Advanced Time Series Analysis: Computer Exercise 2

Anders Launer Bæk (s160159)

05 November 2017

Sparring partners:

- Anja Liljedahl Christensen (s162876)
- Marie Mørk (s112770)

Part 1

There has been simulated $n = 3000$ samples of noise where $\epsilon_t \sim \mathcal{N}(0, 1)$. ϵ_t is used as noise input for simulations in part one.

The equation below shows the used parameters in the selected system: a SETAR(2,1,1) model. The parameters for the two regimes are defined by eqn. 1 and eqn. 2.

$$a_0 = [0.125, -0.125] \tag{1}$$

$$a_1 = [0.6, -0.4] \tag{2}$$

Simulation of the SETAR(2,1,1)

The SETAR(2,1,1) model is given by eqn. 3.

$$X_t = a_0^{(J_t)} + \sum_{i=1}^{k_{(J_t)}} a_i^{(J_t)} X_{t-i} + \epsilon^{(J_t)} \tag{3}$$

where J_t are the regime processes. The complete model are defined in eqn. 4.

$$X_t = \begin{Bmatrix} a_{0,1} + a_{1,1} X_{t-1} + \epsilon_t & for & X_{t-1} \leq 0 \\ a_{0,2} + a_{1,2} X_{t-1} + \epsilon_t & for & X_{t-1} > 0 \end{Bmatrix} \tag{4}$$

A simulation of the model (eqn. 4) is showed in fig. 1.

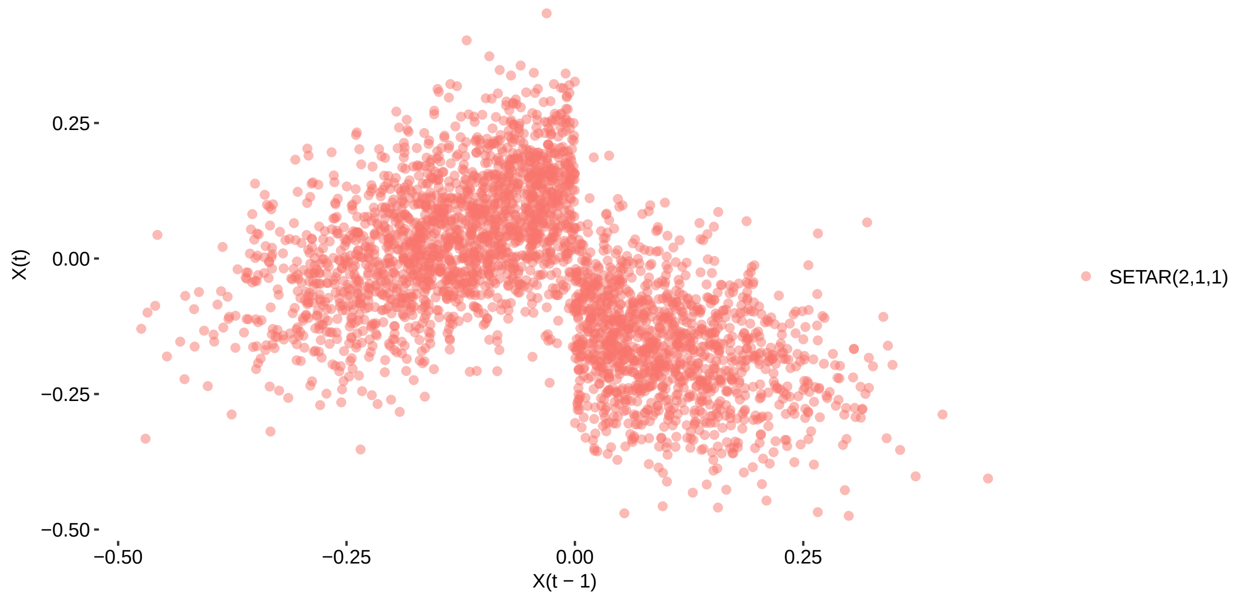

Figure 1: One simulation of a SETAR(2,1,1) model.

Estimate the parameters using conditional least squares

The following code snippet contain three functions: `Setar()`, `RSSSetar` and `PESetar` calculates the conditional means, the squared residuals and the total squared prediction error respectively.

```r
# calculation of the conditional mean for the SETAR(2,1,1) model
Setar <- function(par, model) {
    # initialize conditional mean vector
    e_mean <- rep(NA, length(model))
    # loop through observations
    for (t in 2:length(model)) {
        if (model[t - 1] <= 0) {
            e_mean[t] <- par[1] + par[2] * model[t - 1]
        } else {
            e_mean[t] <- par[3] + par[4] * model[t - 1]
        }
    }
    # return the conditional mean vector
    return(e_mean)
}

RSSSetar <- function(par, model) {
    # conditional mean
    e_mean <- Setar(par, model)
    # calculate and return the squared residuals
    return((model - e_mean)^2)
}

# summed prediction error
PESetar <- function(par, model) {
    # conditional mean
```

Table 1: Table shows the real parameters, the estimated parameters and their percentage diviation.

Parameter	Real value	Optimized value	Change in (%)
a0_1	0.125	0.1267946	1.4153889
a1_1	0.600	0.6127632	2.0828869
a0_2	-0.125	-0.1239285	-0.8645787
a1_2	-0.400	-0.4062514	1.5388065

```
    e_mean <- Setar(par, model)
    # calculate and return the objective function value
    return(sum((model - e_mean)^2, na.rm = TRUE))
}
```

The native R function `optim()` has been applied to estimate the parameters of the simulated process. The objective cost function is the total squared prediction error, which needs to be minimized.

eqn. 5 and eqn. 6 have been used as initial parameter input to optimization function. The "initial" conditional mean is computed with the initial parameters (eqn. 5 and eqn. 6) and illustrated in fig. 2 with the label: `M(x) initial`.

$$a_0 = [0.1, -0.02] \tag{5}$$

$$a_1 = [0.4, -0.25] \tag{6}$$

The estimated parameters are listed in table 1.

