TMA4285 Time series models

Exercise 8: Comparison of a state space model and a SARIMA model

Sivert Selnes, Kristine L. Mathisen, Gina Magnussen

20th of October 2018

```r
# Libraries
library(astsa)
library(forecast)
library(ggplot2)
library(dlm)
library(KFAS)
library(zoo)
library(boot)
# libraries for state-space: dse, dlm, KFAS, stsm
```

Remember:

- Note how uncertainty should be written: 0.056(7) (See sources on web page)

Abstract

Introduction

Theory

* State-space models

General:

$$\begin{aligned} Y_t &= G_t\mathbf{X}_t + W_t & \{\mathbf{W}_t\} &\sim \text{WN}\left(\mathbf{0}, \{R_t\}\right) \\ \mathbf{X}_{t+1} &= F_t\mathbf{X}_t + \mathbf{V}_t & \{\mathbf{V}_t\} &\sim \text{WN}\left(\mathbf{0}, \{Q_t\}\right) \end{aligned}$$

When F, G, R and Q are time independent:

$$\begin{aligned} Y_t &= G\mathbf{X}_t + W_t & \{W_t\} &\sim \text{WN}\left(\mathbf{0}, R\right) \\ \mathbf{X}_{t+1} &= F\mathbf{X}_t + \mathbf{V}_t & \{\mathbf{V}_t\} &\sim \text{WN}\left(\mathbf{0}, Q\right) \end{aligned}$$

* SARIMA

A SARIMA model is an extended version of the ARIMA model which can also consider a seasonal component in a time series. Its general form

$$SARIMA(p,d,q) \times (P,D,Q)_s$$

is a seasonal ARIMA process with period s and is written as

$$\phi(B)\Phi\left(B^S\right)Y_t = \theta(B)\Theta\left(B^S\right)Z_t, \qquad \{Z_t\} \sim \mathrm{WN}(0,\sigma^2)$$

if the differenced series $Y_t = (1-B)^d(1-B^s)^D X_t$ is a causal ARMA process defined as above. Here, $\phi(z) = 1-\phi_1 z-\ldots-\phi_p z^p$, $\Phi(z) = 1-\Phi_1 z-\ldots-\Phi_P z^P$, $\theta(z) = 1+\theta_1 z+\ldots+\theta_q z^q$ and $\Theta(z) = 1+\Theta_1 z+\ldots+\Theta_Q z^Q$.

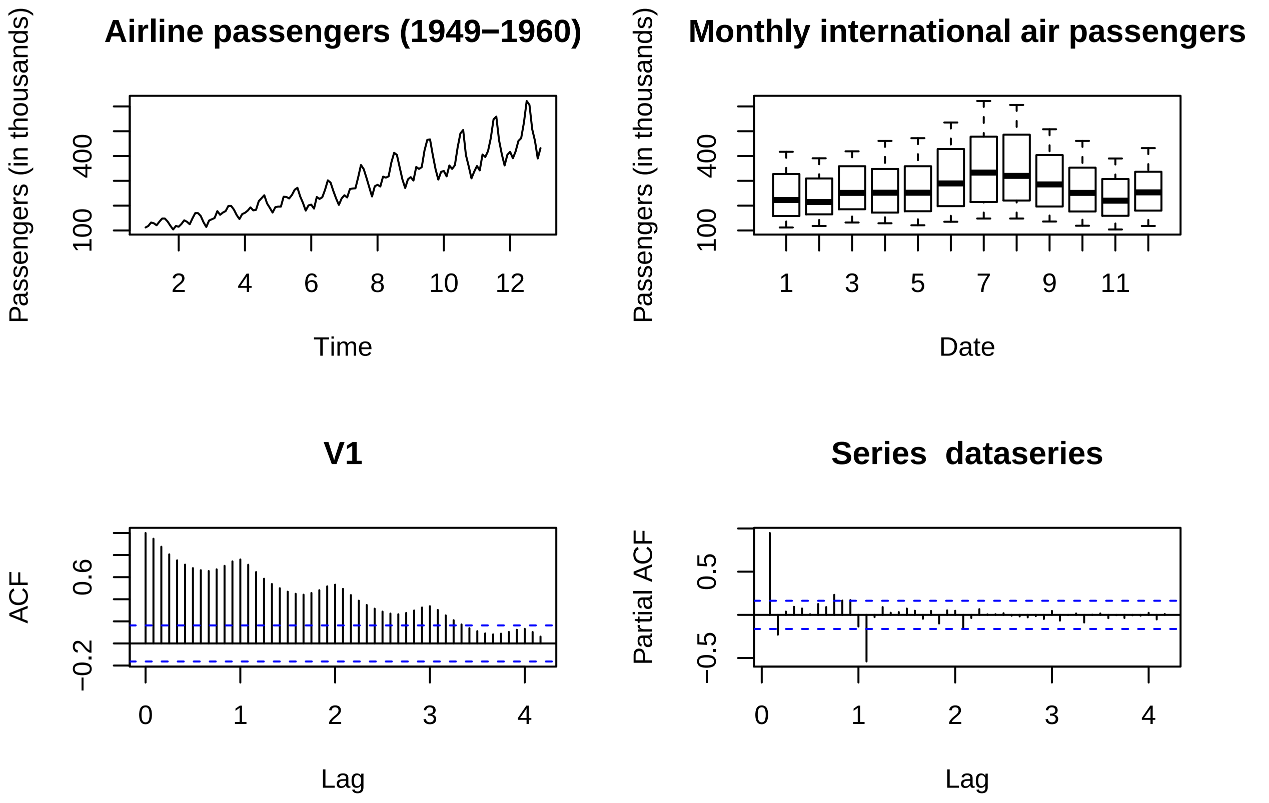
The SARIMA model has two parts: A seasonal component and a non-seasonal component. (P,D,Q) is the ARIMA model between each season and is the seasonal component of the SARIMA model. This part models behaviour for for instance the January months in a time series with monthly observations and where the season is a year long. (p,d,q) is the non-seasonal part of the model and is an ARIMA model *within* each season. The interpretation of p,d,q,P,d and Q are known from previous knowledge on the ARIMA models, and B is the backshift operator as before.

