Eco 5316 Time Series Econometrics

Lecture 24 State Space Models

Motivation

- time series models in economics and finance may often be represented in state space form
- state space model consists of a measurement equation relating the observed time series $\boldsymbol{y}_t$ to an unobserved state vector $\boldsymbol{s}_t$ and a state transition equation that describes the evolution of the state vector $\boldsymbol{s}_t$ over time
- state-space model provides a flexible approach to time series analysis, simplifying maximum-likelihood estimation and handling of missing values

Local Level Model

- consider time series y_t satisfying

$$\begin{aligned} y_t &= \mu_t + \varepsilon_t \quad \varepsilon_t \sim N(0, \sigma_\varepsilon^2) \\ \mu_{t+1} &= \mu_t + \zeta_t \quad \zeta_t \sim N(0, \sigma_\zeta^2) \end{aligned}$$

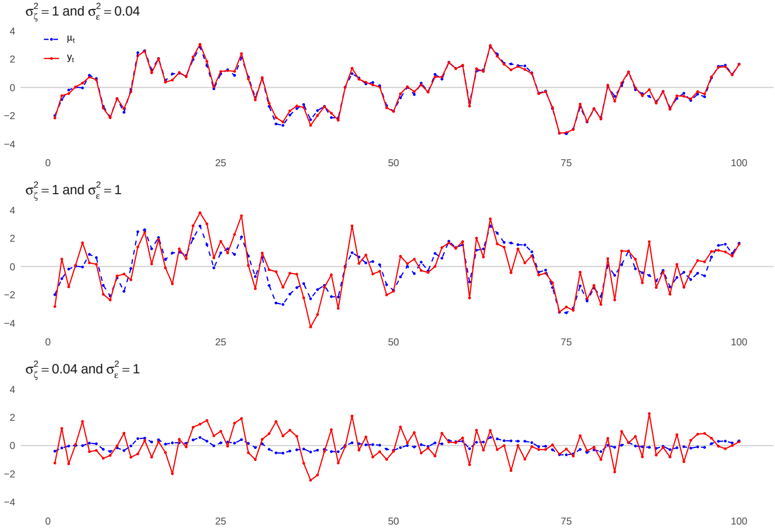
  where $E(\varepsilon_t \zeta_t) = 0$ and μ_1 is either known or drawn from known distribution
- here μ_t is an unobserved random walk, only y_t is observable, with noise ε_t
- the simple model above could be used to model either log of an asset price, log of its volatility, detrended log of national income, detrended log of labor productivity ...
- this model is an example of a linear Gaussian state space model

Local Level Model

note that in the local level model

- since $y_t = \sum_{j=1}^{t} \zeta_j + \varepsilon_t$ innovations ζ_t have a permanent effect on y_t, but innovations ε_t have only a temporary effect on y_t
- if $\sigma_\varepsilon^2 = 0$ then y_t follows a pure random walk
- if $\sigma_\zeta^2 = 0$ then y_t fluctuates around constant mean μ_1

Local Level Model



Local Level Model

- note that the Local Level Model implies that

$$y_{t+1} = y_t + \varepsilon_{t+1} - \varepsilon_t + \zeta_t$$

and so

$$Var(\Delta y_t) = 2\sigma_\varepsilon^2 + \sigma_\zeta^2$$
$$cov(\Delta y_t, \Delta y_{t-1}) = -\sigma_\varepsilon^2$$
$$cov(\Delta y_t, \Delta y_{t-j}) = 0 \quad \text{for } j > 1$$

- the above implies that y_t follows an ARIMA(0,1,1) model

$$y_t = y_{t-1} + a_t + \theta_1 a_{t-1}$$

since

$$Var(\Delta y_t) = (1+\theta_1^2)\sigma_a^2$$
$$cov(\Delta y_t, \Delta y_{t-1}) = \theta_1 \sigma_a^2$$
$$cov(\Delta y_t, \Delta y_{t-j}) = 0 \quad \text{for } j > 1$$

and by setting θ_1 to the solution of $1+\theta_1^2 - \theta_1(2+\sigma_\zeta^2/\sigma_\varepsilon^2) = 0$ we obtain exactly the above variance and covariances

Preview: Filtering, Prediction, Smoothing

- suppose that we have information $F_t = \{y_1, , \ldots, y_t\}$
- if parameters $\sigma_\varepsilon, \sigma_\zeta$ of the Local Level Model are known, the statistical inference we are interested in entails the following:
- **filtering**: recovering μ_t and removing measurement error ε_t
- **prediction**: forecasting μ_{t+h} and y_{t+h} for $h > 0$
- **smoothing**: recovering μ_j for $j < t$

Preview: Filtering, Prediction, Smoothing

notation

- $\mu_{t|j} = E(\mu_t|F_j)$ is the conditional mean of μ_t given F_j
- $\Sigma_{t|j} = Var(\mu_t|F_j)$ is the conditional variance of μ_t given F_j
- $y_{t|j} = E(y_t|F_j)$ is the conditional mean of y_t given F_j
- $v_t = y_t - y_{t|t-1}$ is the one step ahead forecast error
- $V_t = Var(v_t|F_{t-1})$ is the variance of the one step ahead forecast error

Local Level Model

for the Local Level Model we have

- one step ahead forecast

$$y_{t|t-1} = E(y_t|F_{t-1}) = E(\mu_t + \varepsilon_t|F_{t-1}) = \mu_{t|t-1}$$

- one step ahead forecast error

$$v_t = y_t - y_{t|t-1} = y_t - \mu_{t|t-1}$$

- variance of the one step ahead forecast error

$$\begin{aligned} V_t &= Var(v_t|F_{t-1}) = Var(y_t - y_{t|t-1}|F_{t-1}) = Var(y_t - \mu_{t|t-1}|F_{t-1}) \\ &= Var(\mu_t + \varepsilon_t - \mu_{t|t-1}|F_{t-1}) = Var(\mu_t - \mu_{t|t-1}|F_{t-1}) + Var(\varepsilon_t|F_{t-1}) \\ &= \Sigma_{t|t-1} + \sigma_\varepsilon^2 \end{aligned}$$

