Seminario GARSF

Durbin-Koopman:
Modelo de nivel local

Viernes 23 de diciembre de 2022

2.6 Simulación del modelo

Generar muestras del nivel local: $Y_t^+ = (y_1^+, \ldots, y_n^+)$ I

Este es el caso más sencillo

1. Se generan observaciones $\varepsilon_t^+ \sim \mathcal{N}\left(0, \sigma_\varepsilon^2\right)$ y $\eta_t^+ \sim \mathcal{N}\left(0, \sigma_\eta^2\right)$
2. Se utilizan las recursiones

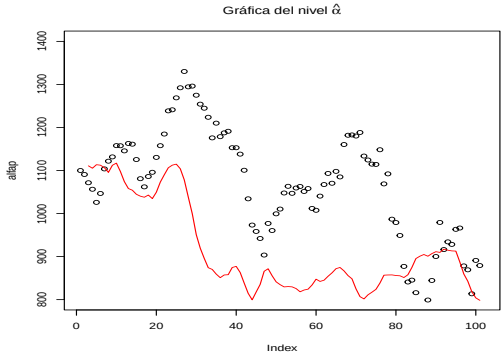
$$\begin{aligned} y_t^+ &= \alpha_t^+ + \varepsilon_t^+ \\ \alpha_{t+1} &= \alpha_t^+ + \eta_t^+ \end{aligned}$$

para $t = 1, \ldots, n$ y α_1^+ dado.

```
source("../DKCap2.R") # trabajo previo
# Graficar el nivel
alfap <- vector(mode = "numeric", length = 101)
alfap[1] <- 1100 # valor inicial
sigma_eta = sqrt(1469.1)
set.seed(56) # fija semilla aleatoria para generar muestra de errores.
etap <- rnorm(100, sd = sigma_eta)

for (i in 1:101){
alfap[i+1] <- alfap[i] + etap[i]
}
plot(alfap, ylim = c(800,1400), main = expression(paste("Gráfica del nivel ", hat(alpha))))
lines(a$alfahat, col = "red")
```

Generar muestras del nivel local: $Y_t^+ = (y_1^+, \ldots, y_n^+)$ II



- En este ejemplo, se usa como valor inicial el valor de la serie, y se puede ver que la serie simulada no se parece a la serie suavizada

Muestras de los errores condicionales a las observaciones disponibles $\varepsilon_t|Y_n$ para $t = 1, \dots, n$ I

El proceso de generar errores condicionales es mucho más elaborado. Se requieren varias estimaciones:

1. Se generan observaciones $\varepsilon_t^+ \sim \mathcal{N}\left(0, \sigma_\varepsilon^2\right)$ y $\eta_t^+ \sim \mathcal{N}\left(0, \sigma_\eta^2\right)$
2. Se estiman los errores condicionales $\varepsilon_t|Y_n$ como

$$\tilde{\varepsilon}_t = \varepsilon_t^+ - \hat{\varepsilon}_t^+ + \hat{\varepsilon}_t$$

donde:

- $\hat{\varepsilon}_t = \mathrm{E}(\varepsilon_t|Y_t)$, que se calcula con $\varepsilon_t = \sigma_t^2 u_t$, $u_t = F_t^{-1}\nu_t - K_t r_t$
- $\hat{\varepsilon}_t^+ = \mathrm{E}(\varepsilon_t|Y_n^+)$ con $Y_n^+ = (y_1^+, \dots, y_n^+)$, por lo que se requiere aplicar otra vez el filtro para simular estos nuevos valores, y se calcula del mismo modo que el caso previo.

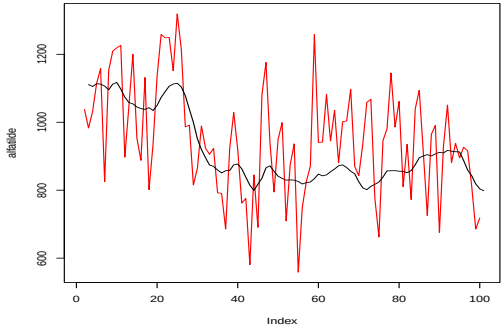
3. Una vez que se obtienen los valores $\tilde{\varepsilon}_1, \dots, \tilde{\varepsilon}_n$ se pueden calcular los niveles $\tilde{\alpha}_t = y_t - \tilde{\varepsilon}_t$ y los errores $\tilde{\eta}_t = \tilde{\alpha}_{t+1} - \tilde{\alpha}_t$ para $t = 1, \dots, n$.

