Homework 2 Exercise 5

Mads Nielsen & Daniel Winkler

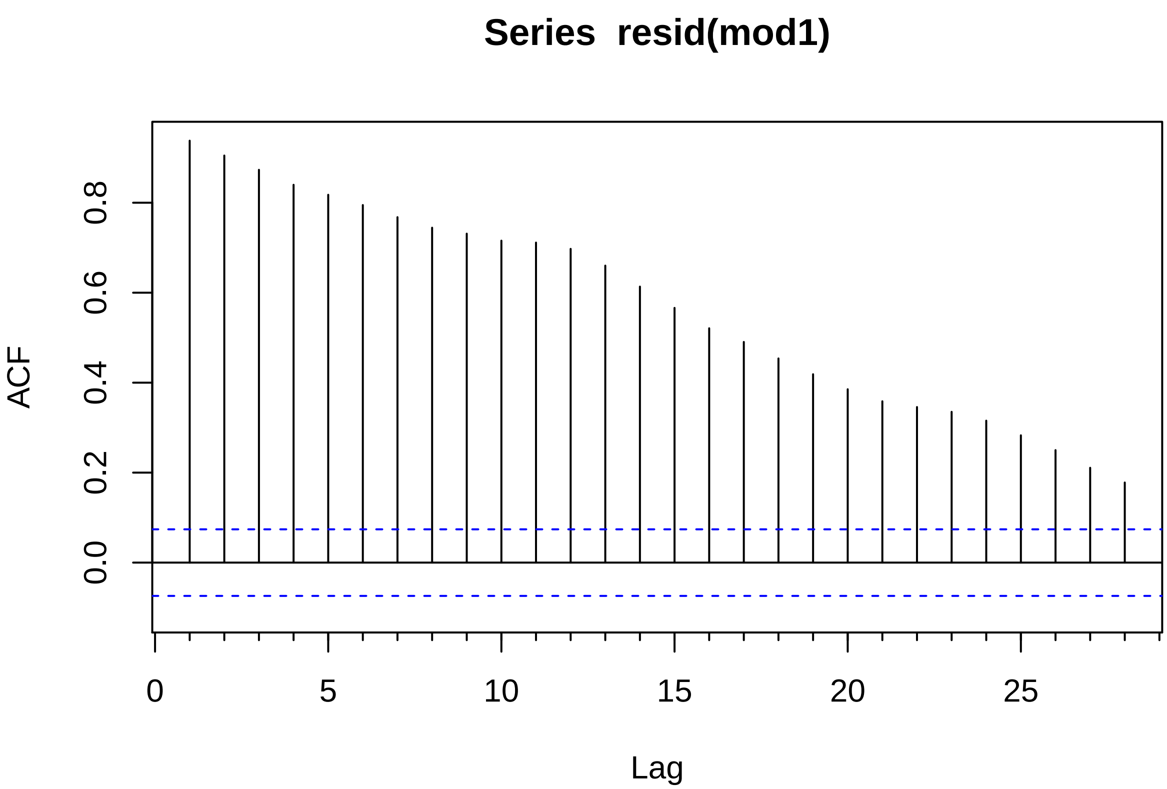
(a) Estimate the model with $\phi(B) = \theta(B) = 1$

We first create a time series for the house data and variables for the seasonal dummies (Jan-Nov) and a trend. Next the model

