Introdução à Inferência Bayesiana

Démerson André Polli

ENAP - 14/01/2020

Modelos de Espaço de Estados (I)

O modelo *Dynamic Linear Model* pode ser definido na forma

$$\boldsymbol{y}_t = \boldsymbol{F}_t\boldsymbol{\theta}_t + \boldsymbol{\nu}_t, \quad \boldsymbol{\nu}_t \sim N(\mathbf{0}, \boldsymbol{V}_t)$$

$$\boldsymbol{\theta}_t = \boldsymbol{G}_t\boldsymbol{\theta}_{t-1} + \boldsymbol{\omega}_t, \quad \boldsymbol{\omega}_t \sim N(\mathbf{0}, \boldsymbol{W}_t)$$

$$\boldsymbol{\theta}_0 \sim N(m_0, C_0)$$

- Estes modelos estão implementados no pacote `dlm` no `R`.
- O pacote `dlm` permite o ajuste por *máxima verossimilhança* ou *inferência bayesiana* de modelos de espaço de estados.
  - *`dLmModARMA()`: processo ARMA*
  - *`dLmModPoly()`: modelos polinomiais*
  - *`dLmModReg()`: regressão linear*
  - *`dLmModSeas()`: modelo sazonal (periódico)*
  - *`dLmModTrig()`: modelo sazonal (trigonométrico)*

Modelos de Espaço de Estados (II)

Modelo **autoregressivo de ordem p** - AR(p):

$$\boldsymbol{F}_t = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \end{bmatrix}, \quad \nu_t = 0$$

$$\boldsymbol{G}_t = \begin{bmatrix} \phi_1 & 1 & 0 & \cdots & 0 & 0 \\ \phi_2 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \phi_{p-1} & 0 & 0 & \cdots & 0 & 1 \\ \phi_p & 0 & 0 & \cdots & 0 & 0 \end{bmatrix}, \quad \omega_t = \begin{bmatrix} \sigma_\omega^2 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

Modelos de Espaço de Estados (IV)

Modelo de **média móvel de ordem q**:

$$\boldsymbol{F}_t = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \end{bmatrix}, \quad \nu_t = 0$$

$$\boldsymbol{G}_t = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 0 \end{bmatrix},$$

$$\omega_t = \begin{bmatrix} 1 & \phi_1 & \phi_2 & \cdots & \phi_q \\ \phi_1 & \phi_1^2 & \phi_1\phi_2 & \cdots & \phi_1\phi_q \\ \phi_2 & \phi_1\phi_2 & \phi_2^2 & \cdots & \phi_2\phi_q \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \phi_q & \phi_1\phi_q & \phi_2\phi_q & \cdots & \phi_q^2 \end{bmatrix}$$

Modelos de Espaço de Estados (V)

Modelo **ARMA(p, q)**

$$\boldsymbol{F}_t = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \end{bmatrix}, \quad \nu_t = 0$$

$$\boldsymbol{G}_t = \begin{bmatrix} \phi_1 & 1 & 0 & \cdots & 0 & 0 \\ \phi_2 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \phi_{p-1} & 0 & 0 & \cdots & 0 & 1 \\ \phi_p & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 \end{bmatrix},$$

$$\omega_t = \sigma^2 \begin{bmatrix} 1 & \phi_1 & \phi_2 & \cdots & \phi_q \\ \phi_1 & \phi_1^2 & \phi_1\phi_2 & \cdots & \phi_1\phi_q \\ \phi_2 & \phi_1\phi_2 & \phi_2^2 & \cdots & \phi_2\phi_q \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \phi_q & \phi_1\phi_q & \phi_2\phi_q & \cdots & \phi_q^2 \end{bmatrix}$$

Modelo de Espaço de estados (VI)

Modelo polinomial de ordem p:

$$\boldsymbol{F}_t = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \end{bmatrix}, \quad \nu_t = \sigma_\nu^2$$

$$\boldsymbol{G}_t = \begin{bmatrix} 1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix},$$

$$\omega_t = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & \sigma_\omega^2 \end{bmatrix}$$

Modelos de Espaço de Estados (V)

```
if(!require("dlm")) install.packages("dlm")
if(!require("numDeriv")) install.packages("numDeriv")

library(dlm)
library(numDeriv)
```

Modelos de Espaço de Estados (VI)