Data analysis

SARIMA$(p,d,q)\times(P,D,Q)_s$ model

The data set consists of total number of international airline passengers, in thousands, for each month from January 1949 to December 1960 (reference to data set), giving a total number of $N = 144$ observations.

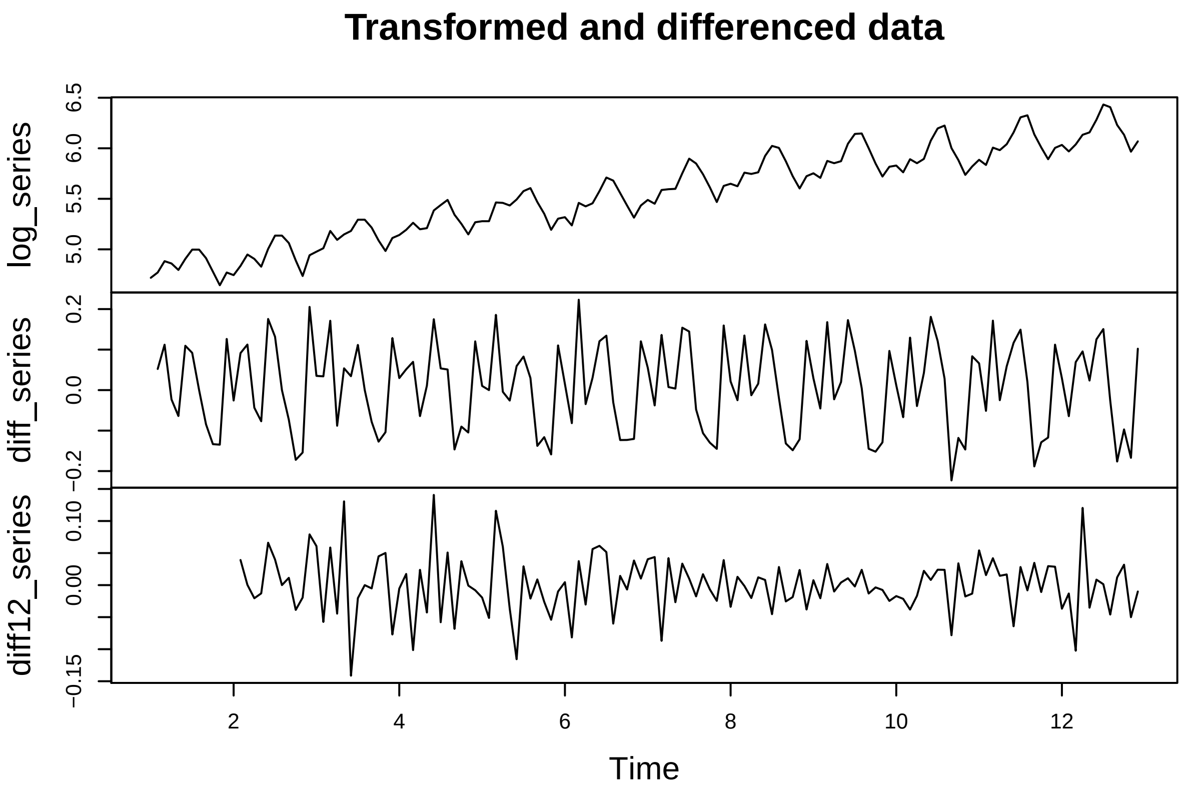
```
dataseries <- ts(read.table("dataEx8.txt"), frequency = 12)
par(mfrow=c(2,2))
plot(dataseries,main = "Airline passengers (1949-1960)", ylab="Passengers (in thousands)")
boxplot(dataseries~cycle(dataseries), xlab="Date", ylab = "Passengers (in thousands)",
        main ="Monthly international air passengers")
acf(dataseries, lag.max = 50)
pacf(dataseries, lag.max = 50)
```



From the time series plot, this is obviously a non-stationary series as both trend and seasonality is visible in the visualization. Trend is there because the total number of airline passengers on average increases for each month, and seasonality because of the wave-like behaviour of the series. Non-stationarity is further supported by looking at the autocorrelation function, which shows considerable correlation between observations even up to lag h of size 30, and again a periodic wave-like pattern. Furthermore, the box plot shows both higher mean number of passengers and also higher variance for months 6 to 9 in the year, i.e. June til September.

