Zeitreihenanalyse und Prognose

Zeitreihenanalyse

R Package für die Time Serie Analyse

```
motor.df <- read.csv("~/Dokumente/R-Alle Data1/motororg.dat", header = TRUE)
```

```
head(motor.df)
```

```
##   complaints
## 1         27
## 2         34
## 3         31
## 4         24
## 5         18
## 6         19
```

Datum-Spalte einfügen

```
motor.df<-motor.df%>%
mutate(Date= seq.Date(from=as.Date("2016-01-01"), by="months",length.out = 48))%>%
  mutate(jahr=as.factor(year(Date)))
```

Daten von data.Frame zu tsibble transformieren

```
motor_tsibble<-as_tsibble(motor.df,key=complaints,index = Date)
head(motor_tsibble)
```

```
## # A tsibble: 6 x 3 [1D]
## # Key:       complaints [4]
##   complaints Date       jahr
##        <int> <date>     <fct>
## 1          4 2017-11-01 2017
## 2          6 2019-04-01 2019
## 3          9 2019-05-01 2019
## 4         10 2017-05-01 2017
## 5         10 2017-10-01 2017
## 6         10 2018-07-01 2018
```

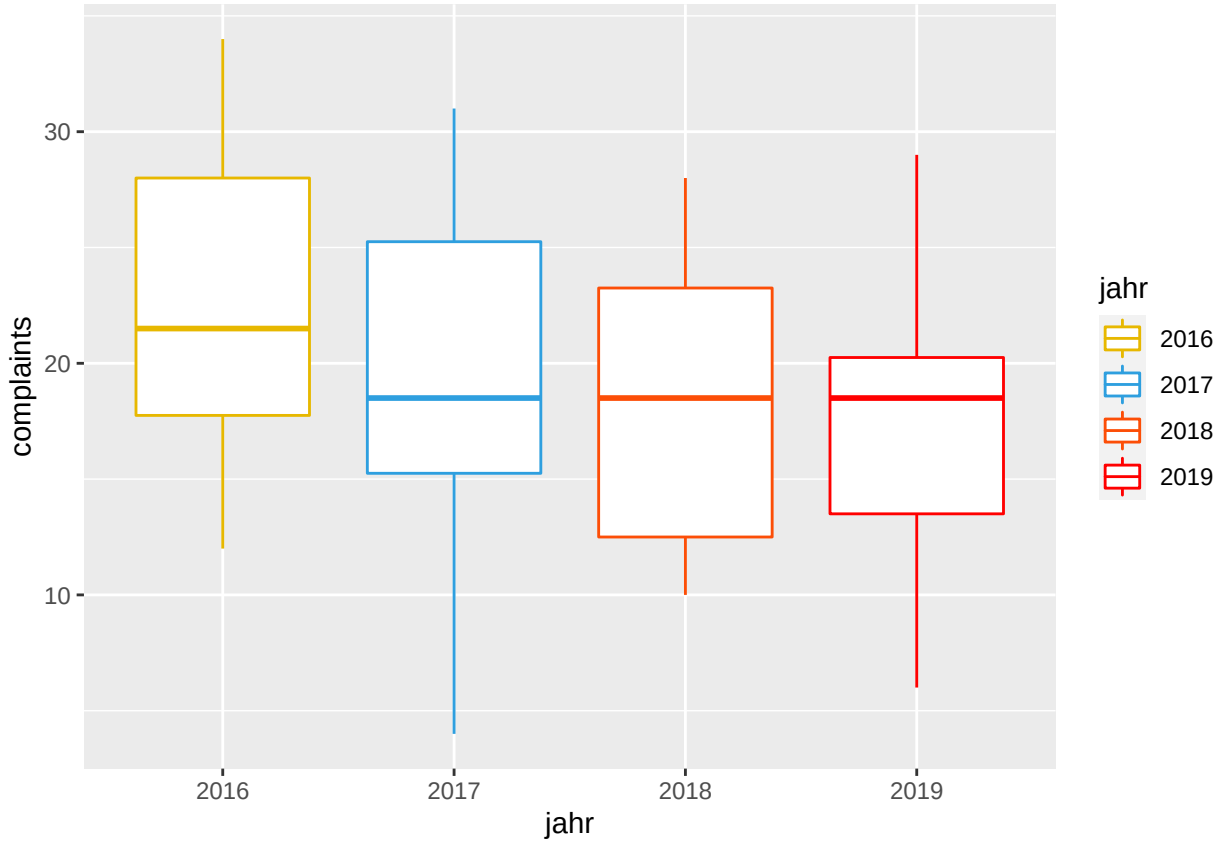
Daten-Verteilung visualisieren

```
Farben <- c("#E7B800", "#2E9FDF", "#FC4E07","red","green")

p <- ggplot(motor.df, aes(x =jahr, y = complaints))
bxp <- p + geom_boxplot(aes(color = jahr)) +
  scale_color_manual(values = Farben)
dp <- p + geom_dotplot(aes(color = jahr, fill = jahr),
```