Muestras de los errores condicionales a las observaciones disponibles $\varepsilon_t|Y_n$ para $t = 1, \ldots, n$ II

```r
source("../DKCap2.R") # trabajo previo
A <- RecKalman(y)
sigma_eta <- sqrt(1469.1)
sigma_eps <- sqrt(15099)
n <- 100
epshat <- yp <- numeric(n)
set.seed(56) # fija semilla aleatoria para generar muestra de errores.
etap <- rnorm(n, sd = sigma_eta)
epsp <- rnorm(n, sd = sigma_eps)
B <- as.data.frame(A$rec)
for (i in 2:(n+1)){
yp[i] <- alfap[i] + epsp[i]
alfap[i+1] <- alfap[i] + etap[i]
epshat[i] <- sigma_eta^2*(B$vt[i]/B$Ft[i]-B$Kt[i]*B$rt[i])
}
ap <- llm(yp[-101], a = 0, P = 10e7, sigma2_eps = 15099, sigma2_eta = 1469.1)
vt <- yp - a$y
epshatp <- sigma_eta^2*(vt/ap$F_ - ap$K*ap$r)

epstilde <- c(epsp,NA) - epshatp + epshat
alfatilde <- a$y - epstilde
etatilde <- diff(alfatilde)
plot(alfatilde,type = "l", col = "red")
lines(a$alfahat, type = "l")
```

Muestras de los errores condicionales a las observaciones disponibles $\varepsilon_t|Y_n$ para $t = 1, \dots, n$ III



2.7 Valores Faltantes

Estimación con valores faltantes I

- Se considera que $\{y_t\}$ es la serie observada con $j \in \{\tau, \dots, \tau^* - 1\}$ los índices con valores faltantes.
- Hay dos opciones para tratar el caso de valores faltantes, pero la conclusión importante es que el problema se reduce a considerar que $K_t = 0$ para los valores t en donde hay datos faltantes.
- Para $t = \tau, \dots, \tau^* - 1$ se tiene:

$$\begin{aligned}
\mathrm{E}(\alpha_t|Y_t) = \mathrm{E}(\alpha_t|Y_{\tau-1}) &= \mathrm{E}(\alpha_\tau + \sum_{j=\tau}^{t-1} \eta_j|Y_{\tau-1}) = a_\tau \\
\mathrm{E}(\alpha_{t+1}|Y_t) = \mathrm{E}(\alpha_{t+1}|Y_{\tau-1}) &= \mathrm{E}(\alpha_\tau + \sum_{j=\tau}^{t} \eta_j|Y_{\tau-1}) = a_\tau \\
\mathrm{Var}(\alpha_t|Y_t) = \mathrm{Var}(\alpha_t|Y_{\tau-1}) &= \mathrm{Var}(\alpha_\tau + \sum_{j=\tau}^{t-1} \eta_j|Y_{\tau-1}) = P_\tau + (t-\tau)\sigma_\eta^2 \\
\mathrm{Var}(\alpha_{t+1}|Y_t) = \mathrm{Var}(\alpha_t|Y_{\tau-1}) &= \mathrm{Var}(\alpha_\tau + \sum_{j=\tau}^{t} \eta_j|Y_{\tau-1}) = P_\tau + (t-\tau+1)\sigma_\eta^2
\end{aligned}$$

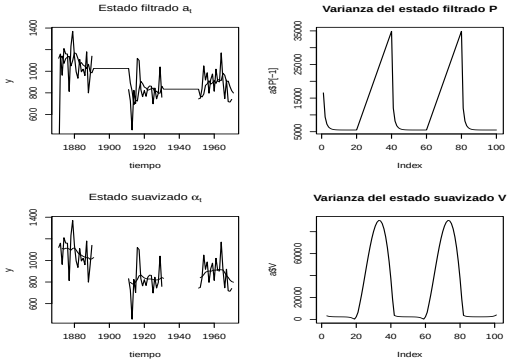
Estimación con valores faltantes II

- Los valores finales se calculan recursivamente con las siguientes recursiones para $t = \tau, \ldots, \tau^* - 1$:
  - $a_{t|t} = a_{t+1} = a_t$
  - $P_{t|t} = P_t$
  - $P_{t+1} = P_t + \sigma_\eta^2$