The percentage deviation from the real parameters are within $\pm 2.083\%$. The optimized squared prediction error (30.058) is smaller than the real squared error (30.061) which is a subject of overfitting. Dividing data into train and test with proper cross validation techniques can solve the issue of overfitting.

Figure 2 illustrate the conditional means for the SETAR(2,1,1) model with initial parameters (eqn. 5 and eqn. 6), the real and estimated parameters (table 1).

The initial parameters are not a good representation of the simulated SETAR model, as illustrated in fig. 2 with the label `M(x) initial`. But after estimating the parameters s.t. a minimization of the squared prediction error, the estimated parameters is a reasonable representation of the SETAR model.

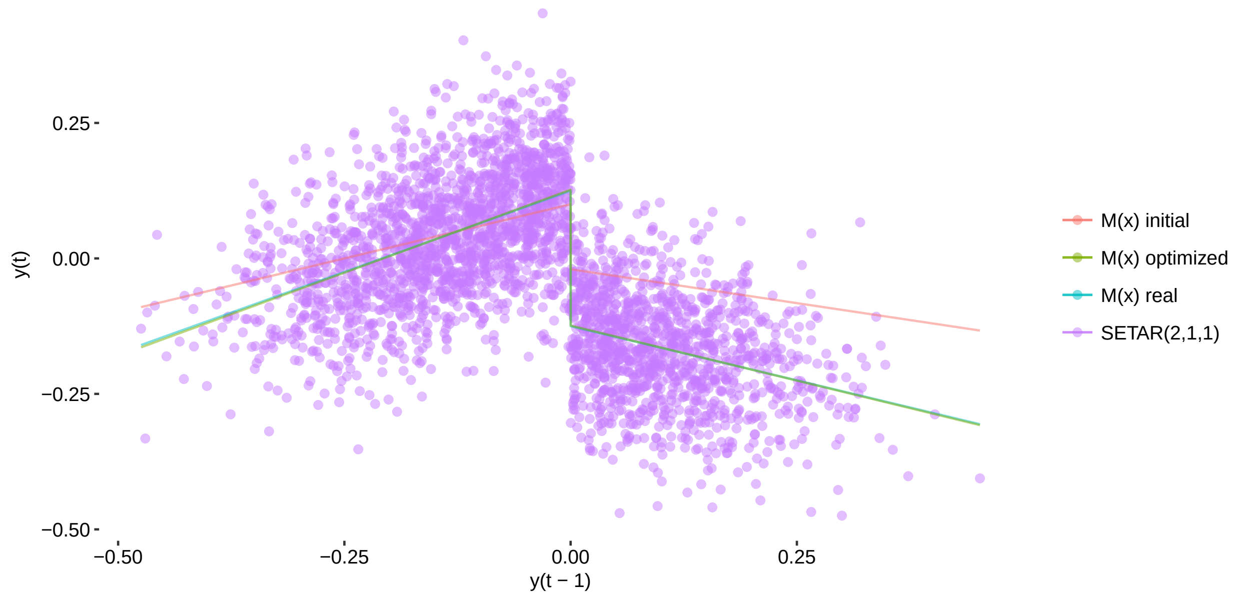

Figure 2: Simulations of a SETAR(2,1,1) model with the initial, real and optimized parameters. The vertical lines in the transition of the regimes does not exist.

Part 2

It has been chosen to create a grid based upon the two slope parameters from eqn. 2. The grid is based on a matrix with the size $(51, 51)$, where the centre element, the intersection of the diagonal and off-diagonal, is the optimal parameter value, and is highlighted with a blue dot. There is also illustrated a red dot, which shows the optimal parameters for the given subset of the simulation. The first axis is $a_{1,1}$ parameter and the second axis in the $a_{1,2}$ parameter.

The the deviation of the parameter values are $\pm 30\%$ (`max_change_p`) from the optimal value. The following function has been used to plot the contours for the different subsets: `plot_contours()`. General to the contour plots, close contour lines identicate a higher slope of the curvature of the objective function.

```r
# only change the slope par[2] and par [4]
plot_contours <- function(model, par = optimal_PE$par, change_p = max_change_p,
    nplot = resolution) {
    # create squence of nplot values with the optimized parameter(s) in center
    par_2_seq <- seq(par[2] - par[2] * change_p, par[2] + par[2] * change_p,
        len = nplot)
    par_4_seq <- seq(par[4] - par[4] * change_p, par[4] + par[4] * change_p,
        len = nplot)

    # create grid
    loess_melt <- expand.grid(par_2_seq, par_4_seq)

    # caculate values
    loess_melt$value <- sapply(1:nrow(loess_melt), function(i) {
        return(PESetar(par = c(par[1], loess_melt$Var1[i], par[3], loess_melt$Var2[i]),
            model = model))
    })

    # find optimal subset value
    min_val <- loess_melt[which.min(loess_melt$value), ]

    # return contour plot
    return(ggplot(loess_melt, aes(x = Var1, y = Var2, z = value)) + stat_contour(geom = "contour",
        alpha = 1/2, binwidth = 0.005) + geom_point(aes(x = par[2], y = par[4],
        color = paste0("optim,\nval=", round(PESetar(par = par, model = model),
            digits = 4))), alpha = 1/2) + geom_point(aes(x = min_val$Var1, y = min_val$Var2,
        color = paste0("best subset,\nval=", round(min_val$value, digits = 4))),
        alpha = 1/2) + labs(x = "a_1,1", y = "a_1,2", color = "parameter and value") +
        theme_TS())
}
```

The contour plots illustrates the objective function (total squared prediction error) with changes in each changeable parameter. The objective function is convex which means there is only one local/global minima.