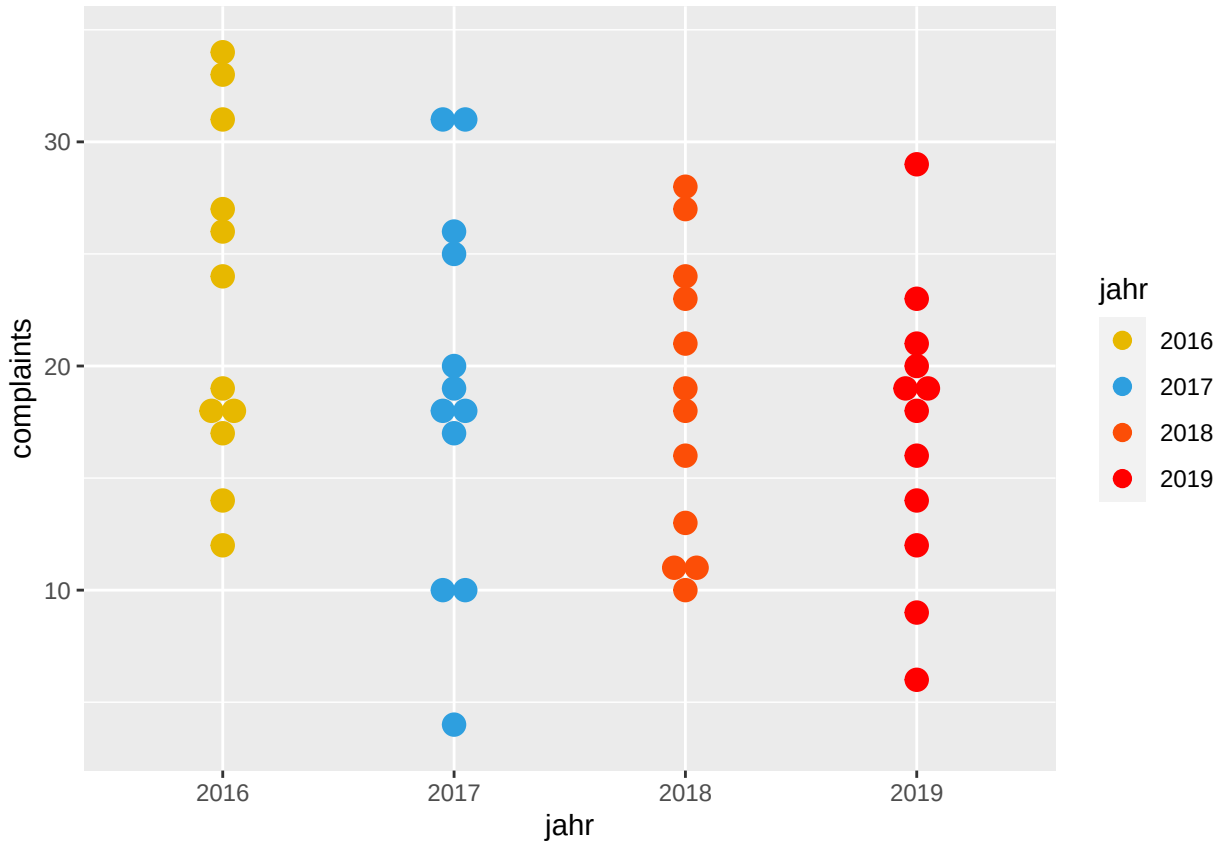
```
                      binaxis='y', stackdir='center') +
  scale_color_manual(values = Farben) +
  scale_fill_manual(values = Farben)
```

```
bxp
```