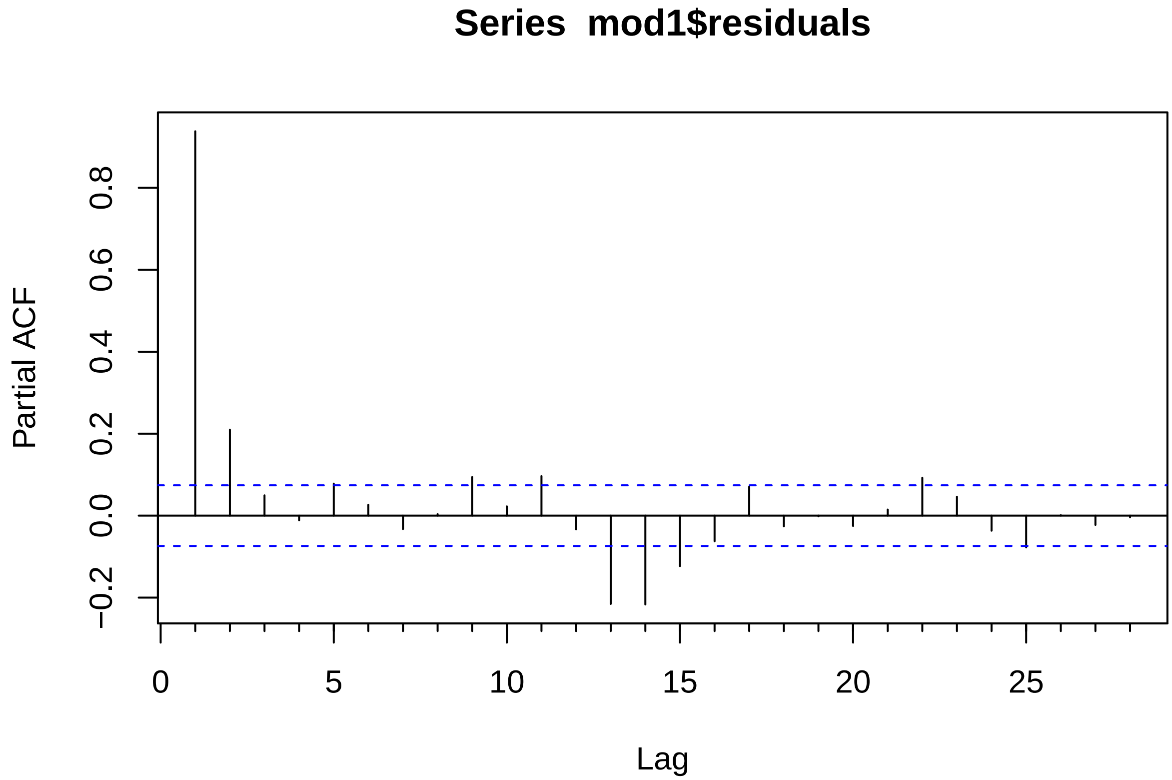
$$y_t = c + \alpha t + \gamma_t + \varepsilon_t$$

is estimated. We look at the ACF and PACF of the error term, ε_t to determine the lag structure for our ARMA model. Since the ACF decays slowly instead of cutting off at a specific lag and the PACF cuts of after lag 2 an AR(2) model might be appropriate [ARMA(2,0)]. Looking at the ACF and PACF of the residuals of the AR(2) we observe that there is no significant (partial) autocorrelation in the error terms, at least for the first few lags.

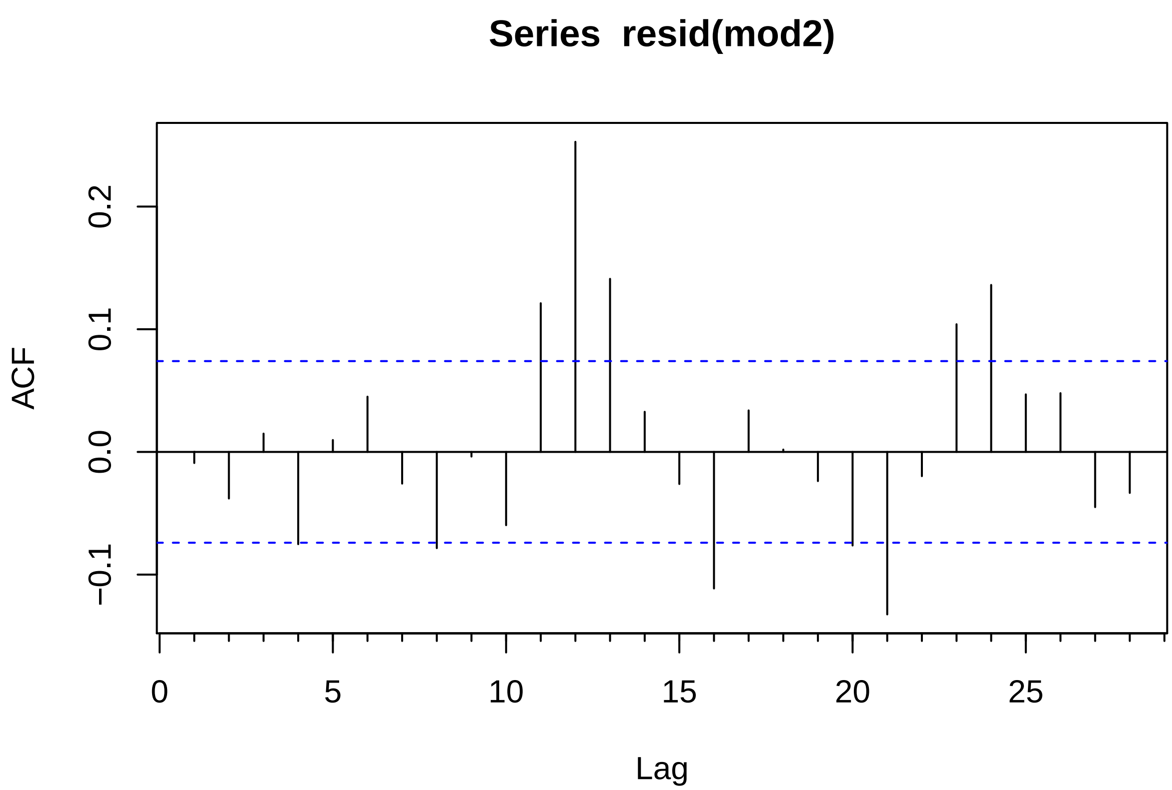
```
house <- ts(house1$HOUSTNSA, start = c(1959,1) , frequency = 12)
seasons <- seasonaldummy(house)
trend <- 1:length(house)

mod1 <- lm(house ~  seasons + trend)
Acf(resid(mod1))
```



