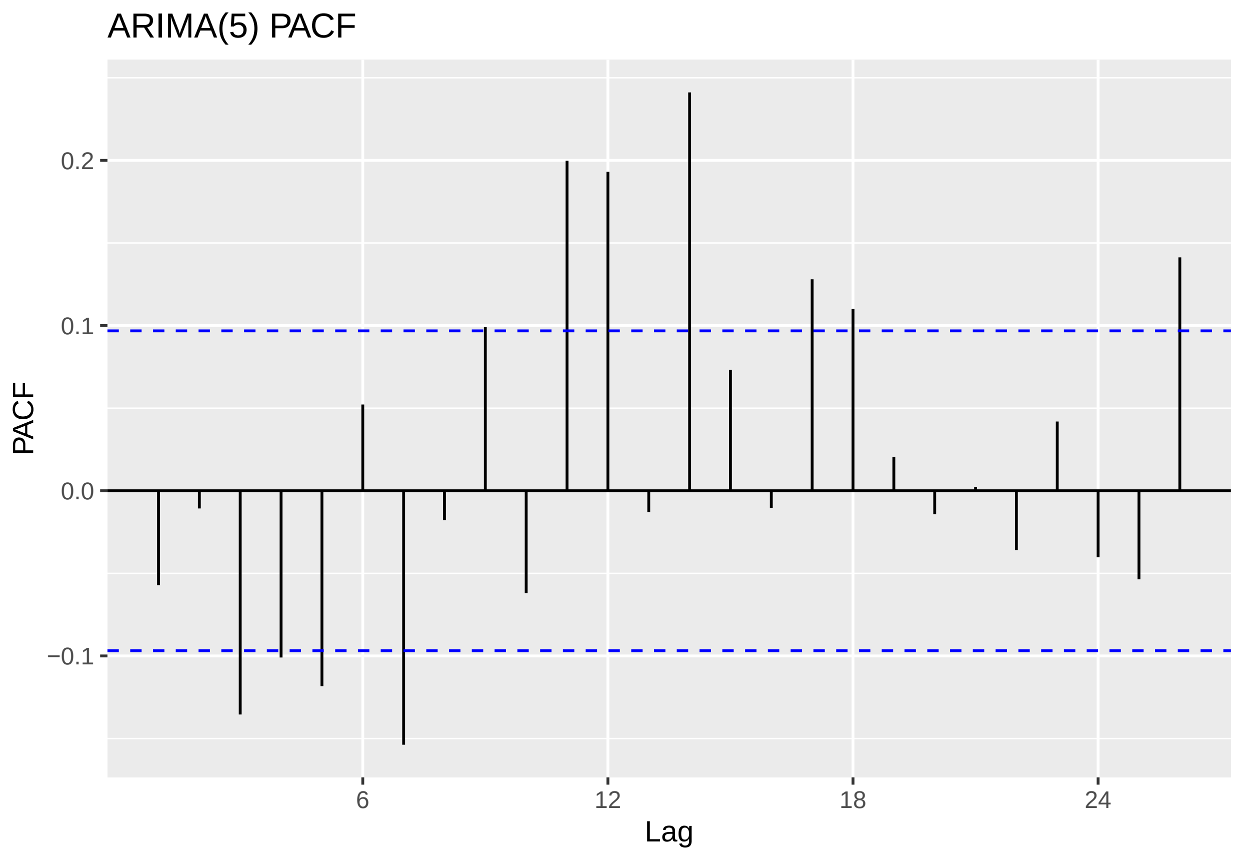


We observe that indeed the model seems to be a lot more centered towards 0, and seems a little bit more like white noise. Let us check this with the ACF and PACF plots:

```
# Plot ACF
ggAcf(residuals(beerdat_ar5)) + ggtitle("AR(5) ACF")
```



```
# Plot PACF
ggPacf(residuals(beerdat_ar5)) + ggtitle("ARIMA(5) PACF")
```



Again, some of the residuals go out bounds, but the guess is that we **could** has estimated better the average (for example with a moving average instead of a quadratic trend).