  Para el resto de los valores el cálculo es el usual. Esto es equivalente a tomar $K_t = 0$ para los valores faltantes.

```
y <- datasets::Nile
y[c(21:40,61:80)] <- NA
a <- llm(y, a = 0, P = 10e7, sigma2_eps = 15099, sigma2_eta = 1469.1)
par(mfrow=c(2,2))
plot(y, type = "l", main  = expression(paste("Estado filtrado ", a[t])), xlab = "tiempo")
lines(1871:1971,a$a)
plot(a$P[-1], type = "l", main = "Varianza del estado filtrado P")
# Gráfica del suavizado
plot(y, type = "l", main = expression(paste("Estado suavizado ",alpha[t])), xlab = "tiempo")
lines(1871:1971,a$alfahat)
plot(a$V, type = "l", main = "Varianza del estado suavizado V")
```

Estimación con valores faltantes III

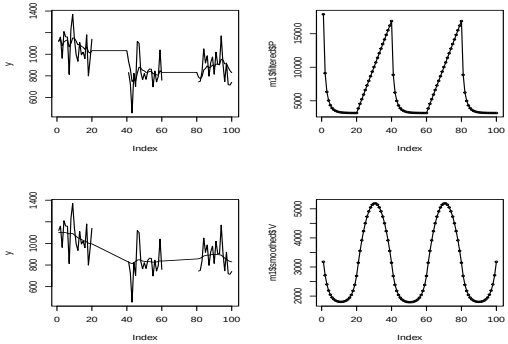


Intento con paquete `statespacer` I

- El paquete `statespacer` implementa los procedimientos del libro. Es importante especificar una condición inicial para el proceso de optimización y estimación.

```
library(statespacer)
y <- Nile
y[c(21:40,61:80)] <- NA  # introduce los valores faltantes
y <- matrix(y)           # requiere que se consideren matrices en los inputs
m1 <- statespacer(y = y, local_level_ind = T, initial = 0.5*log(var(y,na.rm=T)))
par(mfrow=c(2,2))
plot(y, type = "l")
lines(m1$filtered$a)
plot(m1$filtered$P, type = "o", cex = 0.5)
plot(y, type = "l")
lines(m1$smoothed$a)
plot(m1$smoothed$V, type ="o", cex = 0.5)
```

Intento con paquete `statespacer` II



2.8 Pronósticos

Pronósticos I

- Sea $\bar{y}_{n+j} = \arg\min_\theta \mathrm{E}((y_{n+j} - \theta | Y_n)$ el pronóstico de error cuadrático medio (ECM) mínimo, para $j = 1, \dots, J$ con J fija de antemano.
- Se sabe bien que la solución al problema anterior es la media condicional, $\bar{y}_{n+j} = \mathrm{E}(y_{n+j}|Y_n)$
- La varianza del pronóstico de ECM mínimo es $\hat{F}_{n+j} = \mathrm{Var}(y_{n+j}|Y_n)$.

El pronóstico en el modelo de nivel local se obtiene usando las recursiones del filtro sobre la serie $y_1, \dots, y_n, y_{n+1}, \dots y_{n+J}$, considerando las últimas J observaciones como *faltantes*.

Las recursiones para los pronósticos quedan de la forma:

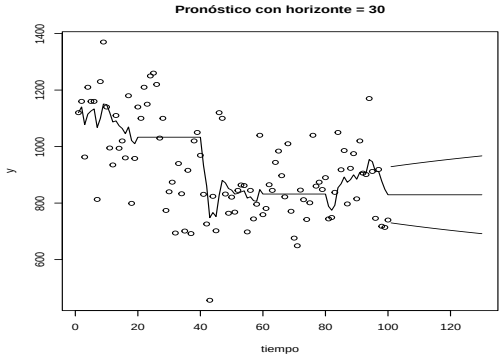
- $\bar{a}_{n+j+1} = \bar{a}_{n+j}, j = 1, \dots, J-1; \bar{a}_{n+1} = a_{n+1}$.
- $\bar{P}_{n+j+1} = \bar{P}_{n+j} + \sigma_\eta^2, j = 1, \dots, J-1, \bar{P}_{n+1} = P_{n+1}$
- $\bar{y}_{n+j} = \bar{a}_{n+j}, j = 1, \dots, J$
- $\bar{F}_{n+j} = \bar{P}_{n+j} + \sigma_\varepsilon^2, j = 1, \dots, J$