Kalman Filter for Local Level Model

- goal of Kalman filter is to obtain one step ahead predictions by updating the current estimate of state variable when new data point becomes available
- since innovations ε_t and ζ_t are Gaussian the joint distribution of $(\mu_t, v_t)'$ given F_{t-1} is also Gaussian

$$\begin{bmatrix} \mu_t \\ v_t \end{bmatrix}_{F_{t-1}} \sim N\left(\begin{bmatrix} \boldsymbol{\mu}_{t|t-1} \\ 0 \end{bmatrix}, \begin{bmatrix} \Sigma_{t|t-1} & \Sigma_{t|t-1} \\ \Sigma_{t|t-1} & V_t \end{bmatrix} \right)$$

- conditional distribution of μ_t given F_t is thus $N(\mu_{t|t}, \Sigma_{t|t})$ where

$$\begin{aligned} \mu_{t|t} &= \mu_{t|t-1} + K_t v_t \\ \Sigma_{t|t} &= \Sigma_{t|t-1}(1 - K_t) \end{aligned}$$

  with $K_t = \Sigma_{t|t-1}/V_t$

- finally, given $\mu_{t|t}$ and $\Sigma_{t|t}$ the conditional mean and variance for period $t+1$ state are

$$\begin{aligned} \mu_{t+1|t} &= E(\mu_t + \zeta_t | F_t) = E(\mu_t | F_t) = \mu_{t|t} \\ \Sigma_{t+1|t} &= Var(\mu_{t+1}|F_t) = Var(\mu_t|F_t) + Var(\zeta_t) = \Sigma_{t|t} + \sigma_\zeta^2 \end{aligned}$$

Kalman Filter for Local Level Model

summary of Kalman Filter procedure for local level model:

- parameters σ_ε^2 and σ_ζ^2 are given
- initial state μ_1 is assumed to be distributed as $N(\mu_{1|0}, \Sigma_{1|0})$
- suppose that in period t given previous information F_{t-1} we have conditional mean $\mu_{t|t-1}$ and conditional variance $\Sigma_{t|t-1}$
- after observing y_t we have F_t and obtain conditional mean $\mu_{t+1|t}$ and conditional variance $\Sigma_{t+1|t}$ using

$$\begin{aligned} v_t &= y_t - \mu_{t|t-1} \\ V_t &= \Sigma_{t|t-1} + \sigma_\varepsilon^2 \\ K_t &= \Sigma_{t|t-1} / V_t \\ \mu_{t+1|t} &= \mu_{t|t-1} + K_t v_t \\ \Sigma_{t+1|t} &= \Sigma_{t|t-1}(1 - K_t) + \sigma_\zeta^2 \end{aligned}$$

K_t is called Kalman gain; note that $K_t = \frac{\Sigma_{t|t-1}}{\Sigma_{t|t-1} + \sigma_\varepsilon^2}$, thus if the variance of temporary disturbance ε_t is large then K_t is small and even a large forecasting error leads to only a small update from $\mu_{t|t-1}$ to $\mu_{t|t}$

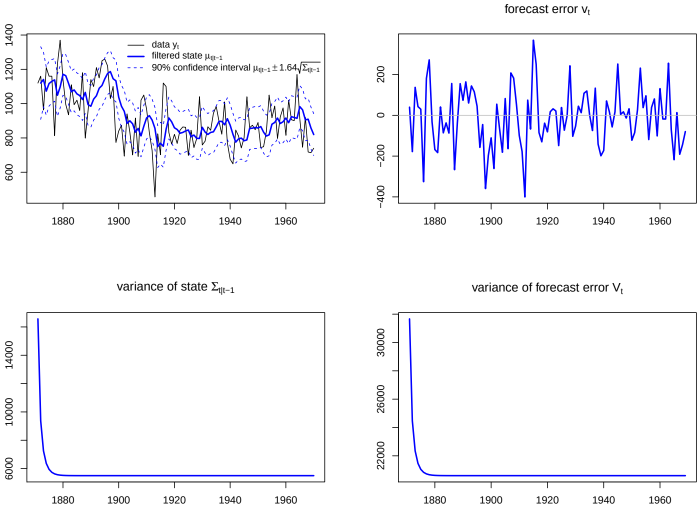
Kalman Filter for Local Level Model - Nile Data

see https://cran.r-project.org/web/packages/KFAS/vignettes/KFAS.pdf for details and examples how to define, estimate and simulate state-space models using KFAS package

```
library(magrittr)
library(KFAS)

y_ts <- datasets::Nile

# specify the state-space local level model
y_LLM <- SSModel(y_ts ~ SSMtrend(degree = 1, Q = list(NA)) , H = NA)
# maximum likelihood estimation of paramaters of Q and H (i.e. variances of the two inovations)
y_LLM_ML <- fitSSM(inits = log(rep(var(y_ts)/1000, 2)) ,  model = y_LLM, method = "BFGS")
# run Kalman Filter and Smoother with estimated parameters
y_LLM_KFS <- KFS(y_LLM_ML$model)

# construct 90% confidence interval for filtered state
y_KF <- predict(y_LLM_ML$model, interval = "confidence", level = 0.9, filtered = TRUE)
y_KF[1,] <- NA
```

```
par(mfrow = c(2,2), cex = 0.8)
cbind(y_ts, y_KF) %>%
    plot.ts(plot.type = "single", col = c(1,4,4,4), lwd = c(1,2,1,1), lty = c(1,1,2,2))
legend("topright", legend = c("data","filtered state","90% confidence interval"),
       col = c(1,4,4), lty = c(1,1,2,2), lwd = c(1,2,1,1), bty = "n", cex=0.9 )
c(y_LLM_KFS$v[-1]) %>% ts(start = 1871) %>%
    plot(main = expression(paste("forecast error ", v["t"])))
abline(h=0,col="grey")
c(y_LLM_KFS$P)[-1] %>% ts(start = 1871) %>%
    plot(main = expression(paste("variance of state ", Sigma["t|t-1"])))
c(y_LLM_KFS$F)[-1] %>% ts(start = 1871) %>%
    plot(main = expression(paste("variance of forecast error ", V["t"])))
```