Forecasting

Just for clarity, let's re-fit the ARIMA(5) model with the trianing data and predict on the test data. However, since we fitted the AR(5) with detrended and deseasonalized data, we will also detrend and deseasonalize the test data

```
# Fit the ARMA model on train data
beerdat_ar5 = auto.arima(beer_dat_train_dd,
                         seasonal=FALSE,
                         stepwise=FALSE,
                         max.d= 0, # don't allow ARIMA
                         ic = c("aicc", "aic", "bic") ,
                         approximation=FALSE,
                         trace=FALSE)


# Forecast ar5 predictions, trend and seasonality
beerdat_ar5_forecasts <- forecast::forecast(beerdat_ar5,h=12) # AR(5) forecasts
forecasted_trend <- forecast::forecast(beerdat_trend_quad, h=12)  # trend
forecasted_season <- forecast::forecast(beerdat_season, h=12) # season
```

Let's see a plot of the deseasoned+detrended data and predictions:

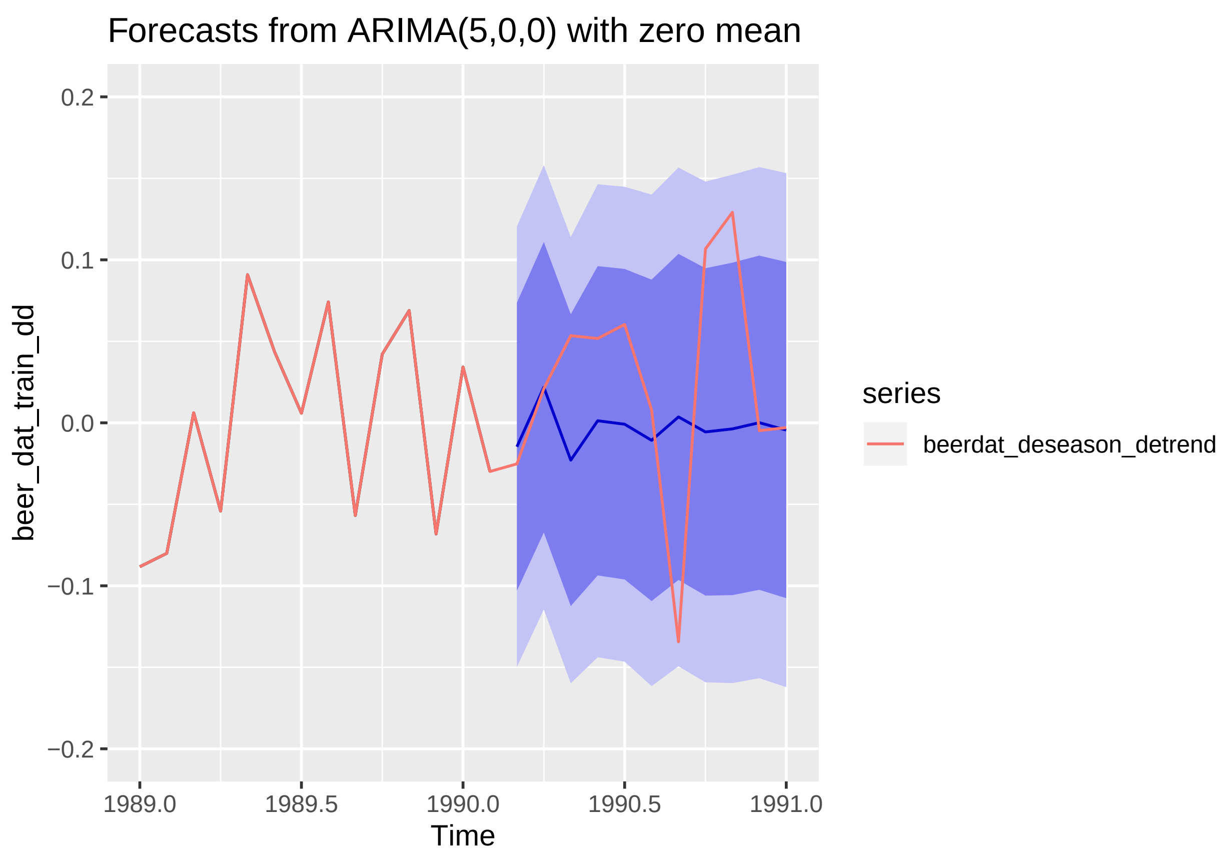
```r
# Print the forecasts along with the boundaries
forecast_table<-print(beerdat_ar5_forecasts) %>%
  mutate(observed=beer_dat_test_dd,
         errors=`Point Forecast`-observed)
```

```
         Point Forecast       Lo 80      Hi 80      Lo 95     Hi 95
Mar 1990  -0.0146694895 -0.10303931 0.07370033 -0.1498195 0.1204805
Apr 1990   0.0218086675 -0.06728707 0.11090441 -0.1144515 0.1580688
May 1990  -0.0229049410 -0.11241529 0.06660541 -0.1597992 0.1139893
Jun 1990   0.0012585370 -0.09360668 0.09612375 -0.1438253 0.1463424
Jul 1990  -0.0008442878 -0.09609642 0.09440784 -0.1465199 0.1448313
Aug 1990  -0.0107530672 -0.10934708 0.08784094 -0.1615396 0.1400335
Sep 1990   0.0036249293 -0.09639938 0.10364924 -0.1493491 0.1565989
Oct 1990  -0.0055925194 -0.10601603 0.09483100 -0.1591770 0.1479920
Nov 1990  -0.0037333040 -0.10570814 0.09824154 -0.1596904 0.1522238
Dec 1990   0.0000847397 -0.10242561 0.10259509 -0.1566913 0.1568608
Jan 1991  -0.0044530742 -0.10757569 0.09866954 -0.1621655 0.1532594
Feb 1991  -0.0010346151 -0.10483030 0.10276107 -0.1597764 0.1577072
```