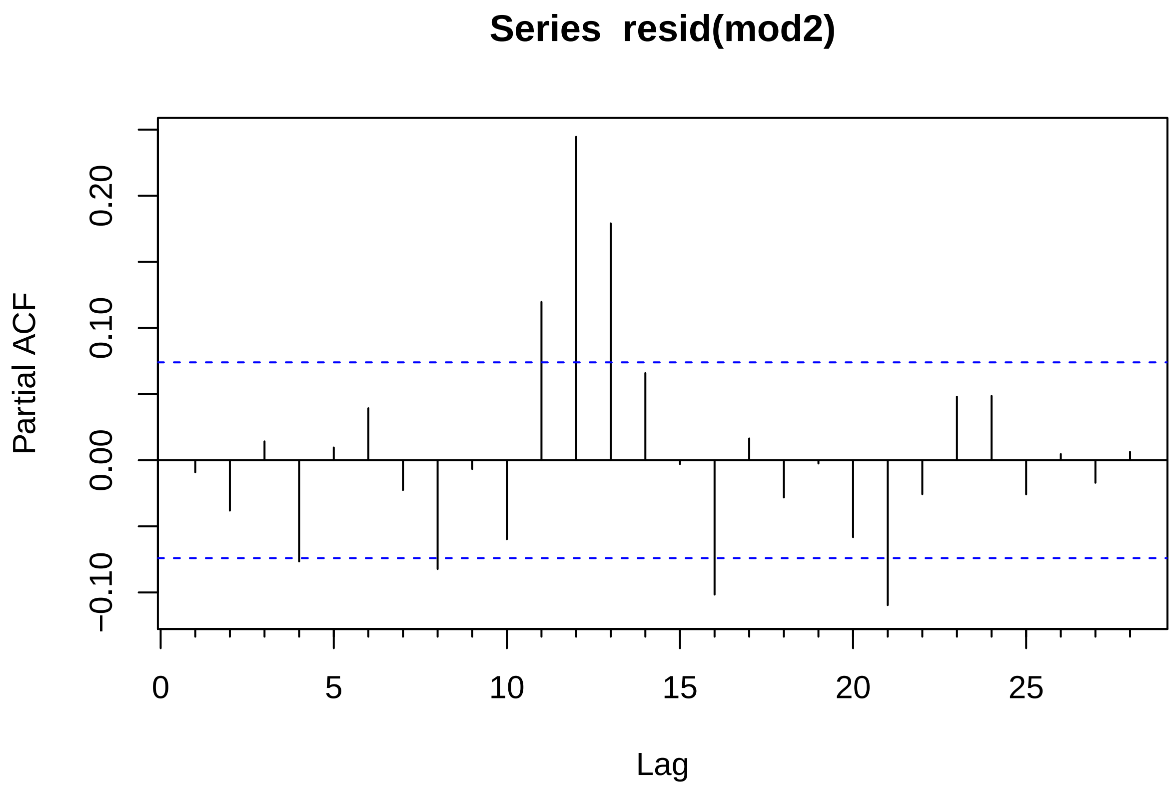

```
Pacf(mod1$residuals)
```




```
mod2 <- arima(resid(mod1), order = c(2,0,0))
Acf(resid(mod2))
```



```
Pacf(resid(mod2))
```