Kalman Filter for Local Level Model - Nile Data



Kalman Smoothing for Local Level Model

- smoothing is essentially a backward estimation of $\{\mu_1, \ldots, \mu_T\}$ given the information set $F_T = \{y_1, \ldots, y_T\}$
- the objective is thus to obtain conditional distributions $N(\mu_{t|T}, \Sigma_{t|T})$ for $t = T-1, T-2, \ldots, 1$ the procedure is going backward
- $\mu_{t|T}$ is called smoothed state and $\Sigma_{t|T}$ smoothed state variance
- applying the properties of the conditional normal distribution one can derive the following backward recursive algorithm to compute smoothed state variables: using initial value $q_T = 0$ and $M_T = 0$ calculate for $t = T, \ldots, 1$

$$
\begin{aligned}
q_{t-1} &= V_t^{-1} v_t + (1-K_t) q_t \\
\mu_{t|T} &= \mu_{t|t-1} + \Sigma_{t|t-1} q_{t-1} \\
M_{t-1} &= V_t^{-1} + (1-K_t)^2 M_t \\
\Sigma_{t|T} &= \Sigma_{t|t-1} + \Sigma_{t|t-1}^2 M_{t-1}
\end{aligned}
$$

Kalman Filtering and Smoothing for Local Level Model - Nile Data

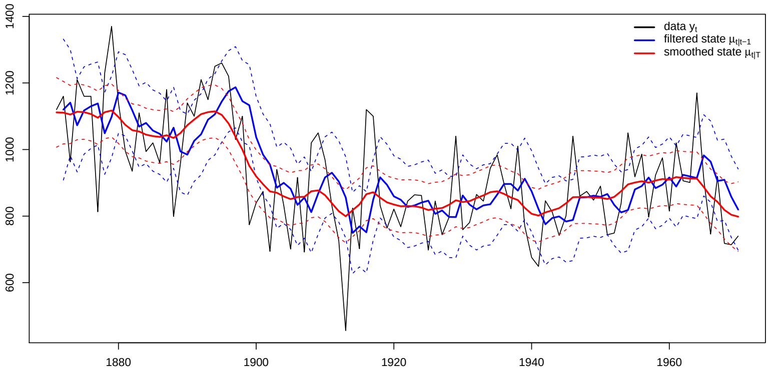
```r
# construct 90% confidence intervals for smoothed state
y_KS <- predict(y_LLM_ML$model, interval = "confidence", level = 0.9)
```

```r
par(mfrow = c(1,1), cex = 0.8)
cbind(y_ts, y_KF, y_KS) %>%
    plot.ts(plot.type = "single", col = c("black","blue","blue","blue","red","red","red"),
            lwd = c(1,2,1,1,2,1,1), lty = c(1,1,2,2,1,2,2), xlab = "", ylab = "", main = "")
legend("topright", legend = c( expression(paste("data ", y["t"])),
                               expression(paste("filtered state ", mu["t|t-1"])),
                               expression(paste("smoothed state ", mu["t|T"])) ),
       col = c("black","blue","red"), lty = 1, lwd = 2, bty = "n")
```

Kalman Filtering and Smoothing for Local Level Model - Nile Data

note that

- smoothed state variable $\mu_{t|T}$ is smoother then filtered state variable $\mu_{t|t-1}$
- confidence intervals for the smoothed state variables are also narrower than those of the filtered state variables



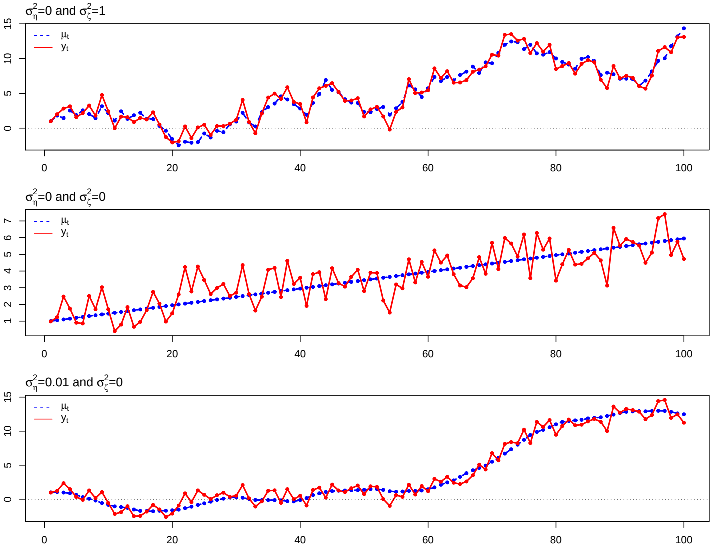
Local Linear Trend Model

- consider time series y_t satisfying

$$\begin{aligned} y_t &= \mu_t + \varepsilon_t & \varepsilon_t &\sim N(0, \sigma_\varepsilon^2) \\ \mu_{t+1} &= \beta_t + \mu_t + \zeta_t & \zeta_t &\sim N(0, \sigma_\zeta^2) \\ \beta_{t+1} &= \beta_t + \eta_t & \eta_t &\sim N(0, \sigma_\eta^2) \end{aligned}$$

  where $\varepsilon_t, \zeta_t, \eta_t$ are independent at all lags and leads, and μ_1, β_1 are either known or from known distribution
- if $\sigma_\eta^2 = 0$, $\sigma_\zeta^2 > 0$ then y_t is random walk with drift β_1
- if $\sigma_\eta^2 = 0$, $\sigma_\zeta^2 = 0$ then y_t fluctuates around a linear trend with slope β_1 and intercept μ_1
- if $\sigma_\eta^2 > 0$, $\sigma_\zeta^2 = 0$ then y_t fluctuates around a non-linear trend