To investigate the data, we first find a transformation to the time series. A typical transformation for making the series more stationary is the log()-transformation. This seems to stabilize the multiplicative behaviour of the variance so that the variance is not increasing with time.

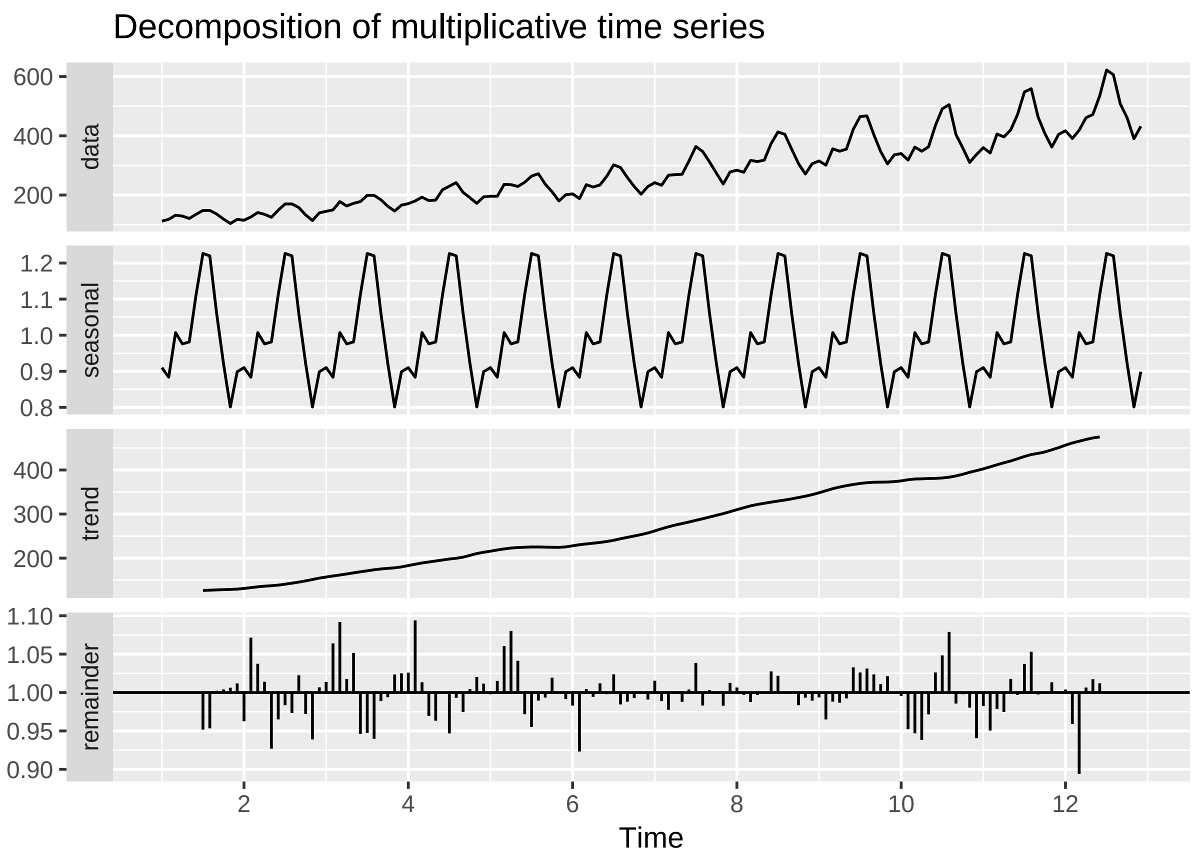
```r
# Log-transform the data
log_series <- log(dataseries)
# Differencing the data
diff_series <- diff(log_series)
diff12_series <- diff(diff_series, lag = 12)

# Plotting the transformed data series
plot.ts(cbind(log_series, diff_series, diff12_series), main="Transformed and differenced data")
```



Secondly, the data is differenced to remove trend, i.e. stabilizing the mean. The parameter d is related to trend within a season, and since this trend seems to be removed by differencing once, we try $d = 1$. See the plot in the middle of the figure for transformed and differenced data. After differencing there still is a wave pattern present, one for each year. This indicates a seasonal trend equal to the length of a year, i.e. we suspect a $s = 12$ in a seasonal ARIMA model. Then $(1 - B^{12})$ is applied to the series. The parameter D is related to the trend between seasons, i.e. the trend one can se from one time of the year to the next, and the next, and so on. We therefore set $D = 1$. Differencing once indicates linear trend, both within and between the seasons.