(b) Determine the lag-structure by AIC and BIC

We create a grid of all possible combinations of lag structures from 0 to 3 for the AR and MA parts. The differencing part is set to 0 for all combinations. The ARMA model is estimated using conditional-sum-of-squares to find starting values, then maximum likelihood. This yields and error for the ARMA(3, 3) and ARMA(3, 2) because the algorithm does not converge. Using the ML estimation with conditional-sum-of-squares yields a minimum AIC for the ARMA(3, 3) model. However, since the algorithm did not converge this result might be erroneous (the BIC prefers the ARMA(1,1)). Using ML both AIC and BIC prefer the ARMA(1, 1) model, however there are possible convergence problems for the last three models. Therfore, the ARMA(1, 1) can be seen as an alternative to the AR(2) implied by the correlogram.

```
p <- 0:3
i <- 0
q <- 0:3
lag_grid <- as.matrix(expand.grid(p, i, q))

ICs <- matrix(NA, ncol = 2, nrow = nrow(lag_grid))
for (i in 1:nrow(lag_grid)){
  mod <- arima(resid(mod1), order = lag_grid[i, ], method = "CSS-ML")
  ICs[i, 1] <- AIC(mod)
  ICs[i, 2] <- BIC(mod)
}
```

```
## Warning in arima(resid(mod1), order = lag_grid[i, ], method = "CSS-ML"):
## possible convergence problem: optim gave code = 1

## Warning in arima(resid(mod1), order = lag_grid[i, ], method = "CSS-ML"):
```

```
## possible convergence problem: optim gave code = 1
```

```
names <- c()
for(i in 1:nrow(lag_grid)){
  names[i] <- paste0("AR ", lag_grid[i,1], ", MA ", lag_grid[i,3])
}
rownames(ICs) <- names
kable(ICs, col.names = c("AIC", "BIC"), row.names = TRUE)
```

	AIC	BIC
AR 0, MA 0	6860.449	6869.554
AR 1, MA 0	5376.460	5390.118
AR 2, MA 0	5347.057	5365.267
AR 3, MA 0	5347.454	5370.216
AR 0, MA 1	6242.422	6256.079
AR 1, MA 1	5345.510	5363.720
AR 2, MA 1	5347.508	5370.271
AR 3, MA 1	5347.056	5374.371
AR 0, MA 2	5955.998	5974.208
AR 1, MA 2	5347.512	5370.275
AR 2, MA 2	5347.173	5374.489
AR 3, MA 2	5348.764	5380.631
AR 0, MA 3	5738.008	5760.771
AR 1, MA 3	5349.437	5376.752
AR 2, MA 3	5348.895	5380.762
AR 3, MA 3	5336.723	5373.143

```
# Change order to c(3, 0, 2) for second error in loop
error_mod <- arima(resid(mod1), order=c(3,0,3), method = "CSS-ML") # gives error!
```

```
## Warning in arima(resid(mod1), order = c(3, 0, 3), method = "CSS-ML"):
## possible convergence problem: optim gave code = 1
```

```
# Prefered by AIC
which(ICs[,1]==min(ICs[,1]))
```

```
## AR 3, MA 3
##         16
```

```
# Prefered by BIC
which(ICs[,2]==min(ICs[,2]))
```

```
## AR 1, MA 1
##          6
```

(c) Plots and final specification of the model

The final specification of the model is:

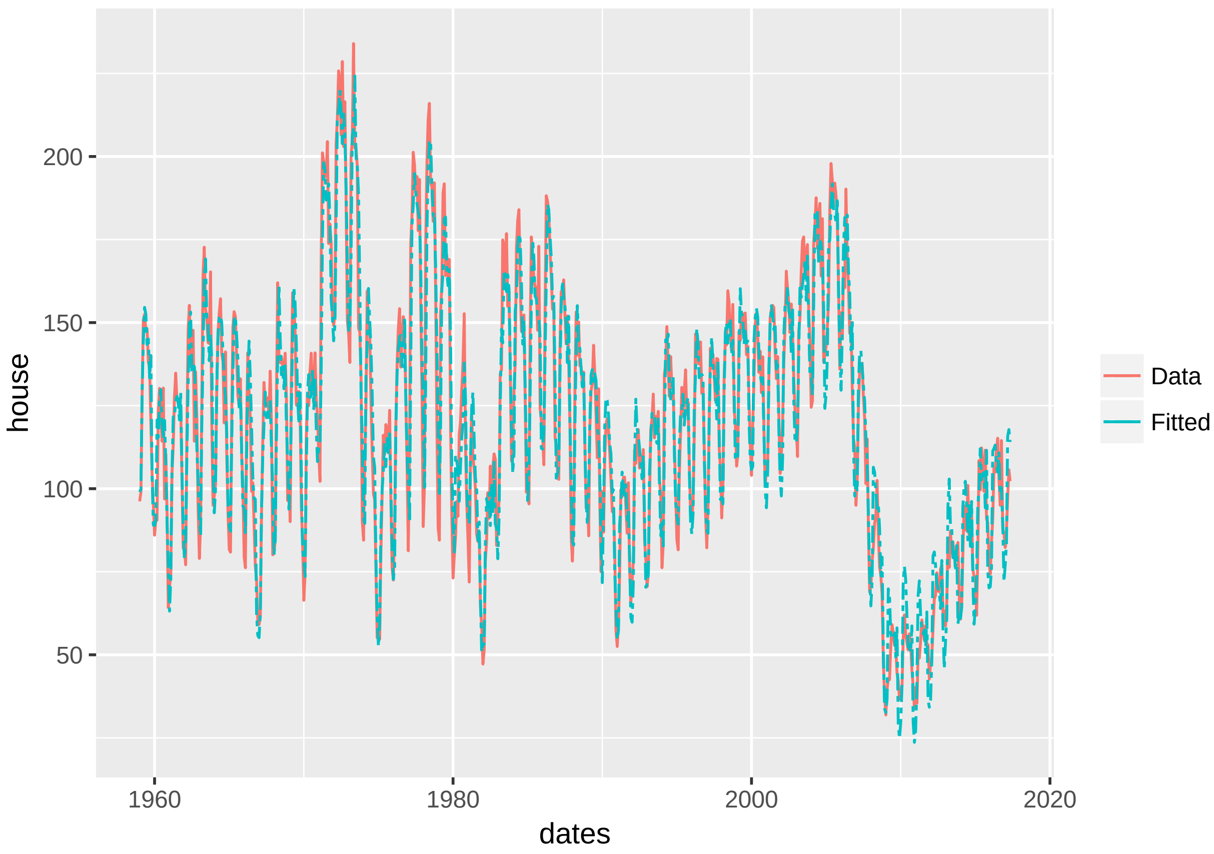
$$y_t = c + \alpha t + \gamma_t + \varepsilon_t$$