N = 1:3000

Figure 3 shows the contour plot for the complete simulated model. The optimal parameters gives the global minima of the objective function, and the two dots are identical as illustrated.

The $a_{1,1}$ parameter will result in a higher objective function with equally changes in both parameter.

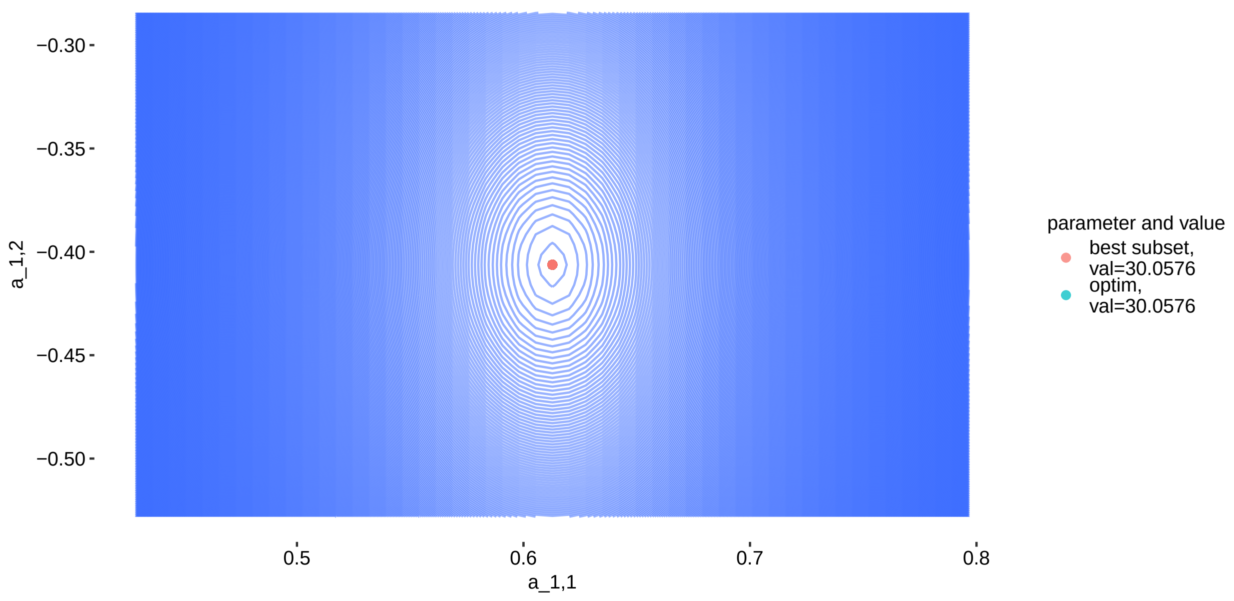

Figure 3: Contour plot the slope parameters for N = 1:3000.

N = 1:300

Figure 4 shows the contour plot with the first 300 samples of the simulated model. The number of samples are decreased with a factor of 100, which is similar to the decrease in the objective function.

The optimal parameters does not represent the minimum value of the of the objective function. There is a clear separation of the two dots and the their values are different, see labels in fig. 4.

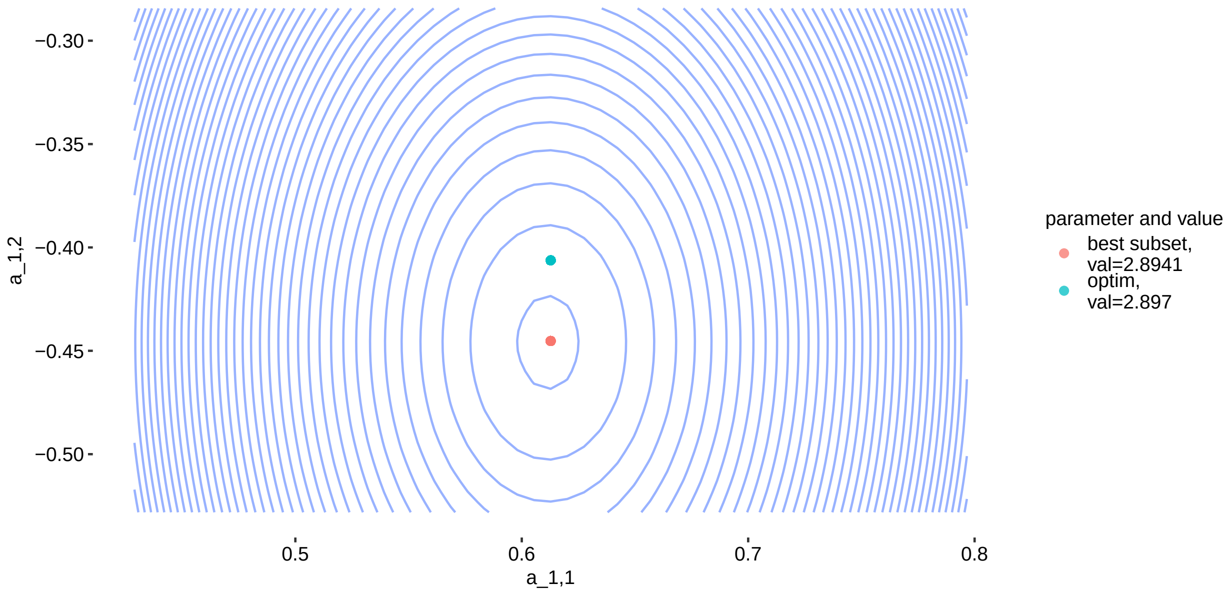

Figure 4: Contour plot the slope parameters for N = 1:300.