Local Linear Trend Model



Linear Gaussian State Space Model

- local level and local linear trend models are particular cases of linear Gaussian state space model
- linear Gaussian state space model implemented in `KFAS` package

$$\begin{aligned} \boldsymbol{y}_t &= \boldsymbol{Z}_t \boldsymbol{s}_t + \varepsilon_t & \qquad \varepsilon_t &\sim N(\boldsymbol{0}, \boldsymbol{H}_t) \\ \boldsymbol{s}_{t+1} &= \boldsymbol{T}_t \boldsymbol{s}_t + \boldsymbol{R}_t \eta_t & \qquad \eta_t &\sim N(\boldsymbol{0}, \boldsymbol{Q}_t) \end{aligned}$$

where

$\boldsymbol{y}_t$ is an $k \times 1$ vector of observations
$\boldsymbol{s}_t$ is an $m \times 1$ state vector
$\boldsymbol{T}_t$ is an $m \times m$ matrix
$\boldsymbol{R}_t$ is an $m \times n$ matrix
$\boldsymbol{Z}_t$ is an $k \times m$ matrix
η_t is an $n \times 1$ vector
ε_t is an $k \times 1$ vector

with initial state $\boldsymbol{s}_1 \sim N(\boldsymbol{\mu}_{1|0}, \boldsymbol{\Sigma}_{1|0})$ and $E(\varepsilon_t \eta_t') = \boldsymbol{0}$ so innovations in state transition equation and measurement equation are independent

- system matrices $\boldsymbol{T}_t, \boldsymbol{R}_t, \boldsymbol{Z}_t, \boldsymbol{Q}_t$ and $\boldsymbol{H}_t$ can be functions of some parameters $\boldsymbol{\theta}$ that are estimated
- in many cases system matrices are actually time invariant

Linear Gaussian State Space Model

- Kalman Filter algorithm, given initial values $\boldsymbol{\mu}_{1|0}$ and $\boldsymbol{\Sigma}_{1|0}$

$$
\begin{aligned}
\boldsymbol{v}_t &= \boldsymbol{y}_t - \boldsymbol{Z}_t \boldsymbol{s}_{t|t-1} \\
\boldsymbol{V}_t &= \boldsymbol{Z}_t \boldsymbol{\Sigma}_{t|t-1} \boldsymbol{Z}_t' + \boldsymbol{H}_t \\
\boldsymbol{K}_t &= \boldsymbol{T}_t \boldsymbol{\Sigma}_{t|t-1} \boldsymbol{Z}_t' \boldsymbol{V}_t^{-1} \\
\boldsymbol{s}_{t+1|t} &= \boldsymbol{T}_t \boldsymbol{s}_{t|t-1} + \boldsymbol{K}_t \boldsymbol{v}_t \\
\boldsymbol{\Sigma}_{t+1|t} &= \boldsymbol{T}_t \boldsymbol{\Sigma}_{t|t-1} (\boldsymbol{T}_t - \boldsymbol{K}_t \boldsymbol{Z}_t)' + \boldsymbol{R}_t \boldsymbol{Q}_t \boldsymbol{R}_t'
\end{aligned}
$$

- this can be collapsed into two equations

$$
\begin{aligned}
\boldsymbol{s}_{t+1|t} &= \boldsymbol{T}_t \boldsymbol{s}_{t|t-1} + \boldsymbol{T}_t \boldsymbol{\Sigma}_{t|t-1} \boldsymbol{Z}_t' \left(\boldsymbol{Z}_t \boldsymbol{\Sigma}_{t|t-1} \boldsymbol{Z}_t' + \boldsymbol{H}_t \right)^{-1} \left(\boldsymbol{y}_t - \boldsymbol{Z}_t \boldsymbol{s}_{t|t-1} \right) \\
\boldsymbol{\Sigma}_{t+1|t} &= \boldsymbol{T}_t \boldsymbol{\Sigma}_{t|t-1} \boldsymbol{T}_t' - \boldsymbol{T}_t \boldsymbol{\Sigma}_{t|t-1} \boldsymbol{Z}_t' \left(\boldsymbol{Z}_t \boldsymbol{\Sigma}_{t|t-1} \boldsymbol{Z}_t' + \boldsymbol{H}_t' \right)^{-1} \boldsymbol{Z}_t \boldsymbol{\Sigma}_{t|t-1} \boldsymbol{T}_t' + \boldsymbol{R}_t \boldsymbol{Q}_t \boldsymbol{R}_t'
\end{aligned}
$$

Linear Gaussian State Space Model

notation

	Tsay	KFAS
state	$\boldsymbol{s}_t$	$\boldsymbol{\alpha}_t$
conditional mean for state	$\boldsymbol{s}_{t\|t-1}$	$\boldsymbol{a}_t$
smoothed state	$\boldsymbol{s}_{t\|T}$	$\hat{\boldsymbol{\alpha}}_t$
conditional variance of state	$\boldsymbol{\Sigma}_{t\|t-1}$	$\boldsymbol{P}_t$
variance of smoothed state	$\boldsymbol{\Sigma}_{t\|T}$	$\boldsymbol{V}_t$
variance of forecast error	$\boldsymbol{V}_t$	$\boldsymbol{F}_t$
sample size	T	n
measurements	$\boldsymbol{y}_t$ is $k \times 1$ vector	$\boldsymbol{y}_t$ is $p \times 1$ vector
state	$\boldsymbol{s}_t$ is $m \times 1$ vector	$\boldsymbol{\alpha}_t$ is $m \times 1$ vector
transition equation innovation	$\boldsymbol{\eta}_t$ is $n \times 1$ vector	$\boldsymbol{\eta}_t$ is $k \times 1$ vector