$$\varepsilon_t = \phi_1 \varepsilon_{t-1} + \eta_t + \theta_1 \eta_{t-1}$$

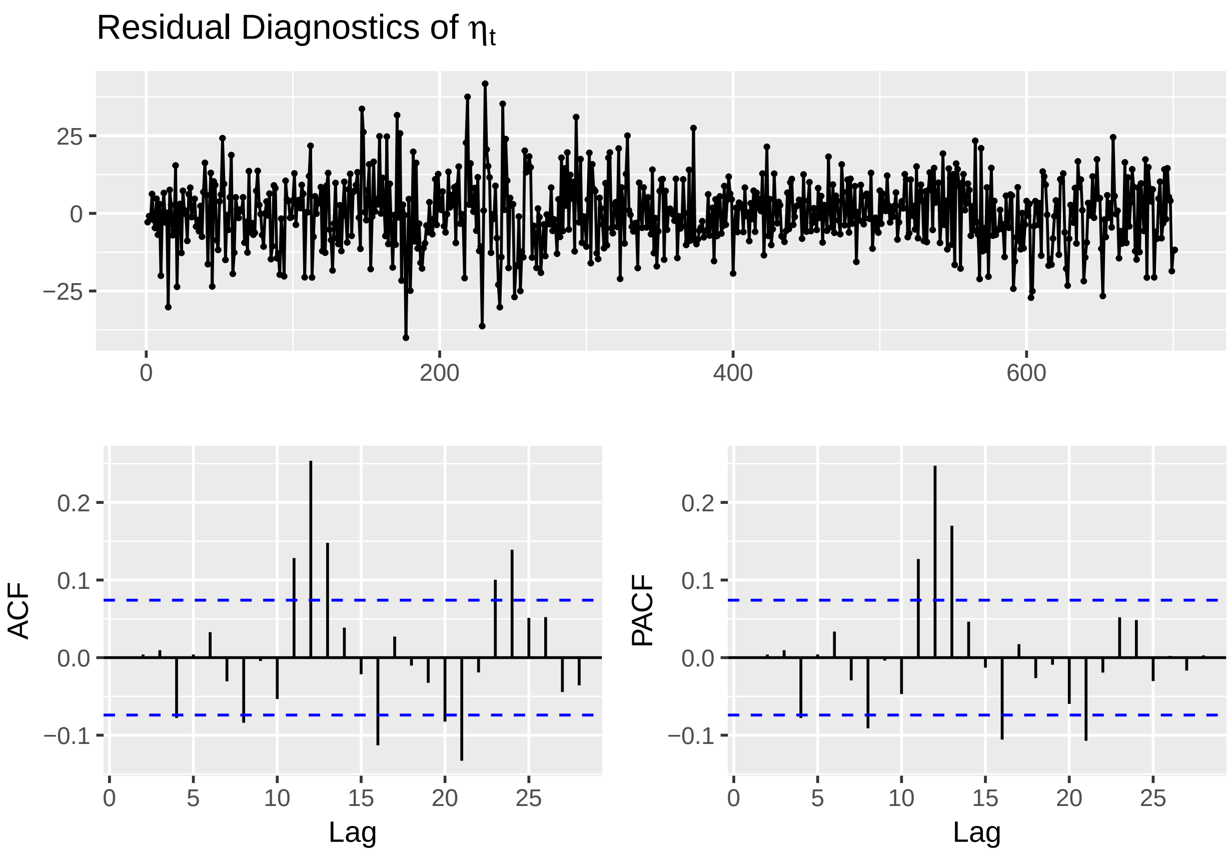
$$\eta_t \sim WN(0, \sigma^2)$$

In order to obtain the fitted values of y_t we need the fitted values of ε_t and add the constant, trend and seasonal component obtained in the OLS model (i.e. estimates for c, αt and γ_t). We first create a vector for the first part of the model and then add it to the fitted values of ε_t to obtain the fitted values of y_t. We observe that our estimate fits the data well. Looking at the residuals we do not observe (very) significant autocorrelation with the exception of lags 11 to 13 and 16 as well as 21. Thus it is unsurprising that the residuals look a lot like white noise.

```
arma11 <- arima(resid(mod1), c(1, 0, 1))
plotdat <- data.frame(house, residuals = resid(arma11))
plotdat$dates <- seq(as.Date("1959-01-01"), by = "month", length.out = length(house))

fstep <- mod1$coefficients[1] + seasons %*% as.matrix(mod1$coefficients[2:12], ncol = 1) + trend * mod1$
actualfitted <- fitted(arma11)+ fstep

library(ggplot2)
library(latex2exp)
ggplot(plotdat, aes(x = dates)) +
  geom_line(aes(y = house, color = "Data"))+
  geom_line(aes(y = actualfitted, color = "Fitted"), linetype = "twodash")+
  theme(legend.title=element_blank())
```




```
ggtsdisplay(plotdat$residuals, main = TeX("Residual Diagnostics of $\\eta_t$"))
```