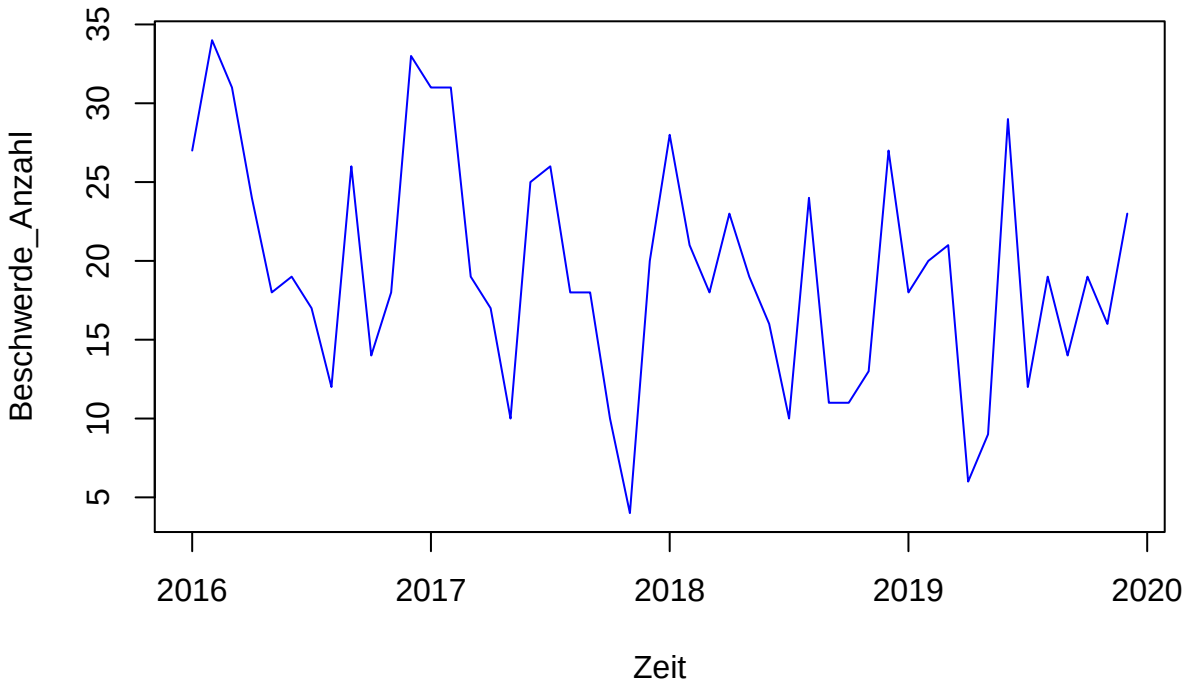

```
dp
```

```
## `stat_bindot()` using `bins = 30`. Pick better value with `binwidth`.
```