Application: CAPM with Time-Varying Coefficients

- state-space model framework allows to easily estimate models with time varying parameters
- consider for example a capital asset pricing model with with time-varying intercept and slope

$$
\begin{aligned}
r_t &= \alpha_t + \beta_t r_{M,t} + \varepsilon_t && \varepsilon_t \sim N(0, \sigma_\varepsilon^2) \\
\alpha_{t+1} &= \alpha_t + \zeta_t && \zeta_t \sim N(0, \sigma_\zeta^2) \\
\beta_{t+1} &= \beta_t + \eta_t && \eta_t \sim N(0, \sigma_\eta^2)
\end{aligned}
$$

where r_t is excess return of an asset, and $r_{M,t}$ is excess return of the market

- to obtain state-space representation rewrite above model in matrix form

$$
\begin{aligned}
r_t &= \begin{bmatrix} 1 & r_{M,t} \end{bmatrix} \begin{bmatrix} \alpha_t \\ \beta_t \end{bmatrix} + \varepsilon_t && \varepsilon_t \sim N(0, \sigma_\varepsilon^2) \\
\begin{bmatrix} \alpha_{t+1} \\ \beta_{t+1} \end{bmatrix} &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha_t \\ \beta_t \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \zeta_t \\ \eta_t \end{bmatrix} && \begin{bmatrix} \zeta_t \\ \eta_t \end{bmatrix} \sim N\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \sigma_\zeta^2 & 0 \\ 0 & \sigma_\eta^2 \end{bmatrix} \right)
\end{aligned}
$$

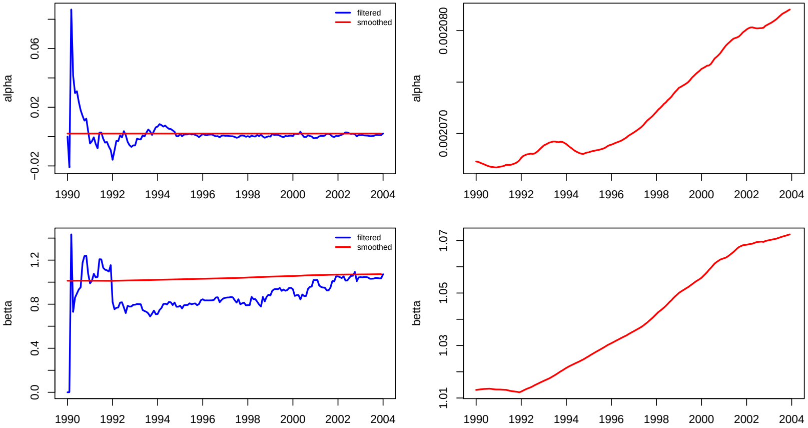
and let $\boldsymbol{y}_t = r_t$, $\boldsymbol{s}_t = (\alpha_t, \beta_t)'$, $\boldsymbol{T}_t = \boldsymbol{R}_t = \boldsymbol{I}_2$, $\boldsymbol{Z}_t = [1, r_{M,t}]$, $\boldsymbol{H}_t = \sigma_\varepsilon^2$, $\boldsymbol{Q}_t = \mathrm{diag}\{\sigma_\zeta^2, \sigma_\eta^2\}$

Application: CAPM with Time-Varying Coefficients

```r
# load data on excess returns from January 1990 to December 2003
#  http://faculty.chicagobooth.edu/ruey.tsay/teaching/fts3/m-excess-c10sp-9003.txt
er_ts <- read.table("m-excess-c10sp-9003.txt", header = TRUE) %>% ts(start = c(1990, 1), frequency = 12)
# extract excess returns for General Motors and for S&P 500
gm <- er_ts[, "GM"]
sp500 <- er_ts[, "SP5"]

# get number of observatons
tobs <- length(sp500)
# construct system matrices for state-space model - a CAPM with time variable alpha and betta
Zt <- array(rbind(rep(1, tobs), sp500), dim = c(1, 2, tobs))
Ht <- matrix(NA)
Tt <- diag(2)
Rt <- diag(2)
Qt <- matrix(c(NA, 0, 0, NA), 2, 2)
# use diffuse prior for initial state
P1inf <- diag(2)

# define state-space CAPM model
y_SS <- SSModel(gm ~ -1 + SSMcustom(Z = Zt, T = Tt, R = Rt, Q = Qt, P1inf = P1inf), H = Ht)
# estimate variances of innovations using maximum likelihood
y_SS_ML <- fitSSM(y_SS, inits = c(0.001,0.001,0.001), method = "BFGS")

# Kalman filtering and smoothing, with parameters in Q and H set to maximum likelihood estimates
y_SS_KFS <- KFS(y_SS_ML$model)

# extract filtered and smoothed alpha and betta
alpha.KFS <- cbind(y_SS_KFS$a[, 1], y_SS_KFS$alphahat[, 1])
betta.KFS <- cbind(y_SS_KFS$a[, 2], y_SS_KFS$alphahat[, 2])
```