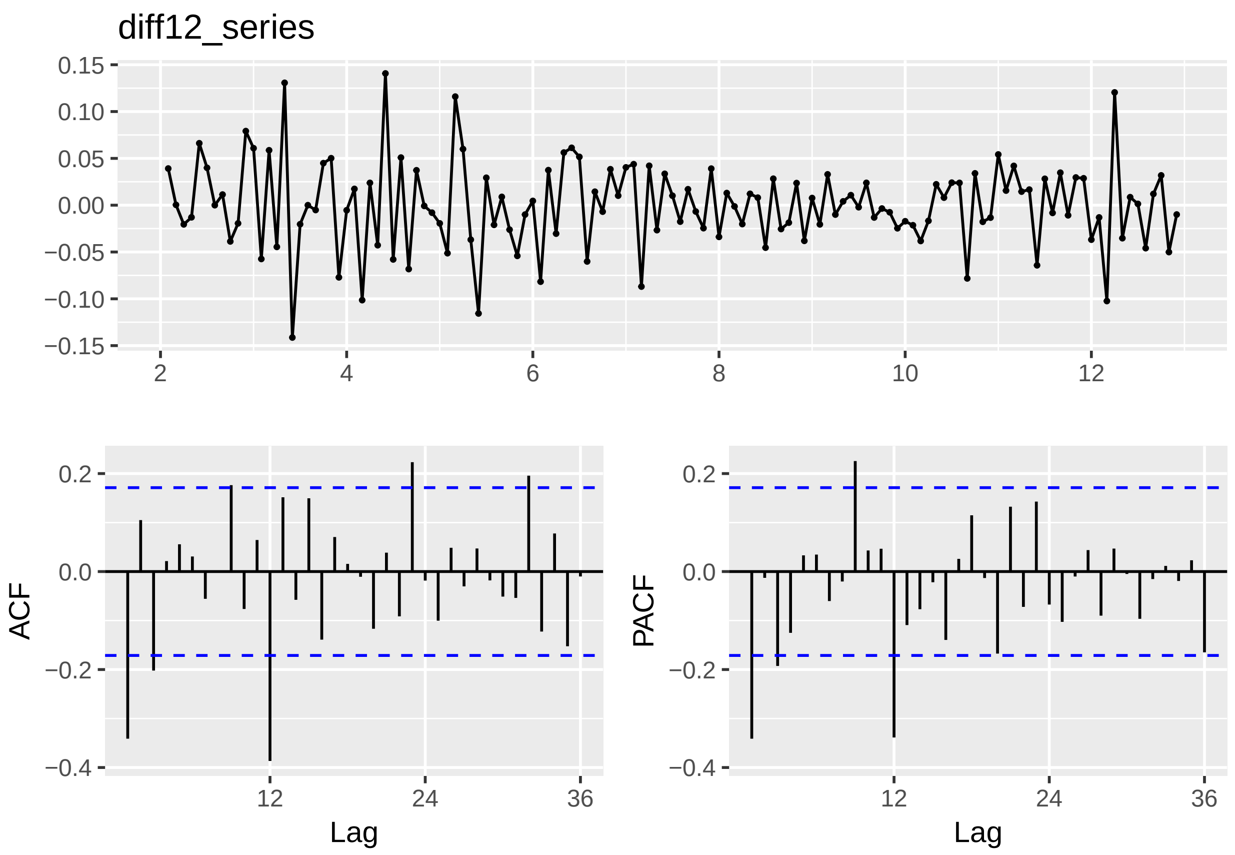
Our model parameter choices can be checked by comparing to the decomposed time series:

```r
autoplot(decompose(dataseries, "multiplicative"))
```



which clearly shows linear trend and a season equal to a year.

```r
ggtsdisplay(diff12_series)
```



The obtained series after differencing the second time seems stationary without any trend or seasonality. We can then estimate the remaining parameters by considering the ACF and the PACF. This could be done by looking at lags equal to $1s, 2s, ...,$ $s = 12$ to determine P and Q in the seasonal component and by looking at smaller lags in each season to determine p and q in the non-seasonal component. In this case, however, we have used the AICC criterion and tested models with $d = 1$, $D = 1$, $s = 12$, which is determined before, and $p, q, P, Q < 5$ which is based by looking at the ACF and PACF. In addition we prefer models with smaller parameters for simplicity if they are a good fit.

The `auto.arima()`-function tests a variety of models and chooses the best one given our chosen criteria:

```
bestfit <- auto.arima(log_series, d=1, D=1, max.p=5,max.q=5, max.P=5, max.Q=5, seasonal=TRUE, ic="aicc"
```

```
##
##  ARIMA(2,1,2)(1,1,1)[12]                    : Inf
##  ARIMA(0,1,0)(0,1,0)[12]                    : -434.799
##  ARIMA(1,1,0)(1,1,0)[12]                    : -474.6299
##  ARIMA(0,1,1)(0,1,1)[12]                    : -483.2101
##  ARIMA(0,1,1)(1,1,1)[12]                    : -481.5957
##  ARIMA(0,1,1)(0,1,0)[12]                    : -449.8857
##  ARIMA(0,1,1)(0,1,2)[12]                    : -481.6451
##  ARIMA(0,1,1)(1,1,2)[12]                    : Inf
##  ARIMA(1,1,1)(0,1,1)[12]                    : -481.582
##  ARIMA(0,1,0)(0,1,1)[12]                    : -467.4644
##  ARIMA(0,1,2)(0,1,1)[12]                    : -481.2991
##  ARIMA(1,1,2)(0,1,1)[12]                    : -481.5633
##
##  Best model: ARIMA(0,1,1)(0,1,1)[12]
```