###Hier wurde ein `tserie` kreiert

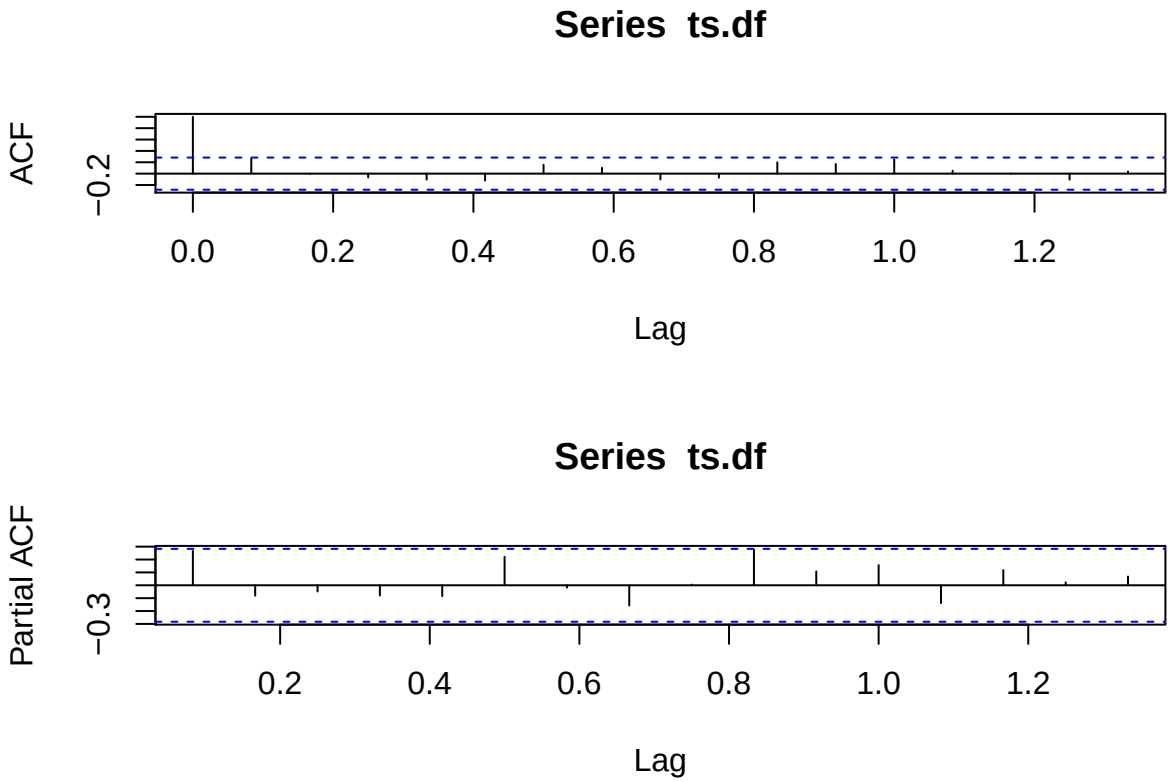
```
ts.df<- ts(motor.df$complaints,start = c(2016,01,01), freq=12)
plot(ts.df,col="blue",xlab="Zeit",ylab="Beschwerde_Anzahl")
```



###

Stationarität Prüfung ### Keine Autokorrelation vorhanden

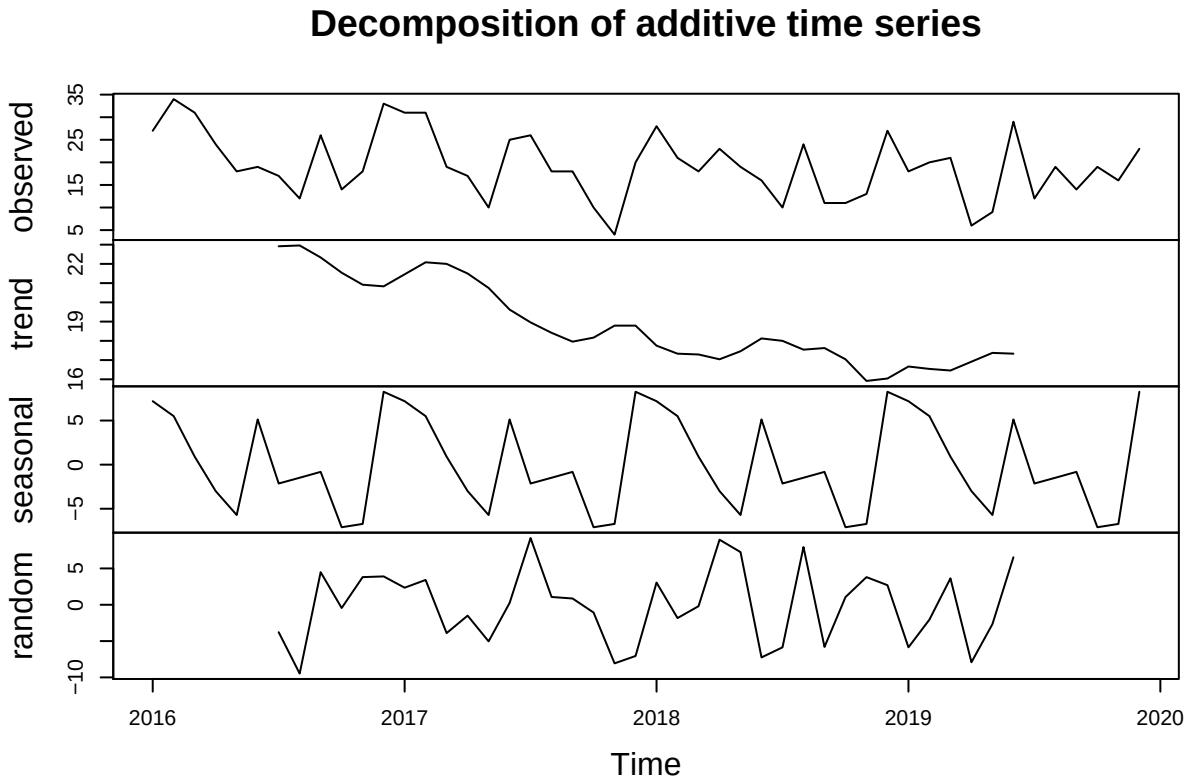
```
layout(1:2)
acf(ts.df)
pacf(ts.df)
```