Application: CAPM with Time-Varying Coefficients

```
par(mfcol=c(2,2), mar=c(2,4,2,1))
# plot filtered and smoothed state alpha and betta
plot.ts(alpha.KFS, plot.type="single", xlab="",ylab="alpha", col=c("blue","red"), lwd=2)
legend("topright", c("filtered","smoothed"), col=c("blue","red"), lwd=2, cex=0.7, bty="n")
plot.ts(betta.KFS, plot.type="single", xlab="",ylab="betta", col=c("blue","red"), lwd=2)
legend("topright", c("filtered","smoothed"), col=c("blue","red"), lwd=2, cex=0.7, bty="n")
# plot smoothed state alpha and betta
plot.ts(alpha.KFS[,2], plot.type="single", xlab="",ylab="alpha", col="red", lwd=2)
plot.ts(betta.KFS[,2], plot.type="single", xlab="",ylab="betta", col="red", lwd=2)
```



Application: CAPM with Time-Varying Coefficients

- note that smoothed states $\alpha_{t|T}$ and $\beta_{t|T}$ are much smoother than filtered state $\alpha_{t|t-1}$ and $\beta_{t|t-1}$, since Kalman smoothing uses information from the whole sample, but Kalman filtering only information up to period t

Unobserved Components Model

- goal: decomposition of time series into trend, seasonal and irregular component

$$y_t = \mu_t + \gamma_t + \varepsilon_t$$

where

y_t is the observed data
μ_t is an slowly changing component (trend)
γ_t is periodic seasonal component
ε_t is irregular disturbance component

and μ_t, γ_t, ε_t are modeled explicitly as stochastic processes

- note that local level model and local linear trend model are special cases of the unobserved components model with no seasonal component γ_t
- seasonal component can be modeled using time varying dummy variables as

$$(1 + B + \ldots + B^{s-1})\gamma_{t+1} = \omega_t \qquad \omega_t \sim N(0, \sigma_\omega^2)$$

so that in expectation the sum of the seasonal effects captured by dummy variables $\gamma_t, \gamma_{t-1}, \ldots, \gamma_{t-s+1}$ is zero

Application: Quarterly earnings per share of Johnson & Johnson

- local linear trend model with seasonal component

$$
\begin{aligned}
y_t &= \mu_t + \gamma_t + \varepsilon_t & \varepsilon_t &\sim N(0, \sigma^2_\varepsilon) \\
\mu_{t+1} &= \beta_t + \mu_t + \zeta_t & \zeta_t &\sim N(0, \sigma^2_\zeta) \\
\beta_{t+1} &= \beta_t + \eta_t & \eta_t &\sim N(0, \sigma^2_\eta) \\
(1 + B + B^2 + B^3)\gamma_{t+1} &= \omega_t & \omega_t &\sim N(0, \sigma^2_\omega)
\end{aligned}
$$

- seasonal dummy approach: $\gamma_{t+1} = -\sum_{j=0}^{2} \gamma_{t-j} + \omega_t$

Application: Quarterly earnings per share of Johnson & Johnson

- to obtain state-space representation rewrite the above model in matrix form

$$\underbrace{y_t}_{\boldsymbol{y}_t} = \underbrace{\begin{bmatrix} 1 & 0 & 1 & 0 & 0 \end{bmatrix}}_{\boldsymbol{Z}_t} \underbrace{\begin{bmatrix} \mu_t \\ \beta_t \\ \gamma_t \\ \gamma_{t-1} \\ \gamma_{t-2} \end{bmatrix}}_{\boldsymbol{s}_t} + \underbrace{\varepsilon_t}_{\boldsymbol{\varepsilon}_t}$$

$$\underbrace{\begin{bmatrix} \mu_{t+1} \\ \beta_{t+1} \\ \gamma_{t+1} \\ \gamma_t \\ \gamma_{t-1} \end{bmatrix}}_{\boldsymbol{s}_{t+1}} = \underbrace{\begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & -1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}}_{\boldsymbol{T}_t} \underbrace{\begin{bmatrix} \mu_t \\ \beta_t \\ \gamma_t \\ \gamma_{t-1} \\ \gamma_{t-2} \end{bmatrix}}_{\boldsymbol{s}_t} + \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{\boldsymbol{R}_t} \underbrace{\begin{bmatrix} \zeta_t \\ \eta_t \\ \omega_t \end{bmatrix}}_{\boldsymbol{\eta}_t}$$

where

$$\varepsilon_t \sim N(0, \underbrace{\sigma_\varepsilon^2}_{\boldsymbol{H}_t})$$

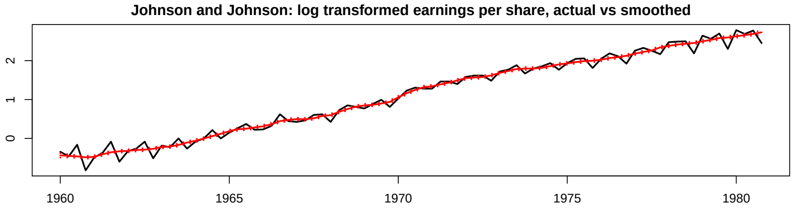
$$\begin{bmatrix} \zeta_t \\ \eta_t \\ \omega_t \end{bmatrix} \sim N\left(\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \underbrace{\begin{bmatrix} \sigma_\zeta^2 & 0 & 0 \\ 0 & \sigma_\eta^2 & 0 \\ 0 & 0 & \sigma_\omega^2 \end{bmatrix}}_{\boldsymbol{Q}_t} \right)$$