Pronósticos II

En este caso, `statespacer` ya incluye el cálculo del pronóstico como parte de su salida. Como en el texto, se considera el intervalo de 50 % de confianza, pero no se incluye la estimación de la varianza.

```
# Pronóstico
h <- predict(m1, forecast_period = 30)
ypred <- c(Nile,h$y_fc)
plot(c(Nile,rep(NA,30)), ylab = "y", xlab = "tiempo", main = "Pronóstico con horizonte = 30")
lines(c(m1$filtered$a,h$a_fc))
lines(c(rep(NA,100), as.vector(h$y_fc + qnorm(.25,lower.tail = F)*as.vector(sqrt(h$Fmat_fc)))))
lines(c(rep(NA,100), as.vector(h$y_fc - qnorm(.25,lower.tail = F)*as.vector(sqrt(h$Fmat_fc)))))
```

Pronósticos III




```
# estimación de la varianza
as.vector(h$P_fc)
```

```
 [1]  3864.691  4550.324  5235.956  5921.589  6607.222  7292.855  7978.488  8664.120  9349.753 10035.386 10721.019 11406.652 12092.284 12777.917 13463.550
[16] 14149.183 14834.815 15520.448 16206.081 16891.714 17577.347 18262.979 18948.612 19634.245 20319.878 21005.511 21691.143 22376.776 23062.409 23748.042
```

2.9 Inicialización

Inicialización

- Para inicializar el filtro, se toma $\alpha_1 \sim \mathcal{N}(a_1, P_1)$ con a_1 y P_1 dados.
- Se puede considerar para inicializar el proceso, un enfoque clásico o Bayesiano, en ambos casos se tiene la misma solución.
  - En el enfoque Bayesiano, asumiendo una distribución difusa para α_1 con a_1 fijo y $P_1 \to \infty$, entonces, recordando $v_1 = y_1 - a_1$ y $F_1 = P_1 + \sigma_\varepsilon^2$:

$$a_2 = a_1 - K_1 v_1 \quad = \quad a_1 + \frac{P_1}{P_1 + \sigma_\varepsilon^2}(y_1 - a_1) \to y_1$$

$$P_2 = P_1(1 - K_1) + \sigma_\eta^2 \quad = \quad P_1\left(1 - \frac{P_1}{P_1 + \sigma_\varepsilon^2}\right) + \sigma_\eta^2 = \frac{P_1}{P_1 + \sigma_\varepsilon^2}\sigma_\varepsilon^2 + \sigma_\eta^2 \to \sigma_\varepsilon^2 + \sigma_\eta^2$$

  y se procede con el filtro para $t = 2, \ldots, n$
  - Bajo el enfoque clásico y estimando bajo máxima verosimilitud, se toma $\alpha_1 \sim \mathcal{N}(y_1, \sigma_\varepsilon^2)$

2.10 Estimación de parámetros

Estimación de parámetros I

- La distribución conjunta de $Y_n = (y_1, \dots, y_n)$ está dada por

$$p(Y_n) = \prod_{t=1}^{n} p(y_t|Y_{t-1}), p(y_1|Y_0) = p(y_1)$$

- Como $y_t|Y_{t-1} \sim \mathcal{N}(a_t, F_t)$ y $v_t = y_t - a_t$, tomando logaritmos:

$$\log L = \log p(Y_n) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\sum_{t=1}^{n}\left(\log(F_t) + \frac{v_t^2}{F_t}\right)$$

Estimación de parámetros II

- También se puede derivar directamente considerando su representación vectorial, recordando $Y_n \sim \mathcal{N}_n(a_1\mathbf{1}, \Omega)$:

$$\log(L) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log(|\Omega|) - \frac{1}{2}(Y_n - a_1\mathbf{1})'\Omega^{-1}(Y_n - a_1\mathbf{1})$$

Considerando la descomposición de Cholesky $\Omega = CFC'$, y que $|C| = 1$, $\Omega^{-1} = C'F^{-1}C$ y $v = C(Y_n - a_1\mathbf{1})$ se tiene que:

$$\log(|\Omega|) = \log(|CFC'|) = \log(|C||F||C'|) = \log(|F|)$$

y

$$(Y_n - a_1\mathbf{1})'\Omega^{-1}(Y_n - a_1\mathbf{1}) = v'Fv$$

Pero $\log(|F|) = \sum_{t=1}^{n}\log(F_t)$ y $v'F^{-1}v = \sum_{t=1}^{n}F_t^{-1}v_t^2$.