N = 1:30

Figure 5 shows a contour plot of the first 30 samples. It is possible to see that the best set of parameter, in the given subset, is not within ±30% of the optimized parameters. The red dot is placed on the boundary of the grid which indicates that the global minimum is not obtained.

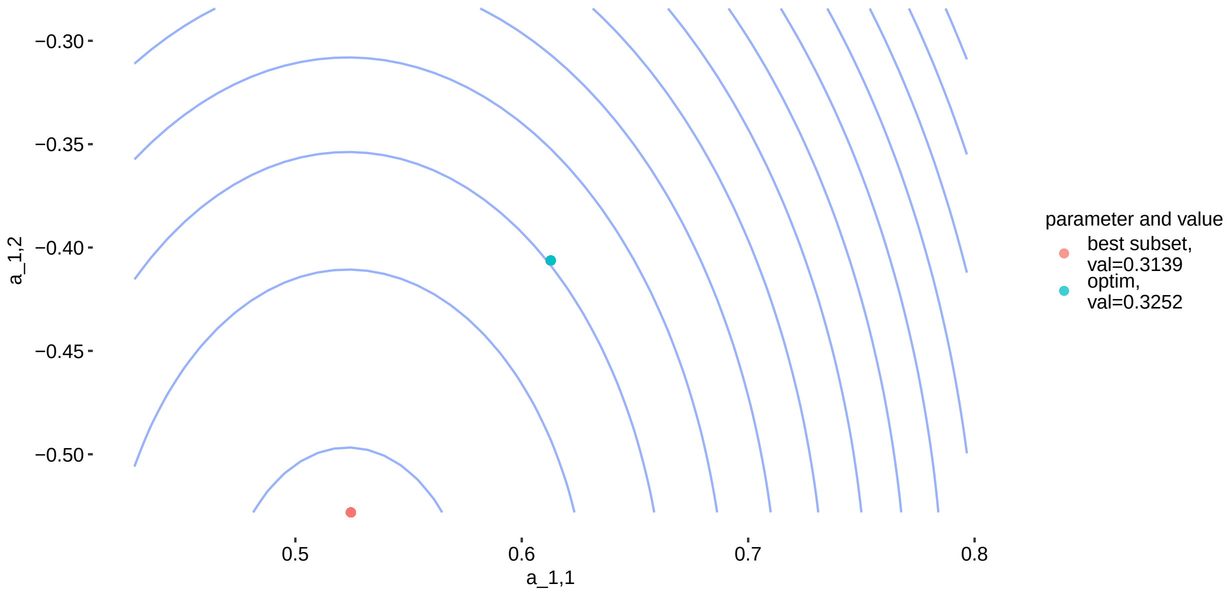

Figure 5: Contour plot the slope parameters for N = 1:30.

N = 1001:1300

Figure 6 uses same number of simulated samples but in an different "location" of the simulated data. The optimal subset parameters is not within ±30% of the optimized parameters.

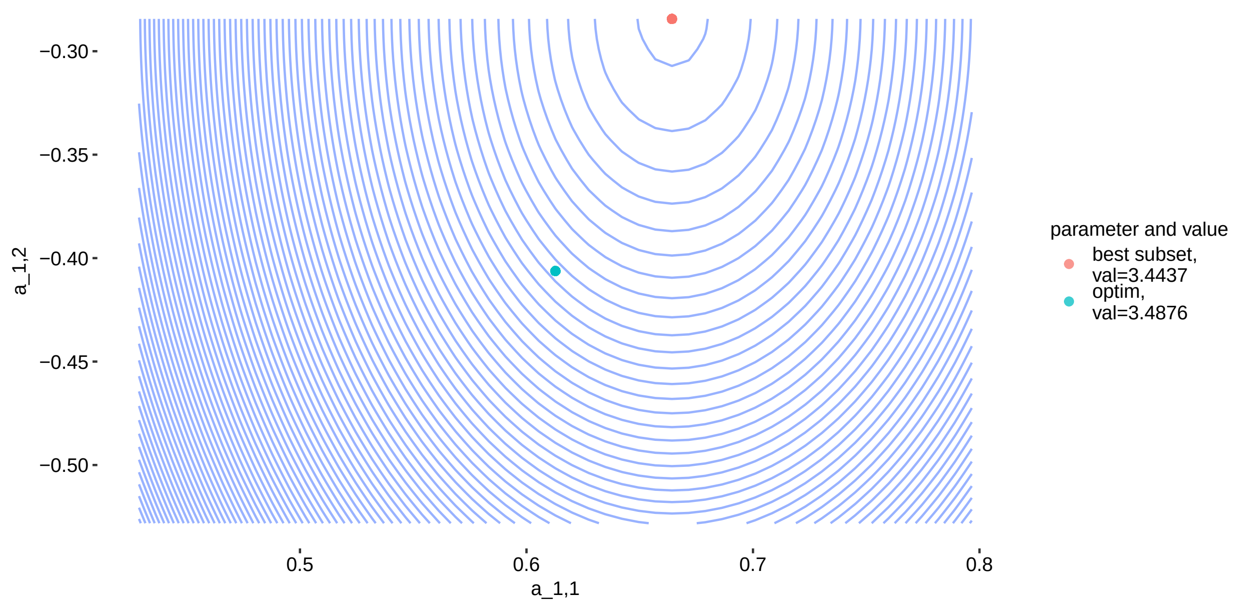

Figure 6: Contour plot the slope parameters for N = 1001:1300.