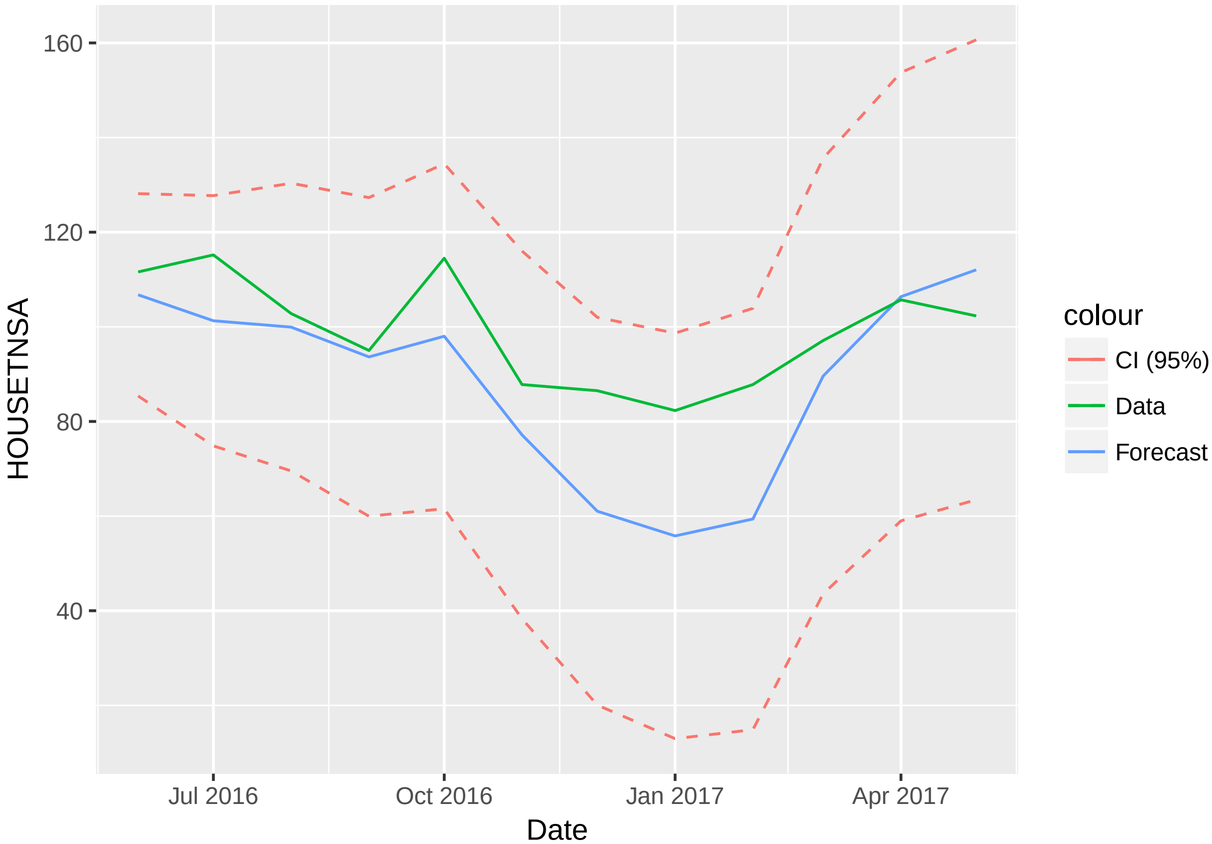


(d) Forecasting

For the pseudo-out-of-sample forecast we restrict our data set to values up to May 2016 and repeat the estimations in (a) and (c) using the ARMA(1,1) in the estimation of ε_t. Some care needs to be taken when forecasting the trend component. We take the last 12 values in the trend vector and add 12 element-wise to get the "future" trend. The seasonal component of course stays the same accross years.

```
rhouse <- window(house, end = c(2016,5))
rseasons <- seasonaldummy(rhouse)
rtrend <- 1:length(rhouse)
rmod <- lm(rhouse ~ rseasons + rtrend)
rarma11 <- Arima(resid(rmod), order = c(1, 0, 1))
armafcast <- forecast(rarma11, 12)