Die Zerlegung der Time Serie in Trend, Saisonalität und Random, dabei kann man die jährliche Saisonalität erkennen

Wir haben einen senkenden Trend

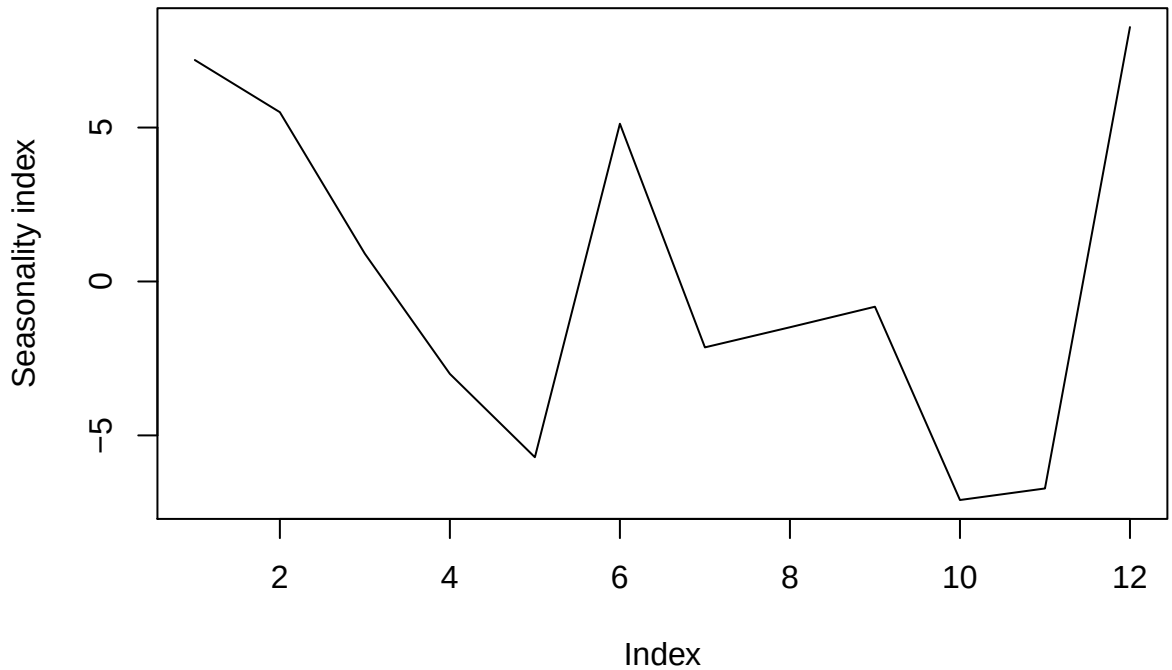
```
ts.decom<-decompose(ts.df)
plot(decompose(ts.df))
```



###

Wir haben eine deutliche 12 monatige Saisonalität ### die 6 monataige Saisonalität ist auch vorhanden