Ajuste de máxima verossimilhança do modelo polinomial (nível local):

```
buildNile = function(theta) {
  dlmModPoly(order = 1, dV = theta[1], dW = theta[2])
}

(fit = dlmMLE(Nile, parm = c(100, 2), buildNile, lower = rep(1e-4, 2)))
```

```
## $par
## [1] 15099.787  1468.438
##
## $value
## [1] 549.6918
##
## $counts
## function gradient
##       32       32
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

Modelos de Espaço de Estados (VII)

O código abaixo calcula a variância dos parâmetros. Observe que as variâncias ficaram demasiadamente grandes. Isto indica que o ajuste por *máxima verossimilhança* não está bom.

```
hs = hessian(function(x) dlmLL(Nile, buildNile(x)), fit$par)
all(eigen(hs, only.values = TRUE)$values > 0)
```

```
## [1] TRUE
```

```
aVar = solve(hs)
sqrt(diag(aVar))
```

```
## [1] 3146.003 1280.180
```

Modelos de Espaço de Estados (VIII)

Os componentes de variância do modelo podem ser obtidos a partir da função `buildNile()`.

```
modNile <- buildNile(fit$par)
sqrt(drop(V(modNile)))
```

```
## [1] 122.8812
```

```
sqrt(drop(W(modNile)))
```

```
## [1] 38.3202
```

Modelos de Espaço de Estados (IX)

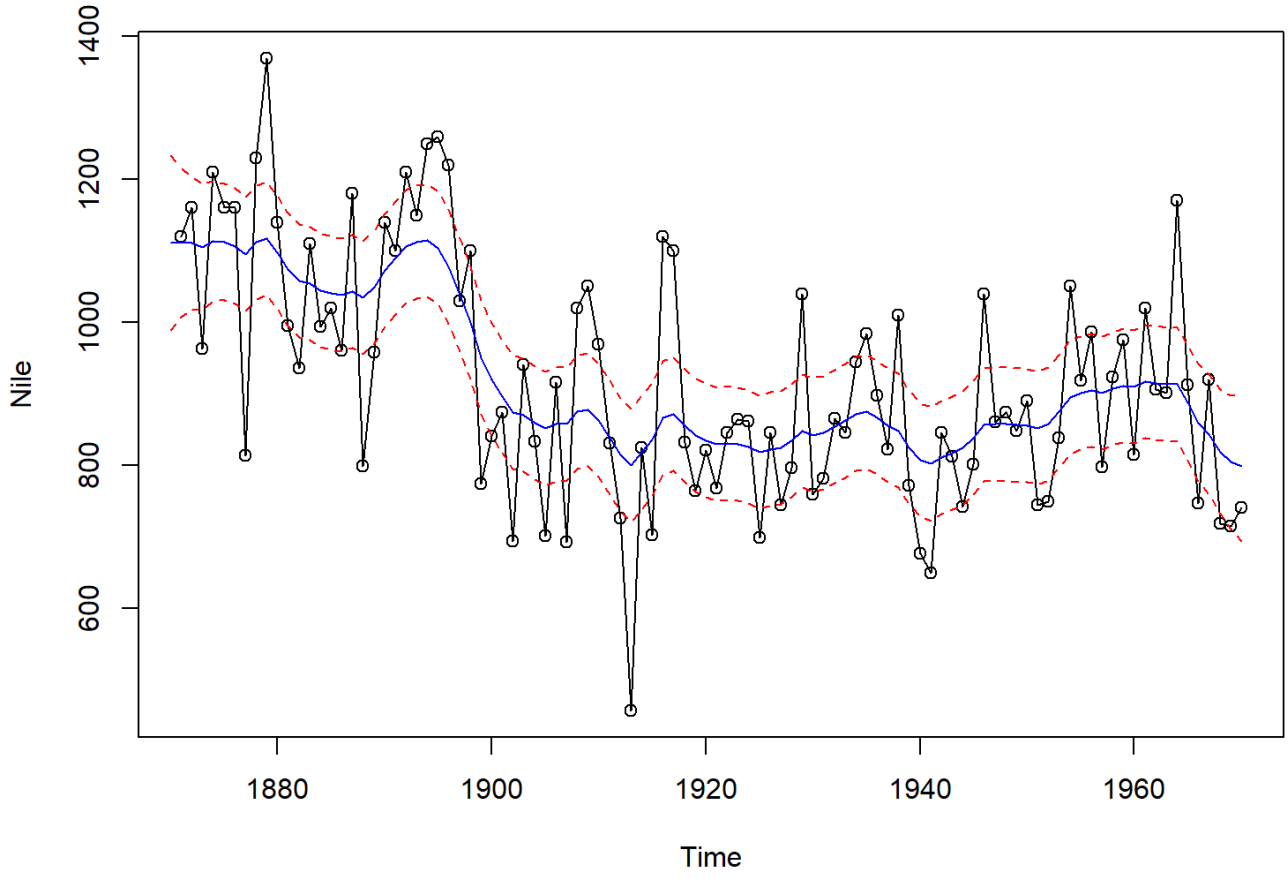
Utilizando um Filtro de Kalman é possível estimar a banda de previsão da série suavizada.