Both parameters are significant in the model, and the residual analysis plot shows that our model is a good fit. In addition, the uncertainty of the parameters are given from `fit$ttable`. Our chosen model is thus

$$\text{SARIMA}(0,1,1)\times(0,1,1)_{12}$$

which is equal to

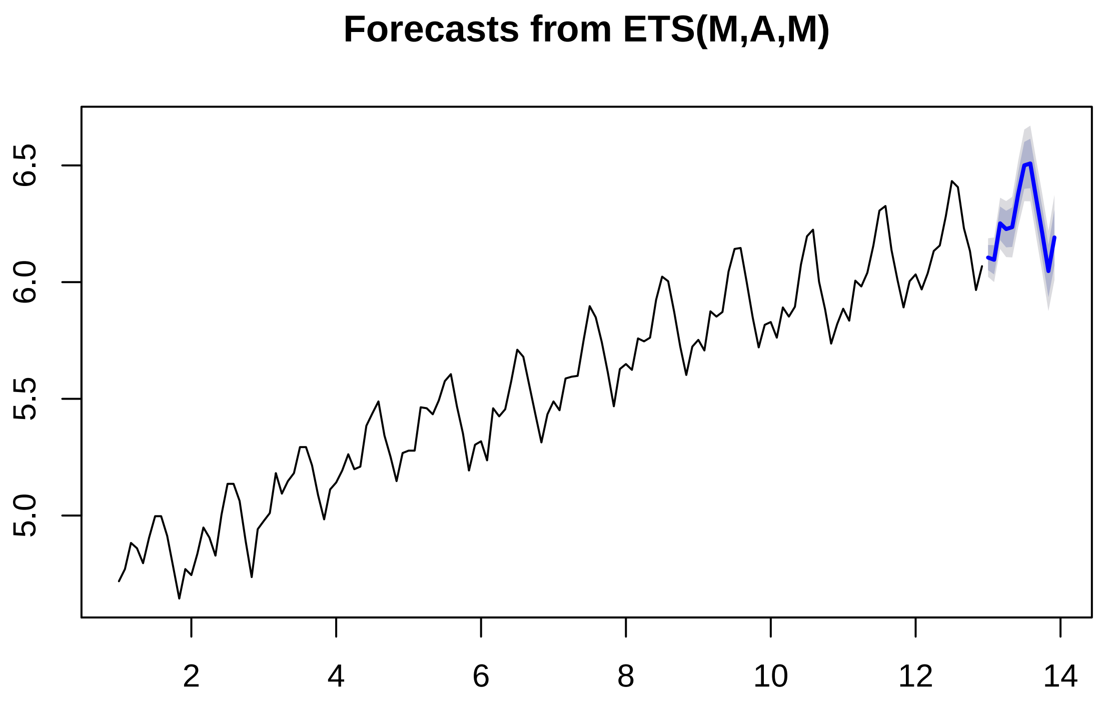
$$\phi(B)\Phi(B^{12})(1-B)(1-B^{12})X_t = \theta(B)\Theta(B^{12})Z_t$$

When we have found our model, we can do forecasting with the SARIMA model for the next twelve months. From the fitting functions in R, we can also extract credible intervals of the forecasts and the fitted values at our given sample points.

```
# Forecasting
future <- 12 # Set no. of months to forecast
# Forecasting
forecast_sarima <- forecast(log_series, h = future) # Forecastin
log_fitted <- forecast_sarima$fitted # Fitted values for already given sample points

# Check model for given sample points
sarima_frame <- data.frame(dataseries, exp(log_fitted), log_series, log_fitted,  1:144)
colnames(sarima_frame) <- c("original", "fitted", "log_original", "log_fitted", "month")
```

The following plot shows the time series and the forecased values for the next twelve months:

```
plot(forecast_sarima)
```