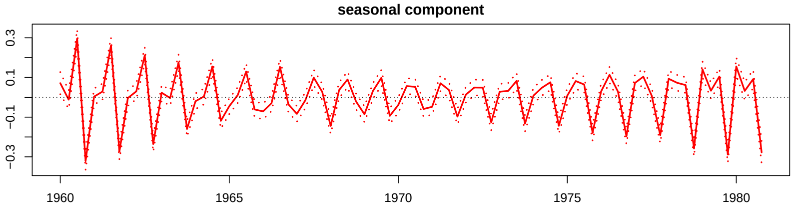
Application: Quarterly earnings per share of Johnson & Johnson

```r
# import quarterly data on earnings per share for Johnson and Johnson available at
#  http://faculty.chicagobooth.edu/ruey.tsay/teaching/fts3/q-jnj.txt
y_ts <- scan(file = "http://faculty.chicagobooth.edu/ruey.tsay/teaching/fts3/q-jnj.txt") %>%
    ts(start = c(1960, 1), frequency = 4) %>% log()

# define a local level model with seasonal component
y_LLT <- SSModel(y_ts ~ SSMtrend(degree = 2,Q = rep(list(NA), 2))
                        + SSMseasonal(period = 4,sea.type = "dummy", Q = NA), H = NA)

# estimate model parameters using maximum likelihood
y_LLT_ML <- fitSSM(y_LLT, inits = log( rep(var(y_ts)/100,4) ), method = "Nelder-Mead")

# construct 90% confidence intervals for smoothed state
y_KS_lvl <- predict(y_LLT_ML$model, states = "level",
                    level = 0.9, interval = "confidence", filtered = FALSE)
y_KS_sea <- predict(y_LLT_ML$model, states = "seasonal",
                    level = 0.9, interval = "confidence", filtered = FALSE)
```

Application: Quarterly earnings per share of Johnson & Johnson

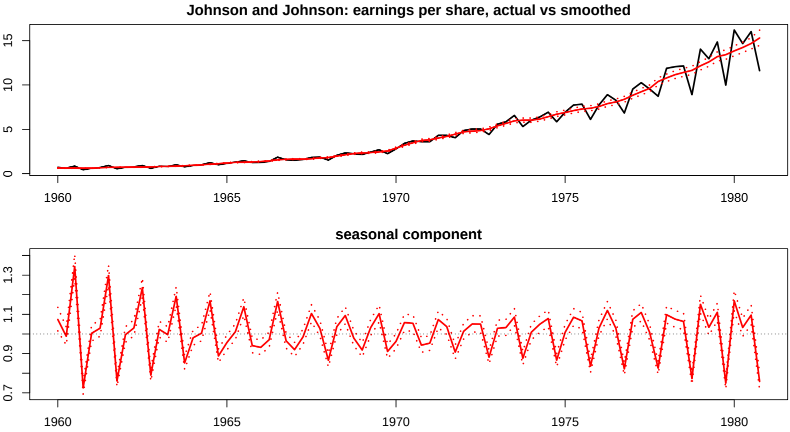
```r
par(mfrow = c(2,1), mar = c(3,3,2,1), cex = 0.9)
cbind(y_ts, y_KS_lvl) %>%
    plot.ts(plot.type = "single", col = c(1,2,2,2), lty = c(1,1,3,3), lwd = 2, xlab = "", ylab = "",
            main = "Johnson and Johnson: log transformed earnings per share, actual vs smoothed" )
y_KS_sea %>%
    plot.ts(plot.type = "single", col = 2, lty = c(1,3,3), lwd = 2, xlab = "", ylab = "",
            main = "seasonal component" )
abline(h = 0, lty = 3)
```





Application: Quarterly earnings per share of Johnson & Johnson

```
par(mfrow=c(2,1), mar=c(3,3,2,1), cex=0.9)
cbind(y_ts, y_KS_lvl) %>% exp() %>%
    plot.ts(plot.type = "single", col = c(1,2,2,2), lty = c(1,1,3,3), lwd = 2, xlab = "", ylab = "",
            main = "Johnson and Johnson: earnings per share, actual vs smoothed")
y_KS_sea %>% exp() %>%
    plot.ts(plot.type = "single", col = 2, lty = c(1,3,3), lwd = 2, xlab = "", ylab = "",
            main = "seasonal component")
abline(h = 1, lty = 3)
```



Missing Values

Kalman Filtering and Smoothing can easily deal with missing data

case 1:

- observations for all variables in $\boldsymbol{y}$ are missing for some periods
- thus no new information available at these time points; Kalman filtering and smoothing procedures remains same but with

$$\boldsymbol{v}_t = \mathbf{0} \qquad \boldsymbol{K}_t = \mathbf{0}$$

for periods with missing values

case 2:

- some components of $\boldsymbol{y}$ are missing for some periods
- let $\boldsymbol{y}_t^* = \boldsymbol{J}\boldsymbol{y}_t$ be the vector of observed data, where $\boldsymbol{J}_t$ are the rows of $k \times k$ identity matrix corresponding to observed variables
- Kalman filtering and smoothing procedure remain same, but observation equation for periods with missing data is replaced with

$$\boldsymbol{y}_t^* = \boldsymbol{c}_t^* + \boldsymbol{Z}_t^* \boldsymbol{s}_t + \varepsilon_t^*$$

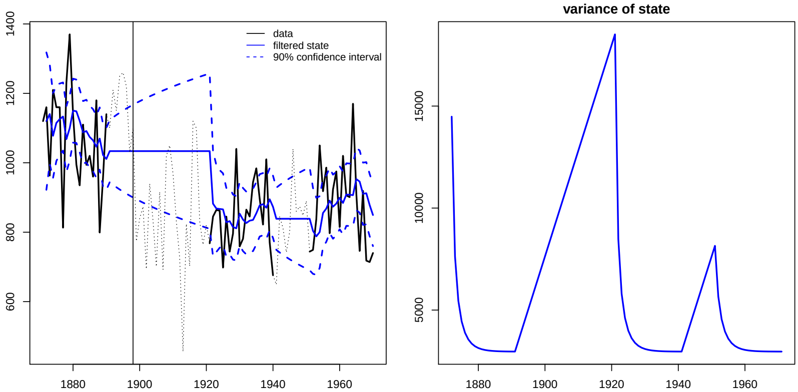
where $\boldsymbol{c}_t^* = \boldsymbol{J}\boldsymbol{c}_t$, $\boldsymbol{Z}_t^* = \boldsymbol{J}\boldsymbol{Z}_t$, $\varepsilon_t^* = \boldsymbol{J}\varepsilon_t$ and $\boldsymbol{H}_t^* = \boldsymbol{J}\boldsymbol{H}_t\boldsymbol{J}'$