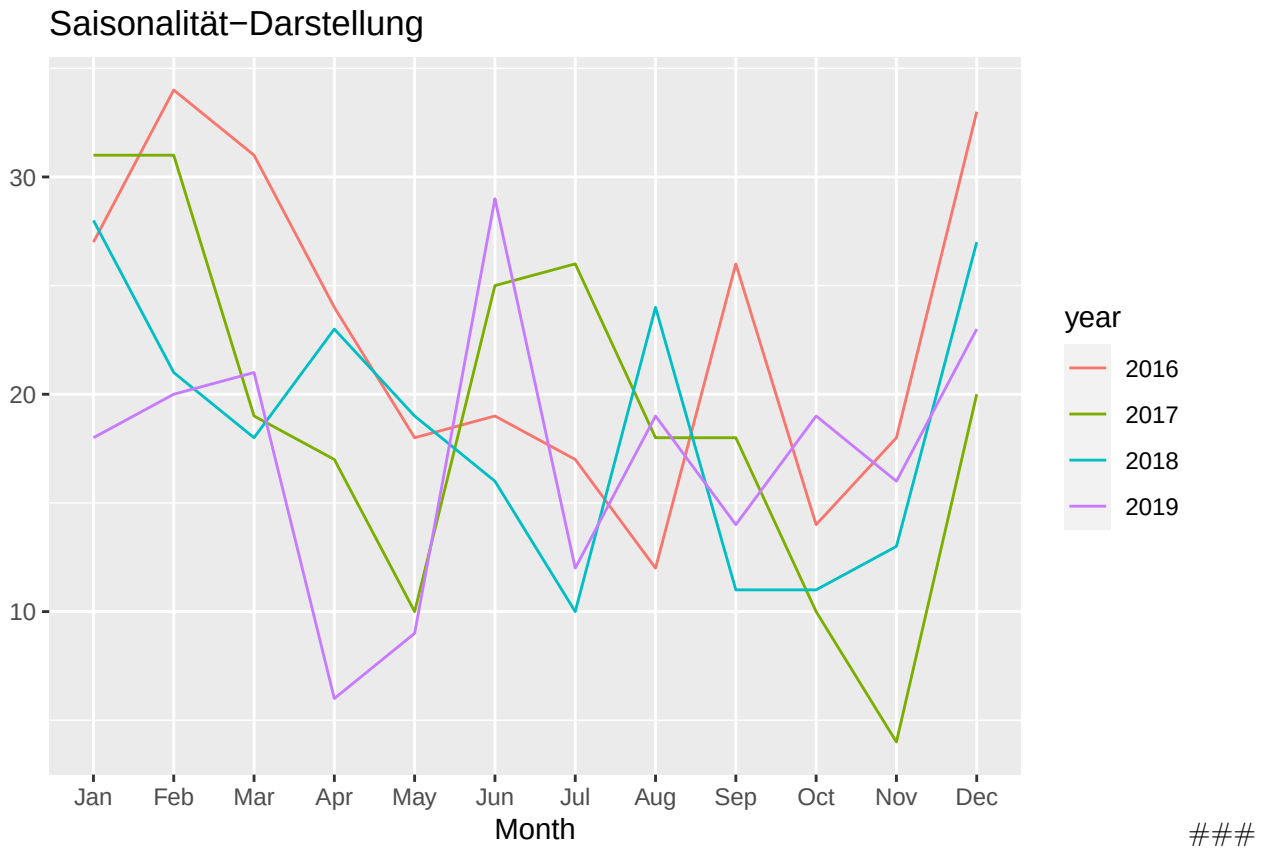
```
plot(ts.decom$figure,type="l",ylab="Seasonality index")
```



###

Die jährliche Saisonalität ist ganz klar zu erkennen, während die 6 monatige Saisonalität nicht ganz deutlich zu erkennen sei.

```
ggseasonplot(ts.df)+ggtitle("Saisonalität-Darstellung")
```