Now we can plot the forecasted points!

```r
# Plot the predictions
autoplot(beerdat_ar5_forecasts) +
  autolayer(beerdat_deseason_detrend) + xlim(c(1989,1991)) + ylim(c(-0.2,0.2))
```

```
Scale for 'x' is already present. Adding another scale for 'x', which will
replace the existing scale.

Warning: Removed 396 rows containing missing values (geom_path).

Warning: Removed 397 rows containing missing values (geom_path).
```



Above: - The red is the original simulated data - Then the other is the best linear predictor for the data (blue line), along with condifence bounds (80% and 95%) - Most of the points are within the 80$ confidence intervals

Evaluating predictions

We will now compare how many of these were "right", by comparing them to the test data:

```
# Check how many points are inside the 95% confidence intervals
forecast_table %>%
  mutate(outside95=I(observed<`Lo 95` | observed>`Hi 95`)) %>%
  count(outside95)
```

```
# A tibble: 1 x 2
  outside95     n
  <I<lgl>>  <int>
1 FALSE        12
```

```
forecast_table %>%
  mutate(outside80=I(observed<`Lo 80` | observed>`Hi 80`)) %>%
  count(outside80)
```

```
# A tibble: 2 x 2
  outside80     n
  <I<lgl>>  <int>
1 FALSE         9
2 TRUE          3
```

Just as before, we obtain the ame amount of points outside the interval, indicating that the predictive power of both models is roughly the same.