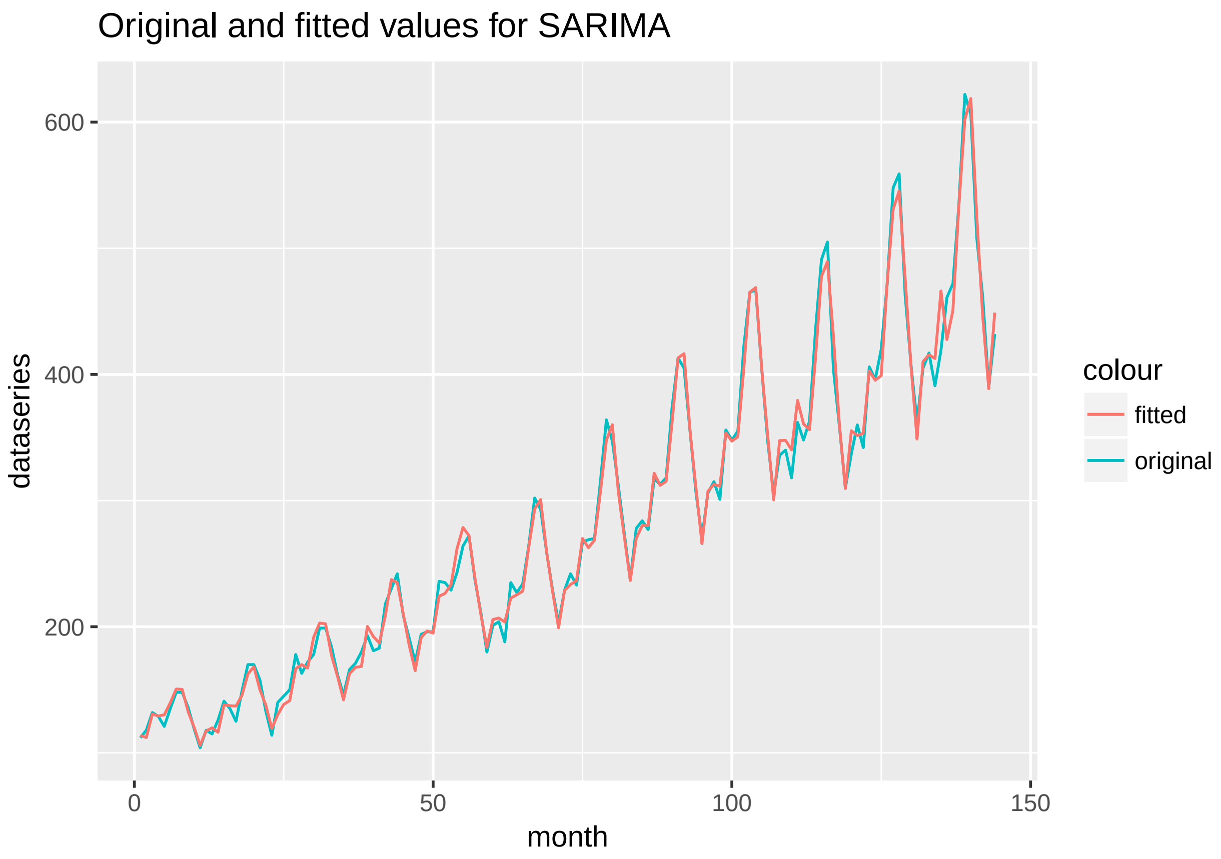


The credible intervals for the forecasts are

```
uncertainty <- data.frame("95percent_lower" = exp(forecast_sarima$lower)[,2], "95percent_upper"=exp(fore
uncertainty
```

```
##    X95percent_lower X95percent_upper
## 1          412.6662         486.7393
## 2          403.4885         488.4474
## 3          465.0367         579.2250
## 4          449.1705         570.9354
## 5          448.4353         581.3636
## 6          511.4191         679.4079
## 7          570.6165         775.6796
## 8          570.3655         788.6390
## 9          487.0434         678.6601
## 10         418.5346         587.0612
## 11         356.2822         502.1102
## 12         407.1544         586.1755
```

```r
ggplot(sarima_frame) +
  geom_line(aes(month, dataseries, color = "original")) +
  geom_line(aes(month,fitted , color="fitted")) +
  labs(title="Original and fitted values for SARIMA", xlab="Months", ylab="Passengers")
```



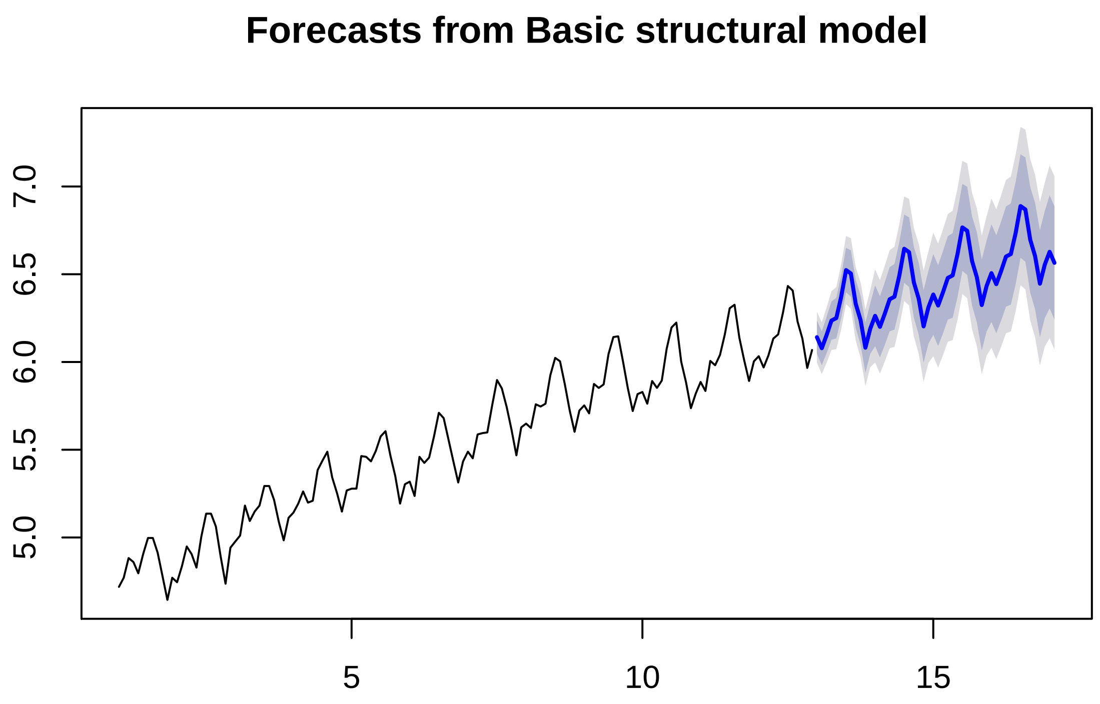
The fitted data looks good compared to the original data and the forecasting seems to continue the season and trend of the data well.