Application: Local Level Model for Nile Data with Missing Values

```r
# annual flow of the river Nile at Ashwan 1871-1970
y_ts <- datasets::Nile

# create missing values
y_ts[21:50] <- NA
y_ts[71:80] <- NA

# define the state-space local level model
y_LLM <- SSModel(y_ts ~ SSMtrend(1, Q = list(NA)), H = NA)

# maximum likelihood estimation of paramaters of Q and H
initvals <- rep(var(y_ts, na.rm = TRUE), 2)/10000
y_LLM_ML <- fitSSM(model = y_LLM, inits = initvals, method = "BFGS")

# Kalman filtering and smoothing
y_KFS <- KFS(y_LLM_ML$model)

# confidence intervals for filtered and smoothed state
y_KF <- predict(y_LLM_ML$model, interval = "confidence", level = 0.9, filtered = TRUE)
y_KS <- predict(y_LLM_ML$model, interval = "confidence", level = 0.9)

# replace filtered state for first period by NA
y_KF[1,] <- NA
```

Application: Local Level Model for Nile Data with Missing Values

```r
par(mfrow = c(1,2), mar = c(3,3,2,1), cex = 0.8)
cbind(y_ts, y_KF, Nile) %>%
    plot.ts(plot.type = "single", col = c(1,4,4,4,1), lwd = c(2,2,2,2,1), lty = c(1,1,2,2,3),
            xlab = "", ylab = "", main = "")
abline(v = 1898)
legend("topright", legend = c("data","filtered state","90% confidence interval"),
       col = c(1,4,4), lty = c(1,1,2), lwd = c(1,1,1), bty = "n", cex = 0.9)
c(y_KFS$P)[-1] %>% ts(start=1872) %>%
    plot( , col = "blue", lwd = 2, xlab = "", ylab = "", main = "variance of state")
```



Application: Local Level Model for Nile Data with Missing Values

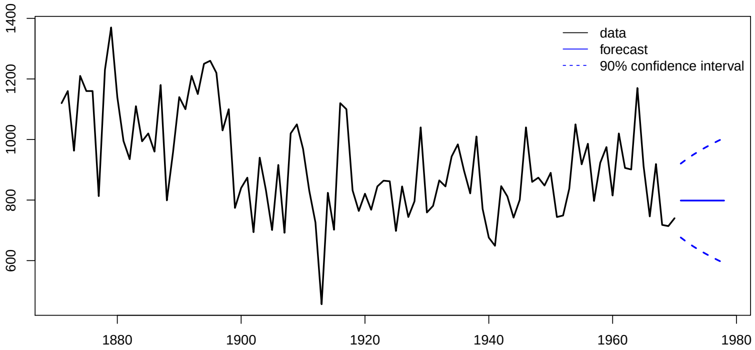
- local level model implies that the filtered state $y_{t|t-1}$ remains constant during the period where no additional information is obtained due to missing values
- the variance of the filtered state is increasing and confidence intervals are getting larger during the period with missing observations
- the error can thus be quite large if a structural break occurs during the period with missing data, due to an event like here the construction of dam in 1898

Forecasting with State Space Models

- essentially identical to having missing observations at the end of the sample
- usual Kalman filter recursion is thus performed, but on an extended sample with missing observations added at the end of the sample (number of missing observations added is the same as the desired forecast horizon)

Application: Local Level Model for Nile Data

```r
# forecast horizon
h <- 8
# create forecast
y_f <- predict(y_LLT_ML$model, interval = "confidence", level = 0.9, n.ahead = h, filtered = TRUE)

# plot the forecast
cols <- c(1,4,4,4)
ltys <- c(1,1,2,2)
cbind(y_ts, y_f) %>%
    plot(plot.type = "single", col = cols, lwd = 2, lty = ltys, xlab = "", ylab = "", main = "")
legend("topright", c("data","forecast","90% confidence interval"), col = cols, lty = ltys, bty = "n")
```



Application: Quarterly earnings per share of Johnson & Johnson

```r
# forecast horizon
h <- 16
# create forecast
y_f <- predict(y_LLT_ML$model, interval = "confidence", level = 0.9, n.ahead = h)

par(mfcol=c(3,1), cex=0.9, mar=c(3,2,2,2))
cols <- c(1,4,4,4)
lwds <- c(2,2,1,1)
ltys <- c(1,1,2,2)

# log
cbind(y_ts, y_f) %>%
    plot.ts(plot.type = "single", col = cols, lwd = lwds, lty = ltys, xlab = "", ylab = "",
            main = "Johnson and Johnson: log transformed earnings per share")
legend("topleft", legend = c("actual data","forecast","90% confidence interval"),
       col = cols, lwd = lwds, lty = ltys, bty = "n", cex = 0.8)

# log-change
cbind(y_ts, y_f) %>% diff() %>%
    plot.ts(plot.type = "single", col = cols, lwd = lwds, lty = ltys, xlab = "", ylab = "",
            main="Johnson and Johnson: change in log transformed earnings per share")
legend("topleft", legend = c("actual data","forecast","90% confidence interval"),
       col = cols, lwd = lwds, lty = ltys, bty = "n", cex = 0.8)

# levels
cbind(y_ts, y_f) %>% exp() %>%
    plot.ts(plot.type = "single", col = cols, lwd = lwds, lty = ltys, xlab = "", ylab = "",
            main="Johnson and Johnson: earnings per share")
legend("topleft", legend = c("actual data","forecast","90% confidence interval"),
       col = cols, lwd = lwds, lty = ltys, bty = "n", cex = 0.8)
```

Application: Quarterly earnings per share of Johnson & Johnson