Aside: Restoring the original scale

- **NOTE:** If we were to convert these values back to the original scale, below would be the beginning of the procedure:

Save each forecast table to access individual values and alter the test data:

```r
# Save data.frame objects
forecast_tbl <- print(beerdat_ar5_forecasts) # ar(5)
```

```
         Point Forecast       Lo 80      Hi 80      Lo 95     Hi 95
Mar 1990  -0.0146694895 -0.10303931 0.07370033 -0.1498195 0.1204805
Apr 1990   0.0218086675 -0.06728707 0.11090441 -0.1144515 0.1580688
May 1990  -0.0229049410 -0.11241529 0.06660541 -0.1597992 0.1139893
Jun 1990   0.0012585370 -0.09360668 0.09612375 -0.1438253 0.1463424
Jul 1990  -0.0008442878 -0.09609642 0.09440784 -0.1465199 0.1448313
Aug 1990  -0.0107530672 -0.10934708 0.08784094 -0.1615396 0.1400335
Sep 1990   0.0036249293 -0.09639938 0.10364924 -0.1493491 0.1565989
Oct 1990  -0.0055925194 -0.10601603 0.09483100 -0.1591770 0.1479920
Nov 1990  -0.0037333040 -0.10570814 0.09824154 -0.1596904 0.1522238
Dec 1990   0.0000847397 -0.10242561 0.10259509 -0.1566913 0.1568608
Jan 1991  -0.0044530742 -0.10757569 0.09866954 -0.1621655 0.1532594
Feb 1991  -0.0010346151 -0.10483030 0.10276107 -0.1597764 0.1577072
```

```r
forecast_trend_tbl <- print(forecasted_trend)  # trend
```

```
         Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
Mar 1991       5.054253 4.869076 5.239430 4.770676 5.337830
Apr 1991       5.053322 4.868108 5.238537 4.769688 5.336956
May 1991       5.052379 4.867126 5.237631 4.768687 5.336071
Jun 1991       5.051423 4.866132 5.236713 4.767672 5.335173
Jul 1991       5.050453 4.865123 5.235783 4.766643 5.334264
Aug 1991       5.049471 4.864102 5.234841 4.765600 5.333343
Sep 1991       5.048477 4.863067 5.233887 4.764543 5.332410
Oct 1991       5.047469 4.862018 5.232920 4.763473 5.331465
Nov 1991       5.046449 4.860956 5.231941 4.762389 5.330509
Dec 1991       5.045415 4.859880 5.230950 4.761291 5.329540
Jan 1992       5.044369 4.858791 5.229947 4.760179 5.328560
Feb 1992       5.043310 4.857689 5.228932 4.759053 5.327568
```

```r
forecast_season_tbl <- print(forecasted_season) # season
```

```
         Point Forecast        Lo 80       Hi 80       Lo 95       Hi 95
Mar 1991     0.05454736 -0.051833321  0.16092804 -0.10836662  0.21746135
```

```
Apr 1991    -0.05022896 -0.156609638  0.05615173 -0.21314294  0.11268503
May 1991    -0.08718772 -0.193568399  0.01919297 -0.25010170  0.07572627
Jun 1991    -0.20553507 -0.311915749 -0.09915438 -0.36844905 -0.04262108
Jul 1991    -0.12233736 -0.228718040 -0.01595667 -0.28525134  0.04057663
Aug 1991    -0.06916555 -0.175546232  0.03721513 -0.23207954  0.09374844
Sep 1991    -0.03109403 -0.137474711  0.07528665 -0.19400801  0.13181996
Oct 1991     0.10142187 -0.004958816  0.20780255 -0.06149212  0.26433585
Nov 1991     0.15246770  0.046087018  0.25884838 -0.01044628  0.31538169
Dec 1991     0.23103190  0.124651221  0.33741259  0.06811792  0.39394589
Jan 1992     0.04672604 -0.059613598  0.15306567 -0.11612509  0.20957716
Feb 1992    -0.02137064 -0.127710269  0.08496900 -0.18422176  0.14148048
```

```r
total_forecast <- forecast_tbl + forecast_trend_tbl + forecast_season_tbl
total_forecast
```

```
         Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
Mar 1990       5.094131 4.714203 5.474059 4.512490 5.675772
Apr 1990       5.024902 4.644211 5.405593 4.442094 5.607710
May 1990       4.942286 4.561143 5.323430 4.358786 5.525786
Jun 1990       4.847146 4.460609 5.233683 4.255397 5.438895
Jul 1990       4.927272 4.540309 5.314235 4.334872 5.519672
Aug 1990       4.969553 4.579208 5.359897 4.371981 5.567125
Sep 1990       5.021008 4.629193 5.412823 4.421186 5.620829
Oct 1990       5.143298 4.751043 5.535554 4.542804 5.743793
Nov 1990       5.195183 4.801335 5.589031 4.592252 5.798114
Dec 1990       5.276532 4.882106 5.670958 4.672717 5.880347
Jan 1991       5.086642 4.691602 5.481683 4.481888 5.691396
Feb 1991       5.020905 4.625148 5.416662 4.415055 5.626755
```