N = 1001:1030

The best subset parameters in fig. 7 are not within ±30% of the optimized parameters. But the subset parameters are much closer to a global minimum then in the prvious example.

Findings

- The optimization of the parameters must occur with respect to the wanted subset of the simulated data.
- The biggest changes for the optimal subset parameters is for the $a_{1,2}$ parameter. The best subset $a_{1,2}$ parameter is smaller then the optimized value of $a_{1,2}$ in figure 4 and figure 5 respectively.
  The best subset $a_{1,2}$ parameter must be higher than the optimized $a_{1,2}$ parameter opposite in figure 6 and figure 7.
- The total squared prediction error scales well with the number of considered simulated data. This can identicate that the residuals are well normally distributed over the entire simulated process.

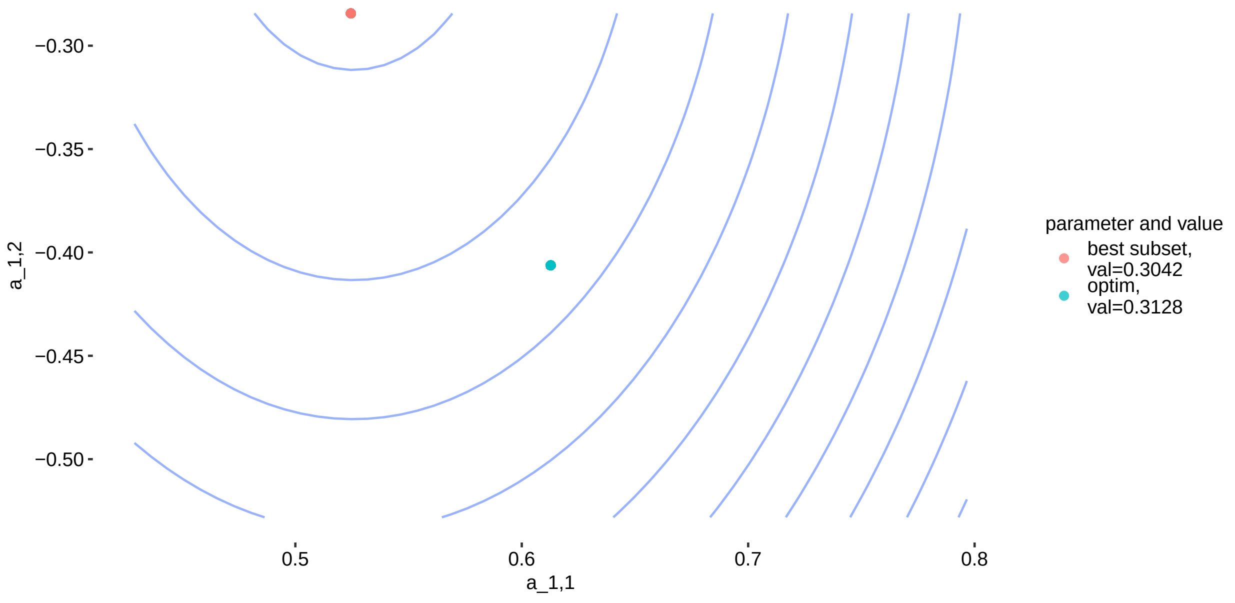

Figure 7: Contour plot the slope parameters for N = 1001:1030.

Part 3

The chosen doubly stochastic model is an AR(2)-AR(4) model given in eqn. 7.

$$\begin{aligned} Y_t &= \sum_{k=1}^{2} \left(\Phi_{t-(1-k)} Y_{t-k}\right) + \epsilon_t \\ \Phi_t - \mu &= \sum_{n=1}^{4} \left(\phi_n \left(\Phi_{t-n} - \mu\right)\right) + \zeta_t \\ \Phi_t &= \sum_{n=1}^{4} \left(\phi_n \left(\Phi_{t-n} - \mu\right)\right) + \zeta_t + \underbrace{\mu\left(1 - \sum_{n=1}^{4}\left(\phi_n\right)\right)}_{\delta_t} \end{aligned} \tag{7}$$

Equation 8 shows eqn. 7 in a reparametrized state space format.

$$\begin{aligned} \begin{pmatrix} \Phi_t \\ \Phi_{t-1} \\ \Phi_{t-2} \\ \Phi_{t-3} \\ \delta_t \end{pmatrix} &= \begin{pmatrix} \phi_1 & \phi_2 & \phi_3 & \phi_4 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \Phi_{t-1} \\ \Phi_{t-2} \\ \Phi_{t-3} \\ \Phi_{t-4} \\ \delta_{t-1} \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \delta_t \\ Y_t &= \begin{pmatrix} Y_{t-1} & Y_{t-2} & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \Phi_t \\ \Phi_{t-1} \\ \Phi_{t-2} \\ \Phi_{t-3} \\ \delta_t \end{pmatrix} + e_t \end{aligned} \tag{8}$$

Simulate

The initial parameters for the simulation are given in eqn. 9.

$$\begin{aligned} n &= 500 \\ \mu &= 0.1 \\ \phi_1 &= 0.099 \\ \phi_2 &= 0.08 \\ \phi_3 &= 0.06 \\ \phi_4 &= 0.085 \\ \zeta &= 0.08 \\ \epsilon &= 0.4 \\ \delta_t &\sim \mathcal{N}(\mu,\, \zeta) \\ e_t &\sim \mathcal{N}(\mu,\, \epsilon) \end{aligned} \tag{9}$$

Figure 8 and figure 9 shows the process and the underlying process respectively.