###

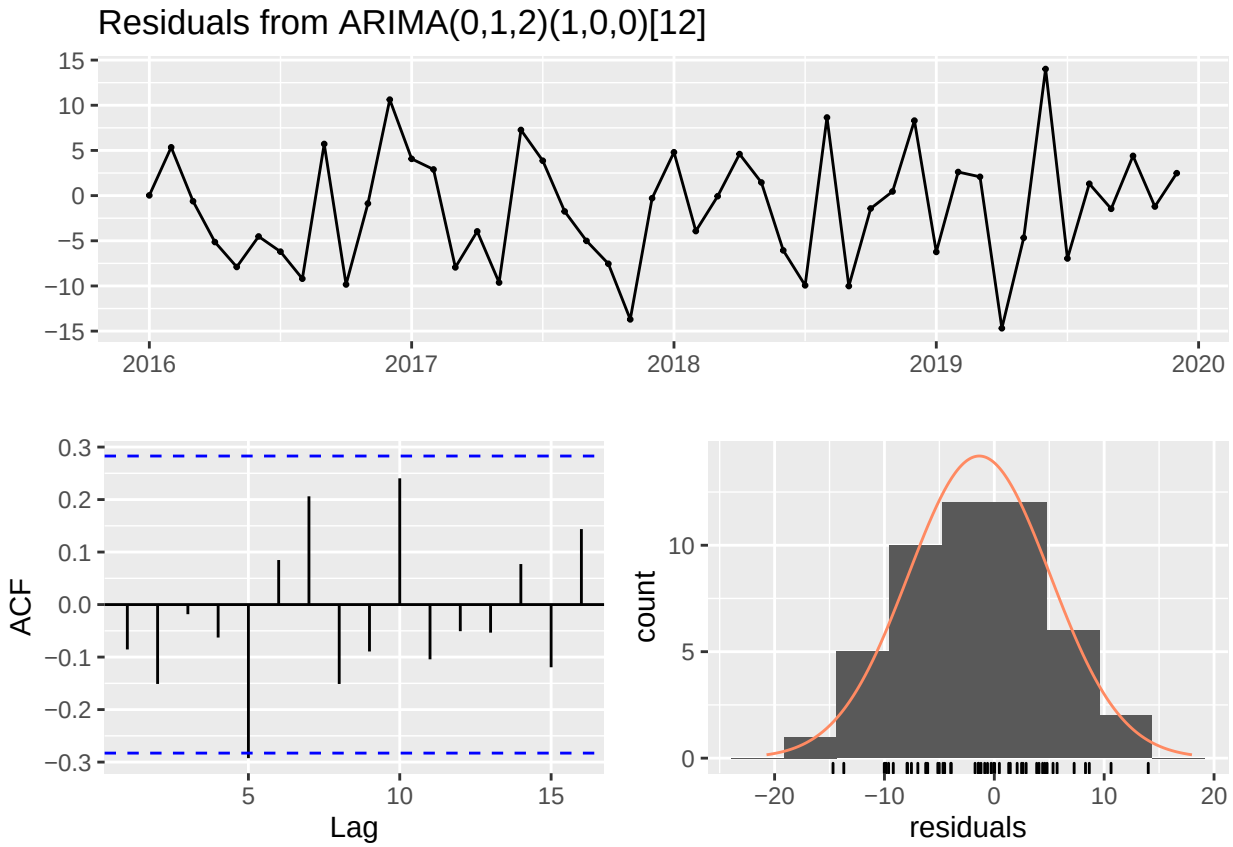
Mit hilfe auto.arima Funktion suchen wir die optimale Order für die Parameter d, q und p

```
df.arima<-auto.arima(ts.df,seasonal = TRUE,
           D=NA,d=NA,
    ic = c("aicc", "aic", "bic"),
  stepwise = TRUE,nmodels = 100,  seasonal.test = c("seas", "ocsb", "hegy", "ch"),
 allowdrift = TRUE,
  allowmean = TRUE,
  lambda = TRUE,
  biasadj = FALSE,
  parallel = FALSE,
  trace = FALSE)
df.arima
```

```
## Series: ts.df
## ARIMA(0,1,2)(1,0,0)[12]
## Box Cox transformation: lambda= TRUE
##
## Coefficients:
##           ma1      ma2    sar1
##       -0.6564  -0.2707  0.3509
## s.e.   0.1528   0.1546  0.1578
##
## sigma^2 estimated as 46.54:  log likelihood=-156.89
## AIC=321.78   AICc=322.74   BIC=329.18
```

Die Resudien der Zeitreihe haben ein p_Value von 3%, sodass die Nullhypothese beibehalten kann. Das Diagramm bestätigt, dass die Residuen weiß Rausch sind. Schließlich kann man mit der Prognose Anfangen.