The above would be the forecast points in the original scale.

CATS

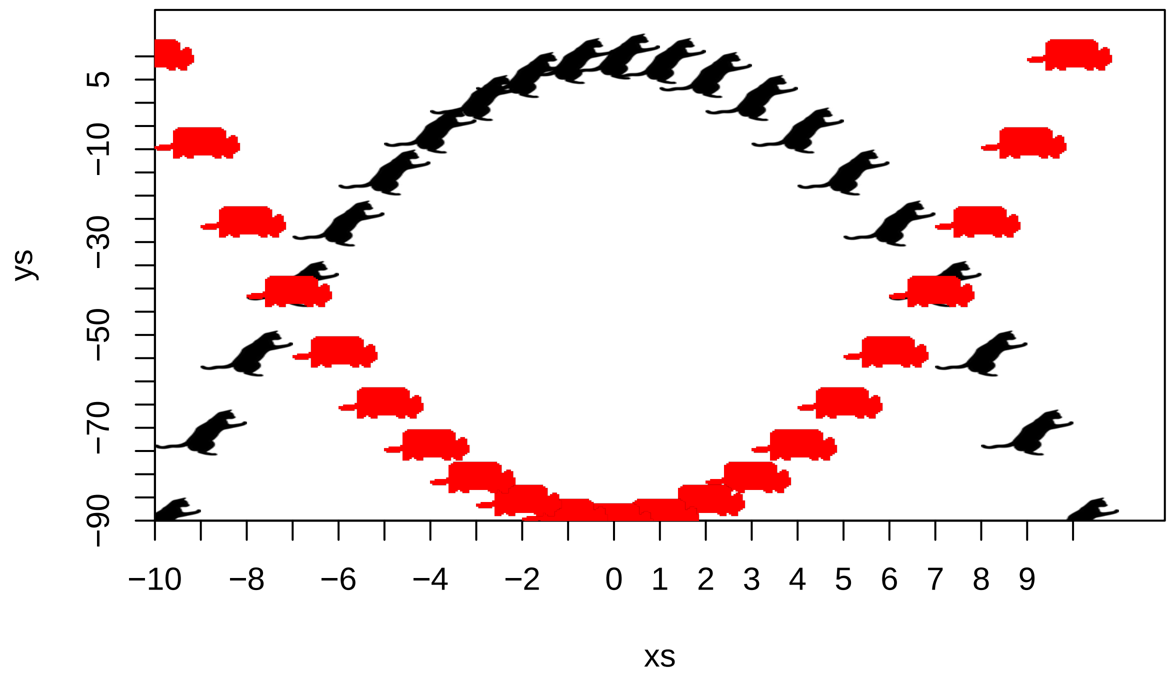
Final Note: Although this semester has been incredibly stressful and has given me anxiety to the point of not being able to sleep every night, and having such an accelerated heart rate that I was forced to start taking medication to prevent a cardica arrest (along with daily exercise), and the material amount was ridiculous given the COVID19 situation… I am still very glad and happy that I took this course, and proud and satisfied with everything I learned, even after the tremendous amount of effort it costed me. That being said, I would like to thank Professor Steele for an amazing semester, and for caring about his students.