coefs <- rmod$coefficients
trendadd <- (tail(1:length(rhouse), 12) + 12) * coefs[13]
seasonsadd <- tail(rseasons,12) %*% as.matrix(coefs[2:12], ncol = 1)
firststep <- coefs[1] + seasonsadd + trendadd

fcastplot <- data.frame(armafcast, dat = tail(house, 12), dates = tail(plotdat$dates, 12))
fcastplot[1:5] <- fcastplot[1:5] + firststep

ggplot(fcastplot, aes(x = dates))+
  geom_line(aes(y = Point.Forecast, color = "Forecast"))+
  geom_line(aes(y = Lo.95, color = "CI (95%)"), linetype = "dashed")+
  geom_line(aes(y = Hi.95, color = "CI (95%)"), linetype = "dashed")+
```

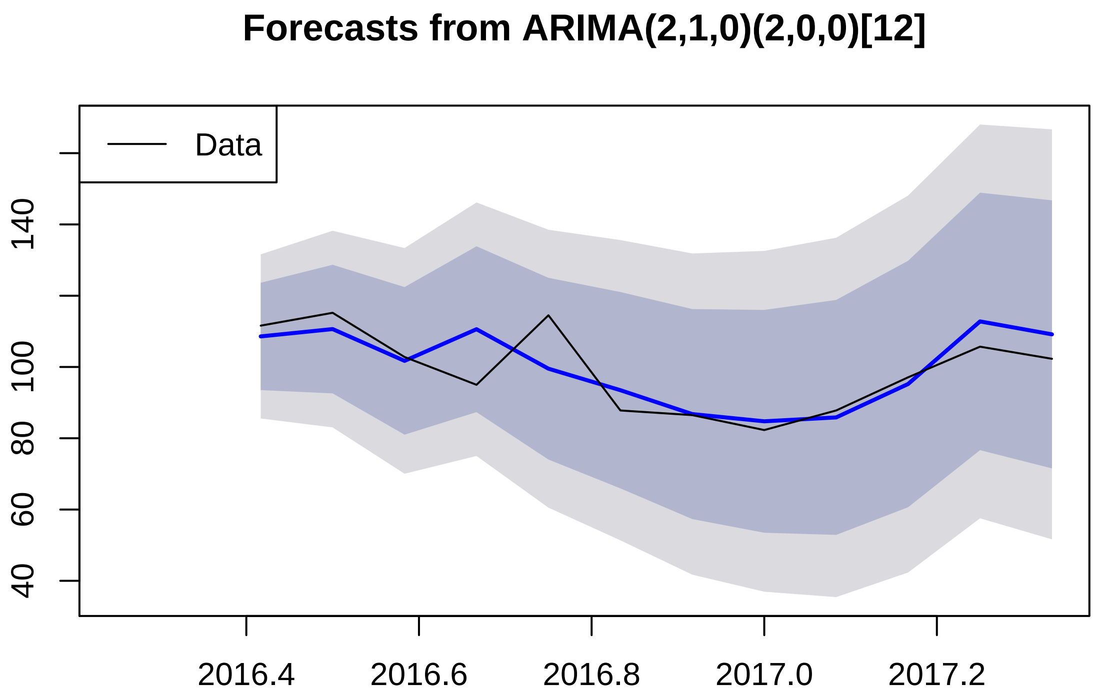
```
  geom_line(aes(y = dat, color = "Data"))+
  labs(y = "HOUSETNSA", x = "Date")
```