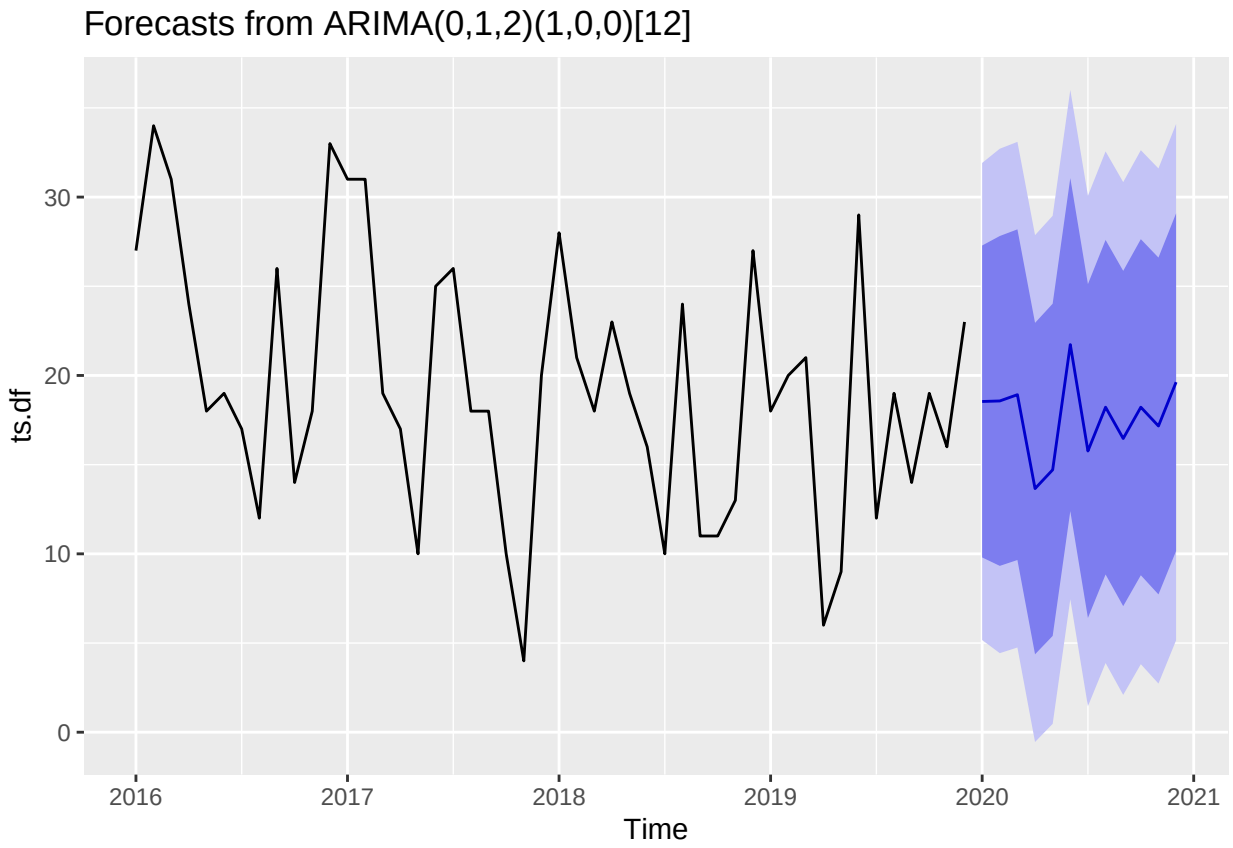
```
checkresiduals(df.arima)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,2)(1,0,0)[12]
## Q* = 14.984, df = 7, p-value = 0.0362
##
## Model df: 3.   Total lags used: 10
```

Prognose durchführen

```
df.arima%>%forecast(h=12)%>%autoplot()
```



Prognose Tabelle

```
df.arima%>%forecast(h=12)
```

```
##          Point Forecast     Lo 80    Hi 80      Lo 95    Hi 95
## Jan 2020       18.54142  9.798248 27.28459  5.1698913 31.91295
## Feb 2020       18.57151  9.326461 27.81657  4.4324259 32.71060
## Mar 2020       18.92243  9.655432 28.18942  4.7497820 33.09507
## Apr 2020       13.65875  4.369865 22.94763 -0.5473721 27.86487
## May 2020       14.71148  5.400763 24.02220  0.4719656 28.95100
## Jun 2020       21.72972 12.397213 31.06223  7.4568827 36.00256
## Jul 2020       15.76422  6.409976 25.11846  1.4581394 30.07030
## Aug 2020       18.22060  8.844674 27.59653  3.8813573 32.55985
## Sep 2020       16.46604  7.068479 25.86361  2.0937097 30.83838
## Oct 2020       18.22060  8.801453 27.63975  3.8152564 32.62595
## Nov 2020       17.16787  7.727181 26.60855  2.7295839 31.60615
## Dec 2020       19.62425 10.162076 29.08642  5.1531048 34.09539
```