- PD1: It would be greatly appreciated if you could type your notex in Latex!
- PD2: Here are some random plots with cats instead of points ^_^

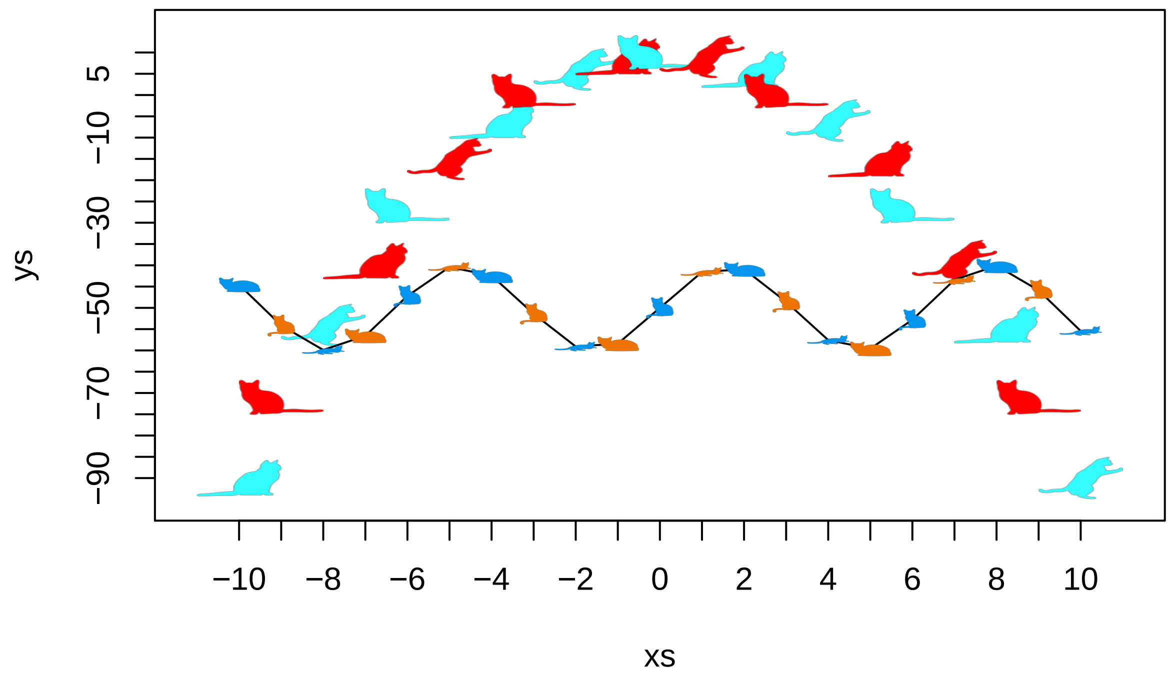
```r
library(CatterPlots)
```

```
Welcome to CatterPlots.
```

```r
x <- -10:10
y <- -x^2 + 10
purr <- catplot(xs=x, ys=y, cat=3, catcolor='#000000FF')
cats(purr, -x, -y, cat=11, catcolor='#FF0000')
```