State-space model

We build a state-space model by ??eqref to theory or just use Rfunctions??, which gives the following matrices

```r
# ap <- log10(AirPassengers) - 2
state_space <- StructTS(dataseries, type = "BSM") # Basic structural model
state_space_log <- StructTS(log_series, type ="BSM")
# state_space$model$T # F in state equation
forecasts <- forecast(state_space_log, h = 50)
```

```
#forecasts$lower
#forecasts$upper
plot(forecasts)
```



```
#state_space$model$T # gives the F matrix
#state_space$model$Z # gives the G matrix
```

$F =$

```
print(state_space$model$T)
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
##  [1,]    1    1    0    0    0    0    0    0    0     0     0     0     0
##  [2,]    0    1    0    0    0    0    0    0    0     0     0     0     0
##  [3,]    0    0   -1   -1   -1   -1   -1   -1   -1    -1    -1    -1    -1
##  [4,]    0    0    1    0    0    0    0    0    0     0     0     0     0
##  [5,]    0    0    0    1    0    0    0    0    0     0     0     0     0
##  [6,]    0    0    0    0    1    0    0    0    0     0     0     0     0
##  [7,]    0    0    0    0    0    1    0    0    0     0     0     0     0
##  [8,]    0    0    0    0    0    0    1    0    0     0     0     0     0
##  [9,]    0    0    0    0    0    0    0    1    0     0     0     0     0
## [10,]    0    0    0    0    0    0    0    0    1     0     0     0     0
## [11,]    0    0    0    0    0    0    0    0    0     1     0     0     0
## [12,]    0    0    0    0    0    0    0    0    0     0     1     0     0
## [13,]    0    0    0    0    0    0    0    0    0     0     0     1     0
```

$G =$

```
print(state_space$model$Z)
```

```
##  [1] 1 0 1 0 0 0 0 0 0 0 0 0 0
```