Estimación de parámetros III

- Para el caso difuso, se separa la observación del tiempo $t = 1$ para remover la influencia de P_1. Se define la *verosimilitud difusa* como la función

$$\begin{aligned}
\log(L_d) &= \lim_{P_1 \to \infty} \left(\log(L) + \frac{1}{2} \log(P_1) \right) \\
&= \underbrace{-\frac{1}{2} \lim_{P_1 \to \infty} \left(\log(F_1/P_1) + \frac{v_1^2}{F_1} \right)}_{= 0, \text{ ya que } F_1/P_1 \to 1 \text{ y } v_1^2/F_1 \to 0} - \frac{n}{2} \log(2\pi) - \frac{1}{2} \sum_{t=2}^{n} \left(\log(F_t) + \frac{v_t^2}{F_t} \right) \\
&= -\frac{n}{2} \log(2\pi) - \frac{1}{2} \sum_{t=2}^{n} \left(\log(F_t) + \frac{v_t^2}{F_t} \right)
\end{aligned}$$

- Como P_1 no depende de las varianzas σ_ε^2 y σ_η^2 sus optimizadores son los mismos en las dos formas de la verosimilitud.

Estimación de parámetros IV

- los máximos se encuentran numéricamente.
  Con el paquete `statespacer` podemos estimar los parámetros σ_ε y σ_η de la siguiente manera:

```
y <- Nile
y <- matrix(y)            # requiere que se consideren matrices en los inputs
m1 <- statespacer(y = y, local_level_ind = T, initial = 0.5*log(var(y,na.rm=T)))
exp(m1$optim$par) # valores de sigma_eps y sigma_eta
```

```
[1] 122.88308  38.32394
```

2.11 Estado estable del proceso

Estado estable del procesos

- Cuando $n \to \infty$ el proceso converge a un estado estable, ya que la ecuación recursiva

$$\bar{P} = \bar{P}\left(1 - \frac{\bar{P}}{\bar{P} + \sigma_\varepsilon^2} + \sigma_\eta^2\right)$$

es equivalente a la cuadrática

$$x^2 - xq - q = 0,$$

con solución $x = (q + \sqrt{q^2 + 4q})/2$, positiva cuando $q > 0$, donde $x = \bar{P}/\sigma_\varepsilon^2$ y $q = \sigma_\eta^2/\sigma_\varepsilon^2$

- En el estado estable $F_t \to \bar{P} + \sigma_\varepsilon^2$ y $K_t \to \frac{\bar{P}}{\bar{P}+\sigma_\varepsilon^2}$.

2.12 Diagnósticos del modelo

Diagnósticos

- Para los diagnósticos, debemos revisar:
  - normalidad de los errores de ε_t y η_t y de los errores de pronóstico $e_t = \frac{v_t}{\sqrt{F_t}}$
  - independencia serial
  - Homoscedasticidad

  en todos los casos lo hacemos a través de los respectivos residuales. Se pueden considerar los residuales suavizados estandarizados:

$$u_t^* = \frac{\hat{\varepsilon}_t}{\sqrt{\text{Var}(\hat{\varepsilon}_t)}} = D_t^{-1/2} u_t,$$

$$r_t^* = \frac{\hat{\eta}_t}{\sqrt{\text{Var}(\hat{\eta}_t)}} = N_t^{-1/2} r_t, t = 1, \dots, n$$

- También es importante verificar por outliers y cambios estructurales, a través de los residuales. s

Pruebas de diagnóstico para errores de pronóstico I

- Bajo los supuestos del modelo,

$$S \overset{n\to\infty}{\sim} \mathcal{N}(0, 6/n) \qquad K \overset{n\to\infty}{\sim} \mathcal{N}(3, 24/n)$$

donde $S = \frac{m_3}{\sqrt{m_2^3}}$ es la asimetría y $K = \frac{m_4}{m_2^2}$ es la curtosis, estimada por momentos

$m_q = \frac{1}{n}\sum_{t=1}^{n}(e_t - m_1)^q$

El paquete `statespacer` muestra los elementos de diagnóstico del modelo.

```
y <- Nile
y <- matrix(y)           # requiere que se consideren matrices en los inputs
m1 <- statespacer(y = y, local_level_ind = T, initial = 0.5*log(var(y,na.rm=T)))
par(mfrow=c(1,3))
plot(as.vector(m1$diagnostics$v_normalised),type="l",
main = "error de predicción estandarizado",ylab = "e")
plot(as.vector(m1$diagnostics$e)/sqrt(as.vector(m1$diagnostics$D)),type="l",
main = "residual de la observacion u_t", ylab = "u_t")
plot(as.vector(m1$diagnostics$r)/sqrt(as.vector(m1$diagnostics$N)),type="l",
main = "residual de la observacion r_t", ylab = "r_t")
```

Pruebas de diagnóstico para errores de pronóstico II