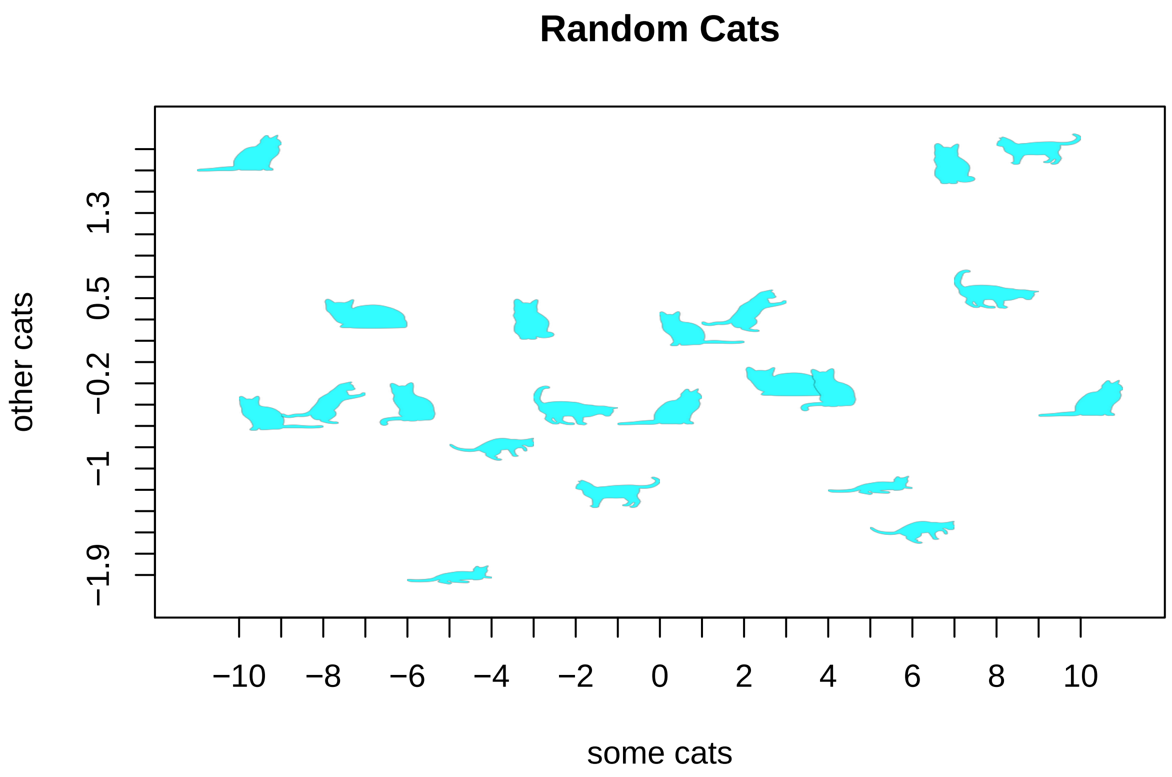


```
# for more fun ...
meow <- multicat(xs=x, ys=y, cat=c(1,2,3), catcolor=list('#33FCFF','#FF0000'), canvas=c(-0.1,1.1, -0.1,
morecats(meow, x, 10*sin(x)+40, size=0.05, cat=c(4,5,6), catcolor=list('#0495EE','#EE7504'), type="line
```



```
-0.025  0.429402111088937  0.025  0.479402111088937
0.025  0.333788151475824  0.075  0.383788151475824
0.075  0.276064175337662  0.125  0.326064175337662
0.125  0.309301340128121  0.175  0.359301340128121
0.175  0.402941549819893  0.225  0.452941549819893
0.225  0.470892427466314  0.275  0.520892427466314
0.275  0.450680249530793  0.325  0.500680249530793
0.325  0.360887999194013  0.375  0.410887999194013
0.375  0.284070257317432  0.425  0.334070257317432
0.425  0.29085290151921  0.475  0.34085290151921
0.475  0.375  0.525  0.425
0.525  0.45914709848079  0.575  0.50914709848079
0.575  0.465929742682568  0.625  0.515929742682568
0.625  0.389112000805987  0.675  0.439112000805987
0.675  0.299319750469207  0.725  0.349319750469207
0.725  0.279107572533686  0.775  0.329107572533686
0.775  0.347058450180107  0.825  0.397058450180107
0.825  0.440698659871879  0.875  0.490698659871879
0.875  0.473935824662338  0.925  0.523935824662338
0.925  0.416211848524176  0.975  0.466211848524176
0.975  0.320597888911063  1.025  0.370597888911063
```

```
# random cats
meow <- multicat(xs=x, ys=rnorm(21),
                 cat=c(1,2,3,4,5,6,7,8,9,10),
                 catcolor=list('#33FCFF'),
                 canvas=c(-0.1,1.1, -0.1, 1.1),
                 xlab="some cats", ylab="other cats", main="Random Cats")
```


Random Cats
other cats
some cats