$Q =$

```r
print(state_space$model$V)
```

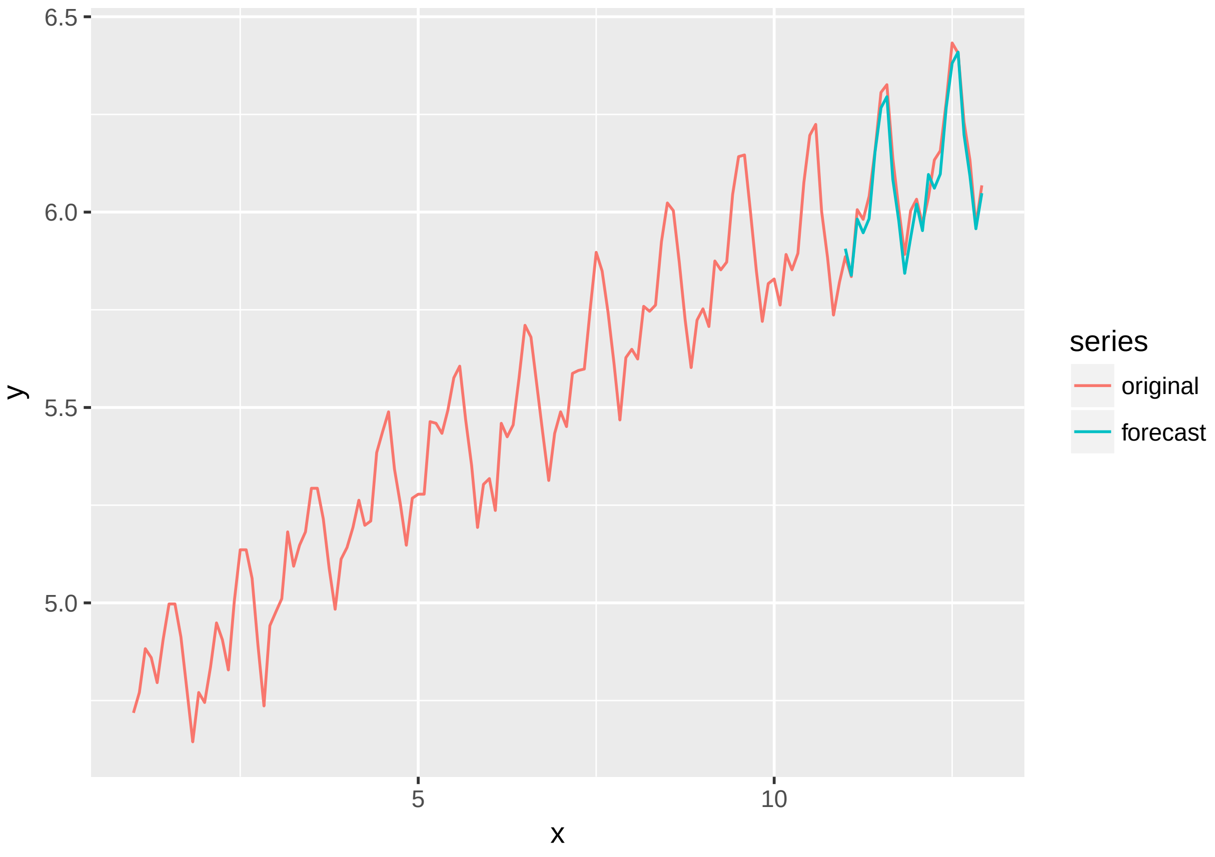
```
##       [,1]     [,2]     [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11]
##  [1,]    0   0.0000  0.00000    0    0    0    0    0    0     0     0
##  [2,]    0 160.9755  0.00000    0    0    0    0    0    0     0     0
##  [3,]    0   0.0000 29.84652    0    0    0    0    0    0     0     0
##  [4,]    0   0.0000  0.00000    0    0    0    0    0    0     0     0
##  [5,]    0   0.0000  0.00000    0    0    0    0    0    0     0     0
##  [6,]    0   0.0000  0.00000    0    0    0    0    0    0     0     0
##  [7,]    0   0.0000  0.00000    0    0    0    0    0    0     0     0
##  [8,]    0   0.0000  0.00000    0    0    0    0    0    0     0     0
##  [9,]    0   0.0000  0.00000    0    0    0    0    0    0     0     0
## [10,]    0   0.0000  0.00000    0    0    0    0    0    0     0     0
## [11,]    0   0.0000  0.00000    0    0    0    0    0    0     0     0
## [12,]    0   0.0000  0.00000    0    0    0    0    0    0     0     0
## [13,]    0   0.0000  0.00000    0    0    0    0    0    0     0     0
##       [,12] [,13]
##  [1,]     0     0
##  [2,]     0     0
##  [3,]     0     0
##  [4,]     0     0
##  [5,]     0     0
##  [6,]     0     0
##  [7,]     0     0
##  [8,]     0     0
##  [9,]     0     0
## [10,]     0     0
## [11,]     0     0
## [12,]     0     0
## [13,]     0     0
```

This gives $(\hat{\sigma}_1^2, \hat{\sigma}_2^2, \hat{\sigma}_2^3) = (0.0000, 160.9755, 29.84652)$.

```r
print(state_space$model$a)
```

```
##  [1] 417.8507614 -29.6143515  14.1492386 -57.4651129 -23.6949346
##  [6] -12.2523959  56.6479592  60.1013022  -9.8206840 -37.0447315
## [11]  -4.5661018  -0.8086603  -7.7654562
```

```r
# State space with DLM method

model.build <- function(p) {
  return(
    dlmModPoly(2, dV=p[1], dW=p[2:3]) +
      dlmModSeas(12, dV=p[4])
  )
}

#log.air <- log(air) + rnorm(length(log.air), 0, 0.15)
log.air <- log_series
train <- log.air[1:120]
test <- log.air[121:144]

model.mle <- dlmMLE(train, parm=c(1, 1, 1, 1), build=model.build)
model.fit <- model.build(model.mle$par)
model.filtered <- dlmFilter(train, model.fit)
model.smoothed <- dlmSmooth(train, model.fit)

n <- 2*12
model.forecast <- dlmForecast(model.filtered, nAhead=n)

x <- index(log.air)
a <- drop(model.forecast$a%*%t(FF(model.fit)))
df <- rbind(
  data.frame(x=index(log.air), y=as.numeric(log.air), series="original"),
  #data.frame(x=x[1:120], y=apply(model.filtered$m[-1,1:2], 1, sum), series="filtered"),
  #data.frame(x=x[1:120], y=apply(model.smoothed$s[-1,1:2], 1, sum), series="smoothed"),
  data.frame(x=x[121:144], y=a, series="forecast")
)
g.dlm <- ggplot(df, aes(x=x, y=y, colour=series)) + geom_line()
g.dlm
```



Bootstrap

We bootstrap the estimated parameters for the state space model.

- Model diagnostics for state-space \
- Model comparison + discussion. \

Discussion

Conclusion

Appendix

## References