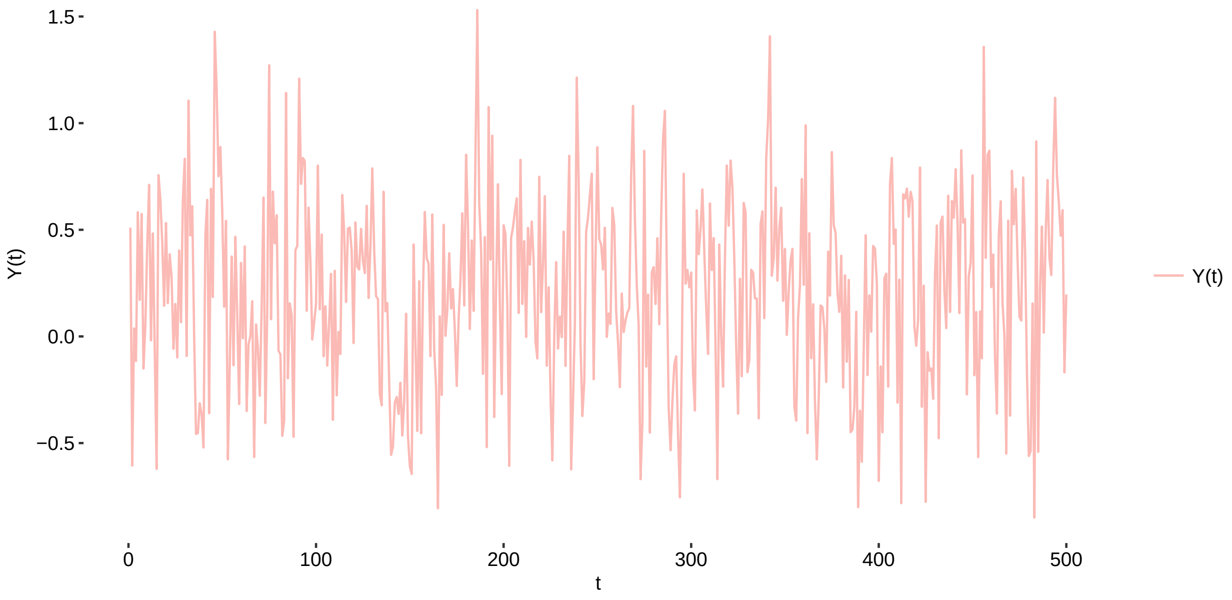

Figure 8: Simulated process of Y(t).

Comment

- Figure 9 shows the underlying process of the doubly stochastic model and its shows a rapidly increase after the first samples where the process more less stabilizes around 0.242.
- The initial parameters ($\phi_{1,2,3,4}$) for this doubly stochastic model is critical and needs to be selected with stationarity in mind. They have been slecected in order to get two stable processes.
  It is possible to modify eqn. 7 to investigate the poles wrt. ϕ. The poles have to be within the unit circle.

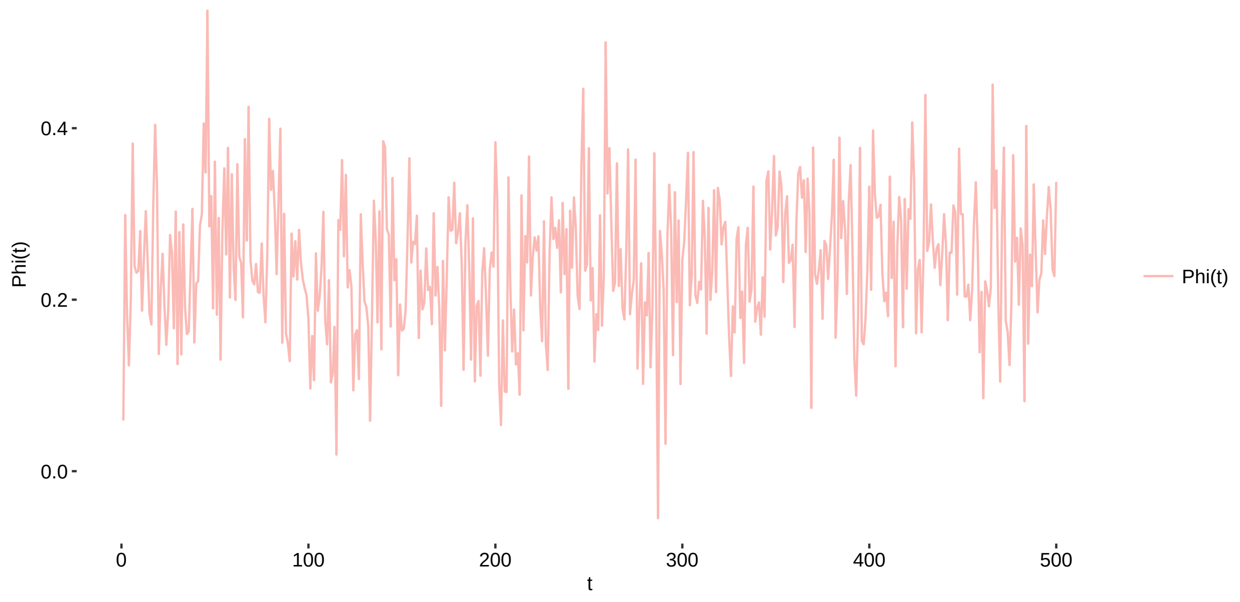

Figure 9: Simulated process of Phi(t).