(extra) Automated ARIMA

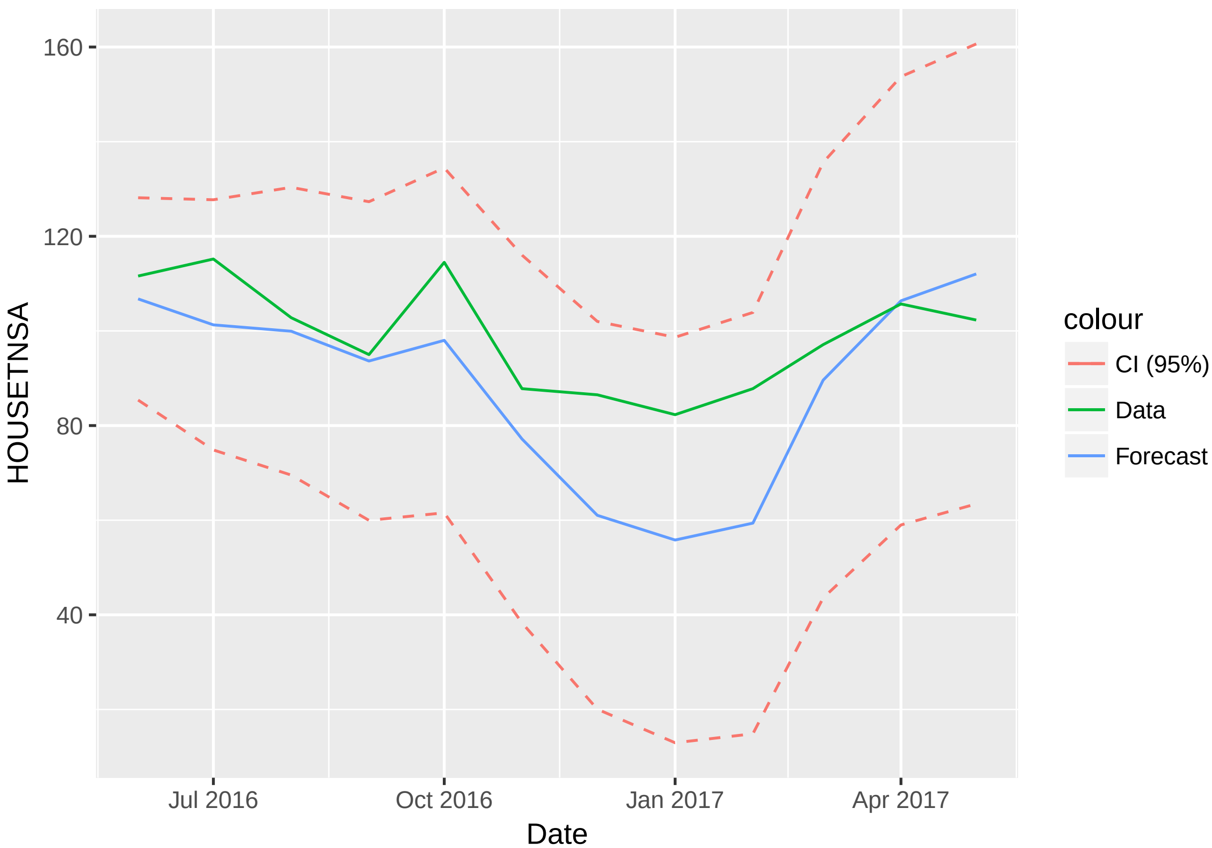
We repeate the exercise using the automated arima command in R. It finds a better fit for the data to be an ARIMA(2,1,0).

```
endmod <- auto.arima(rhouse)
plot(forecast(endmod, 12), include = 0)
lines(fcastplot$dat)
legend(x = "topleft", legend = c("Data"),col = "black", lty = 1)
```



However, an AR(2) on the ε_t, which was our first intuition, does not yield a visibly better result.

```
rarma20 <- Arima(resid(rmod), order = c(2, 0, 0))
armafcast <- forecast(rarma11, 12)

coefs <- rmod$coefficients
trendadd <- (tail(1:length(rhouse), 12) + 12) * coefs[13]
seasonsadd <- tail(rseasons,12) %*% as.matrix(coefs[2:12], ncol = 1)
firststep <- coefs[1] + seasonsadd + trendadd

fcastplot <- data.frame(armafcast, dat = tail(house, 12), dates = tail(plotdat$dates, 12))
fcastplot[1:5] <- fcastplot[1:5] + firststep

ggplot(fcastplot, aes(x = dates))+
  geom_line(aes(y = Point.Forecast, color = "Forecast"))+
  geom_line(aes(y = Lo.95, color = "CI (95%)"), linetype = "dashed")+
  geom_line(aes(y = Hi.95, color = "CI (95%)"), linetype = "dashed")+
  geom_line(aes(y = dat, color = "Data"))+
  labs(y = "HOUSETNSA", x = "Date")
```


160
120
80
40
HOUSETNSA
Jul 2016
Oct 2016
Jan 2017
Apr 2017
Date
colour
CI (95%)
Data
Forecast