```
smoothNile <- dlmSmooth(Nile, modNile)
hwidth <- qnorm(0.05, lower = FALSE) *
  sqrt(unlist(dlmSvd2var(smoothNile$U.S, smoothNile$D.S)))
sm <- cbind(smoothNile$s, as.vector(smoothNile$s) + hwidth %o% c(-1, 1))
```

Modelos de Espaço de Estados (X)

Utilizando um Filtro de Kalman é possível estimar a banda de previsão da série suavizada.

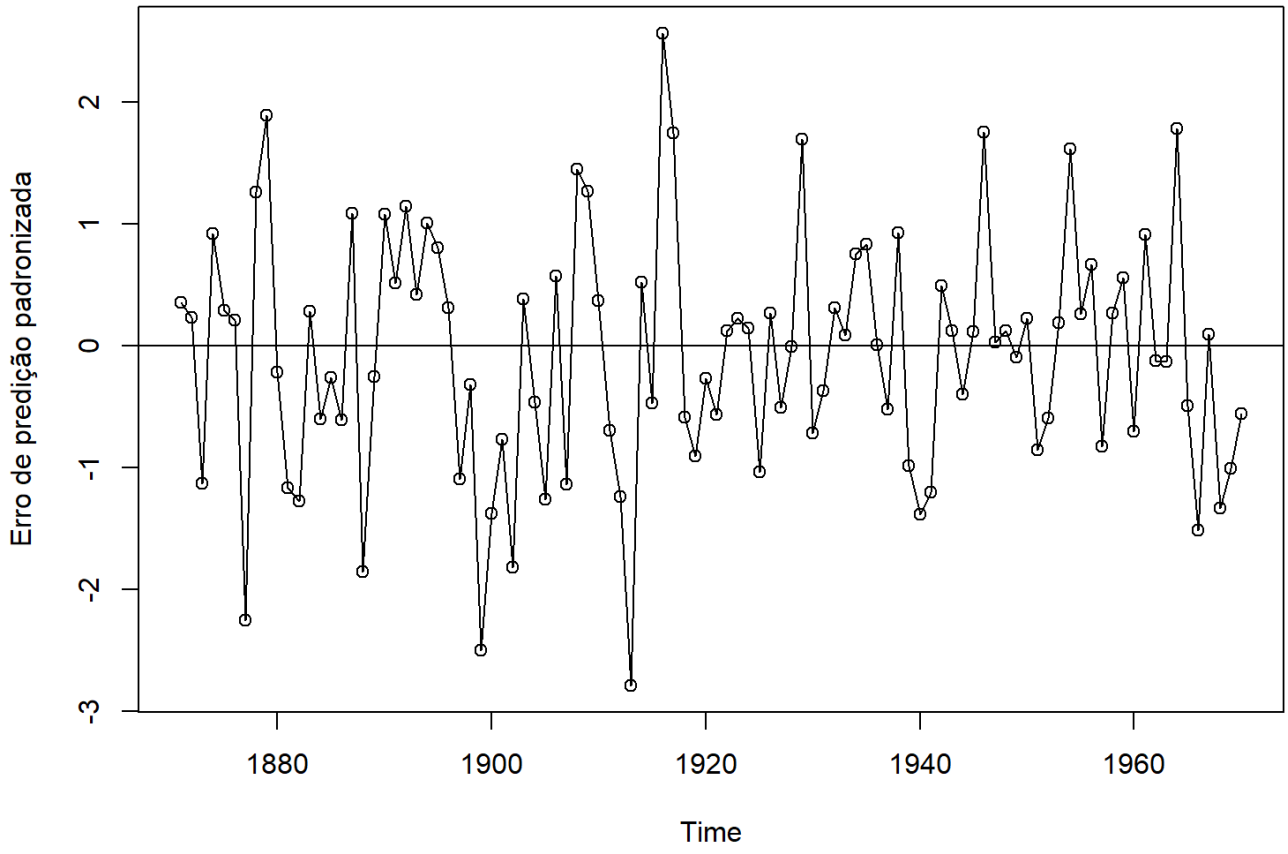
```
plot(Nile, type = "o")
lines(sm[,1], col = "blue")
lines(sm[,2], col = "red", lty = 2)
lines(sm[,3], col = "red", lty = 2)
```



Modelos de Espaço de Estados (X)

Utilizando um Filtro de Kalman é possível também estimar o erro de previsão do modelo:

```
filterNile <- dlmFilter(Nile, modNile)
plot(residuals(filterNile, sd = FALSE), type = "o",
        ylab = "Erro de predição padronizada")
abline(h = 0)
```



Modelos de Espaço de Estados (XI)

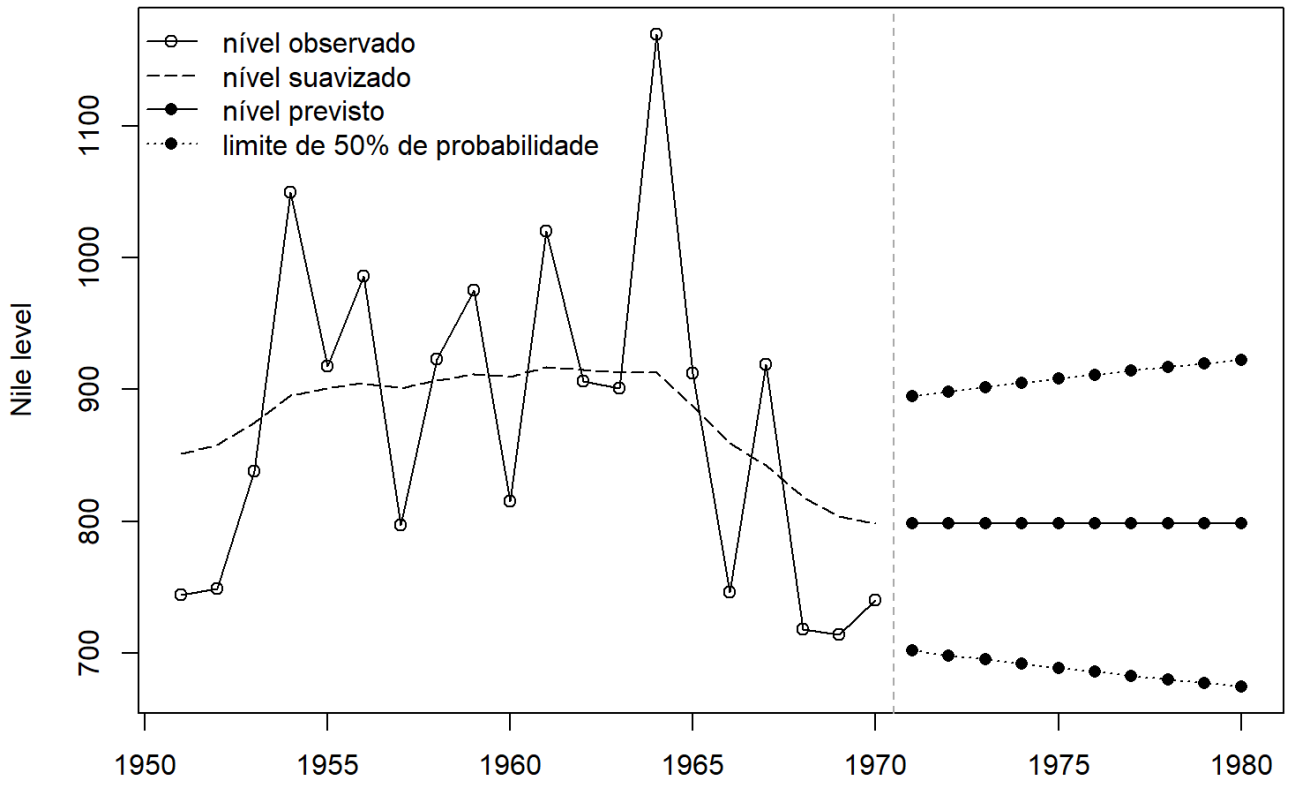
A previsão da série é feita com a função `dlmForecast()` aplicada ao filtro de Kalman:

```
foreNile <- dlmForecast(filterNile, nAhead = 10)
attach(foreNile)
hwidth <- qnorm(0.25, lower = FALSE) * sqrt(unlist(Q))
fore <- cbind(f, as.vector(f) + hwidth %o% c(-1, 1))
rg <- range(c(fore, window(Nile, start = c(1951, 1))))
plot(fore, type = "o", pch = 16, plot.type = "s", lty = c(1, 3, 3),
        ylab = "Nile level", xlab = "", xlim = c(1951, 1980), ylim = rg)
lines(window(Nile, start = c(1951, 1)), type = 'o')

lines(window(smoothNile$s, start = c(1951,1)), lty = 5)
abline(v = mean(c(time(f)[1], tail(time(Nile), 1))),
          lty = "dashed", col = "darkgrey")
legend("topleft", lty = c(1, 5, 1, 3), pch = c(1, NA, 16, 16), bty = "n",
          legend = c("nível observado", "nível suavizado", "nível previsto",
                        "limite de 50% de probabilidade"))
detach(foreNile)
```

Modelos de Espaço de Estados (XII)

A previsão da série é feita com a função `dlmForecast()` aplicada ao filtro de Kalman:



Modelos de Espaço de Estados (XIII)

O exemplo abaixo mostra como ajustar o modelo bayesiano:

```
set.seed(123)
gibbsOut <- dlmGibbsDIG(Nile, mod = dlmModPoly(1), shape.y = 0.1,
                           rate.y = 0.1, shape.theta = 0.1,
                        rate.theta = 0.1, n.sample = 10000,
                           thin = 9)

burn <- 1:1000
```

Modelos de Espaço de Estados (XIV)

```
attach(gibbsOut)

opar = par(mfrow = c(2, 2))
ts.plot(ergMean(dV[-burn]), ylab = "sample mean", xlab = "iterations",
           main = "obs variance")
ts.plot(ergMean(dW[-burn]), ylab = "sample mean", xlab = "iterations",
           main = "evolution variance")
acf(dV[-burn])
acf(dW[-burn])
par(opar)
detach(gibbsOut)
```

Modelos de Espaço de Estados (XV)

```
attach(gibbsOut)

opar = par(mfrow = c(3, 1))
plot(density(dV[-burn]), xlim = c(2000, 34000), ylab = "", main = "")
hist(dV[-burn], prob = TRUE, add = TRUE)
curve(dgamma(1/x, shape = 0.1, rate = 0.1) / x^2, lty = "dashed",
         add = TRUE)

plot(density(dW[-burn]), ylab = "", xlim = c(0, 16000), main = "")
hist(dW[-burn], prob = TRUE, add = TRUE)
curve(dgamma(1/x, shape = 0.1, rate = 0.1) / x^2, lty = "dashed",
         add = TRUE)

plot(dV[-burn], dW[-burn], pch = ".", cex = 1.5, ylab = "")

par(opar)
detach(gibbsOut)
```

Modelos de Espaço de Estados (XVII)

```
attach(gibbsOut)

mcmcMean(dV[-burn])
mcmcMean(dW[-burn])
quantile(dV[-burn], c(0.025, 0.975))
quantile(dW[-burn], c(0.025, 0.975))

detach(gibbsOut)
```

Modelos de Espaço de Estados (XIX)

```
attach(gibbsOut)

mcmcMean(dV[-burn])
mcmcMean(dW[-burn])
quantile(dV[-burn], c(0.025, 0.975))
quantile(dW[-burn], c(0.025, 0.975))

detach(gibbsOut)
```

Próximos passos

- Como incluir covariáveis?
- Como incluir séries de maior frequência como exógena?
- Como combinar os modelos?
- Como modelar séries multivariadas?
- Aplicação aos dados econométricos.