- AR(4)-process

$$\begin{aligned}\phi(B) &= 1 - 0.099B + 0.08B^2 + 0.06B^3 + 0.085B^4 \\ &\Updownarrow \\ \phi\left(z^{-1}\right) &= 1 - 0.099z^{-1} + 0.08z^{-2} + 0.06z^{-3} + 0.085z^{-4} = 0\end{aligned} \tag{10}$$

  Solve for z in order to find the poles: $z = -0.335335 \pm 0.351428i$ and $z = 0.384835 \pm 0.460593i$. The four poles are within the unit circle and they have the maginitudes 0.4857481 and 0.6002032 respectively.

- AR(2)-process

$$\begin{aligned}\phi(B) &= 1 - 0.099B + 0.08B^2 \\ &\Updownarrow \\ \phi\left(z^{-1}\right) &= 1 - 0.099z^{-1} + 0.08z^{-2} = 0\end{aligned} \tag{11}$$

  Solve for z in order to find the poles: $z = 0.0495 \pm 0.278478i$. The two poles are within the unit circle and they have the maginitudes 0.2828431.

- Pushing the poles further towards the egde of the unit circle will result in longer positive/negative runs in the simulation of the process. The process will return to the mean value as long as $|z| < 1$ is ture for the system.

Part 4

The following model is given (eqn. 12).

$$\begin{aligned} x_{t+1} &= a x_t + v_t \\ y_t &= x_t + e_t \end{aligned} \tag{12}$$

where a is an unknown parameter and v_t and e_t are mutually uncorrelated white noise processes with their variances σ_v^2 and σ_e^2.

Part 4a

The model from eqn. 12 has been reparametrized to state space form in eqn. 13.

$$\begin{aligned} \begin{pmatrix} x_{t+1} \\ a_{t+1} \end{pmatrix} &= \begin{pmatrix} a_t & 0 \\ 0 & a_t \end{pmatrix} \begin{pmatrix} x_t \\ 1 \end{pmatrix} + \begin{pmatrix} v_t \\ 0 \end{pmatrix} \\ y_t &= \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} x_t \\ 1 \end{pmatrix} + e_t \end{aligned} \tag{13}$$

a_t has been included in the state space equation, which makes it possible to estimate the parameter using the Kalman filter.

Simulate

There has been simulated 20 time series with the initial parameters given in eqn. 14.

$$\begin{aligned} a &= 0.4 \\ \sigma_v^2 &= 1 \\ \sigma_e^2 &= 1 \\ v_t &\sim \mathcal{N}(0,\, \sigma_v^2) \\ e_t &\sim \mathcal{N}(0,\, \sigma_e^2) \end{aligned} \tag{14}$$

Part 4b

The following function (`ext_kalman()`) has been used to estimate a for the given sets of initial parameters.

```r
##--------------------------------------------------------------
## EKF algorithm for use in Part 4 of computer exercise 2 in Advanced Time
## Series Analysis
##--------------------------------------------------------------

ext_kalman <- function(y, aInit = 0.5, aVarInit = 1, sigma.v = 1) {
    ## aInit : The starting guess of the AR coefficient estimate aVarInit : The
    ## initial variance for estimation of the AR coefficient sigma.v : Standard
    ## deviation of the system noise of x in the filter

    # Initialize---- Init the state vector estimate
    zt <- c(0, aInit)
    # Init the variance matrices
    Rv <- matrix(c(sigma.v^2, 0, 0, 0), ncol = 2)
    # sigma.e : Standard deviation of the measurement noise in the filter
    Re <- 1

    # Init the P matrix, that is the estimate of the state variance
    Pt <- matrix(c(Re, 0, 0, aVarInit), nrow = 2, ncol = 2)
    # The state is [X a] so the differentiated observation function is
    Ht <- t(c(1, 0))
    # Init a vector for keeping the parameter a variance estimates
    aVar <- rep(NA, length(y))
    # and keeping the states
    Z <- matrix(NA, nrow = length(y), ncol = 2)
    Z[1, ] <- zt

    ## The Kalman filtering----
    for (t in 1:(length(y) - 1)) {
        # Derivatives (Jacobians)
        Ft <- matrix(c(zt[2], 0, zt[1], 1), ncol = 2)  # F_t-1
        # Ht does not change

        ## Prediction step
        zt = c(zt[2] * zt[1], zt[2])  #z_t|t-1 f(z_t-1|t-1)
        Pt = Ft %*% Pt %*% t(Ft) + Rv  #P_t|t-1

        ## Update step
        res = y[t] - zt[1]  # the residual at time t
        St = Ht %*% Pt %*% t(Ht) + Re  # innovation covariance
        Kt = Pt %*% t(Ht) %*% St^-1  # Kalman gain
        zt = zt + Kt * res  # z_t|t
        Pt = (diag(2) - Kt %*% Ht) %*% Pt  #P_t|t

        ## Keep the state estimate
        Z[t + 1, ] <- zt
        ## Keep the P[2,2], which is the variance of the estimate of a
        aVar[t + 1] <- Pt[2, 2]
    }
    return(list(zt = zt, Pt = Pt, Rv = Rv, aVar = aVar, Z = Z))
}
```

There has been executed a small test to find the number of needed samples for converges of the filter estimate. The test is based upon the worst case scenario where the initial variance of the filter and initial variance of the parameter a is 10. The initial value of the parameter is $a = 0.5$.

Figure 10 shows the converges test for the filter estimate. The variance of the parameter and the estimated value of the parameter.

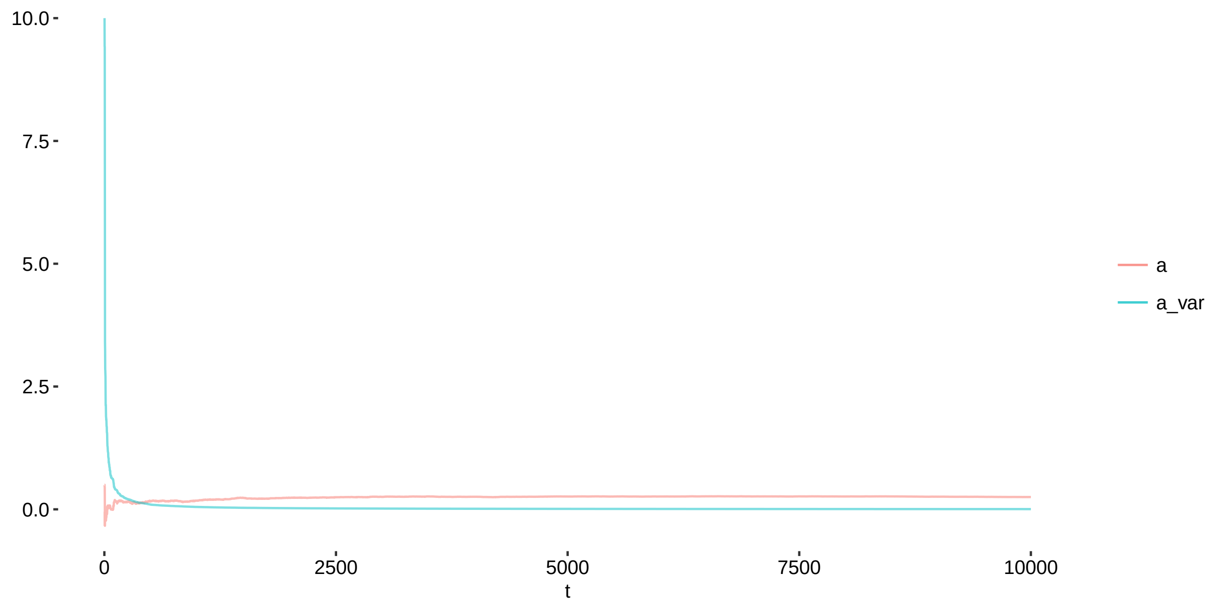

Figure 10: Converges test of the filter estimate.

The filter is converged after the first 3000 samples, (figure 10) and will therefore only consider those first 3000 samples.

Table 2 and table 3 shows the initial variance values for each combination averages over the 20 simulations: The mean estimated value of a (a *mean*), the standard deviation of the estimated a (a *sd*), the mean value of the variance (*a_var mean*) and the standard deviation of the variance (*a_var sd*).

a = 0.5

Table 2 shows the statistics for $a = 0.5$.

Table 2: Statistic for filter estimates.

combination	sigma_vˆ2	sigma_a	a mean	a sd	a_var mean	a_var sd
1	10	1	0.2412421	0.0167668	0.0005955	8.50e-06
2	1	1	0.4013185	0.0217119	0.0002554	1.16e-05
3	10	10	0.2410929	0.0167664	0.0005958	8.60e-06
4	1	10	0.4013258	0.0216101	0.0002554	1.16e-05

a = -0.5

Table 3 shows the statistics for $a = -0.5$.

Table 3: Statistic for filter estimates.

combination	sigma_vˆ2	sigma_a	a mean	a sd	a_var mean	a_var sd
1	10	1	0.2405780	0.0167871	0.0005957	8.60e-06
2	1	1	0.4008391	0.0218301	0.0002564	1.18e-05
3	10	10	0.2410134	0.0167712	0.0005959	8.60e-06
4	1	10	0.4012211	0.0216713	0.0002556	1.17e-05

Comments

According to table 2 and table 3 there is not much of a difference in when selecting different initial parameter value of a. Therefore some common comments.

- The most important parameter, when using filter to estimate parameter(s), is the variance of the filter. The combinations with a low filter variance result in a reasonable parameter estimate, close to the original value of a, regarding the initial value of variance of the parameter.
- In both combinations where $\sigma_v^2 = 1$ is the estimated value of a slightly above the original value of a.

Improvements

Estimate a parameter with an extended Kalman filter will introduce a bias estimate coursed by the properties of the the filter.

- The local linearisation, which introduces the bias, can be removed by adding proper scaled Gaussian noise to the estimated parameter in the state space model (eqn. 13) with an decay of $\frac{1}{t}$.
- By substituting the extended Kalman filter whit an unscented Kalman filter will possibly increase accuracy of the estimated parameter value. The unscented Kalman filter approximates a Gaussian distribution with a deterministic sampling approach and here by captures the mean and covariance, which in most cases outperform the extended Kalman filter[1].
- An different approach could be to use maximum likelihood estimation of the parameter a instead.

[1]Mentioned at the lecture, October 25.