LABORATORY OF APPLIED MATHEMATICAL PROGRAMMING AND STATISTICS

StateSpaceModels.jl

Modelando séries temporais (e outras coisas) em Julia via modelos de espaço de estados

Disponível em https://github.com/LAMPSPUC/StateSpaceModels.jl

Raphael Saavedra, Guilherme Bodin e Mario Souto
LAMPS PUC-Rio

StateSpaceModels.jl

Pacote Julia para modelagem, previsão e simulação utilizando modelos de espaço de estados;

Funcionalidades implementadas:

- Filtro de Kalman e suavizador;
- Filtro e suavizador *square-root*;
- Estimação via máxima verossimilhança;
- Previsão;
- Simulação Monte Carlo;
- Modelagem multivariada;
- Modelos padronizados (usuário define matrizes Z, T e R);
- Modelos pré-definidos: nível local, tendência linear, estrutural;
- Preenchimento automático de dados faltantes;
- Diagnósticos para os resíduos do modelo (testes de normalidade, independência e homoscedasticidade).

Framework de Espaço de Estados

Framework clássico da Engenharia de Controle;

Representa um sistema através da definição de variáveis de entrada, estado e saída;

- Variáveis de entrada: entidades externas que podem representar inputs de controle ou ruído;
- Variáveis de estado: componentes não-observáveis que evoluem ao longo do tempo de acordo com as equações de estado e as variáveis de entrada;
- Variáveis de saída: saída observável do sistema, resultado da realização do estado mais ruído.

Framework de Espaço de Estados

$$
\begin{aligned}
y_t &= Z_t \alpha_t + \varepsilon_t, & \varepsilon_t &\sim N(0, H_t) \\
\alpha_{t+1} &= T_t \alpha_t + R_t \eta_t, & \eta_t &\sim N(0, Q_t)
\end{aligned}
$$

Framework de Espaço de Estados

Equação de observação

$$
\begin{aligned}
y_t &= Z_t\alpha_t + \varepsilon_t, \qquad && \varepsilon_t \sim N(0,H_t) \\
\alpha_{t+1} &= T_t\alpha_t + R_t\eta_t, \qquad && \eta_t \sim N(0,Q_t)
\end{aligned}
$$

Equação de estado

Exemplo: Modelo de tendência linear

$$y_t = \mu_t + \varepsilon_t, \qquad \varepsilon_t \sim N(0, \sigma_\varepsilon^2),$$
$$\mu_{t+1} = \mu_t + \nu_t + \xi_t, \qquad \xi_t \sim N(0, \sigma_\xi^2),$$
$$\nu_{t+1} = \nu_t + \zeta_t, \qquad \zeta_t \sim N(0, \sigma_\zeta^2)$$

Escrevendo esse modelo na forma de espaço de estados:

$$y_t = [1 \quad 0]\, \alpha_t + \varepsilon_t, \qquad \varepsilon_t \sim N(0, \sigma_\varepsilon^2),$$
$$\alpha_{t+1} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \alpha_t + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \eta_t, \qquad \eta_t \sim N\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \sigma_\xi^2 & 0 \\ 0 & \sigma_\zeta^2 \end{bmatrix} \right)$$

Uso do pacote

O uso do pacote é realizado através dos seguintes passos:

1. Especificação do modelo
2. Estimação do modelo
    1. Filtragem
    2. Estimação
    3. Suavização
3. Previsão e simulação

Especificação do modelo

O primeiro passo é a definição de um objeto **StateSpaceModel** que contém as observações e as matrizes do sistema.

Podem ser utilizados modelos pré-definidos ou pode ser criado um modelo personalizado pelo usuário através da definição das matrizes.

```
model = local_level(y)

model = linear_trend(y)

model = structural(y, s; X = X)

model = StateSpaceModel(y, Z, T, R)
```

Estimação do modelo

O modelo é estimado através do Filtro de Kalman e da máxima verossimilhança.

Na estimação, as variâncias dos termos de erro, H e Q, são estimadas.

```
ss = statespace(model; verbose = 1)
```

```
julia> ss = statespace(model)
==============================================================
                 StateSpaceModels.jl v0.2.0
 (c) Raphael Saavedra, Guilherme Bodin, and Mario Souto, 2019
--------------------------------------------------------------
            Starting state-space model estimation.
    Initiating maximum likelihood estimation with 3 seeds.
--------------------------------------------------------------
               Seed 0 is aimed at degenerate cases.
--------------------------------------------------------------
||     seed     |      log-likelihood      |      time (s)     ||
||        0     |          -16217.4939     |          0.21     ||
||        1     |           -1350.4763     |          2.65     ||
||        2     |           -1350.4763     |          4.08     ||
||        3     |           -1350.4763     |          5.69     ||
--------------------------------------------------------------
           Maximum likelihood estimation complete.
                 Log-likelihood: -1350.4763
              End of state-space model estimation.
==============================================================
```

Estimação do modelo

Nessa etapa, o Filtro de Kalman realiza a estimação do **estado preditivo** e do **estado filtrado** a cada instante de tempo:

$$
\begin{aligned}
a_{t+1} &= \mathbb{E}[\alpha_{t+1}|Y_t] \\
P_{t+1} &= \mathbb{V}[\alpha_{t+1}|Y_t]
\end{aligned}
\qquad\qquad
\begin{aligned}
a_{t|t} &= \mathbb{E}[\alpha_t|Y_t] \\
P_{t|t} &= \mathbb{V}[\alpha_t|Y_t]
\end{aligned}
$$

Além disso, o filtro também calcula as **inovações**, ou erros de previsão, e sua matriz de covariância a cada instante de tempo:

$$
\begin{aligned}
v_t &= y_t - Z a_t, \\
F_t &= \mathbb{V}[v_t].
\end{aligned}
$$

No pacote, temos implementado o Filtro de Kalman, o filtro *square-root* e permitimos a definição de qualquer variante de filtro definida pelo usuário através do tipo abstrato `AbstractFilter`.

Estimação do modelo

Porém, para o filtro ser rodado, é necessário ter em mãos as covariâncias dos termos de erro, H e Q. Portanto, esses parâmetros devem ser estimados via máxima verossimilhança.

A função de log-verossimilhança é dada por:

$$\ell(Y_n) = -\frac{np}{2}\log 2\pi - \frac{1}{2}\sum_{t=1}^{n}(\log|F_t| + v_t^\top F_t^{-1} v_t)$$

A maximização da log-verossimilhança é realizada através de um método denominado `RandomSeedsLBFGS`, que utiliza parâmetros iniciais aleatórios e o algoritmo LBFGS. Porém, é possível implementar métodos personalizados através do tipo abstrato `AbstractOptimizationMethod`.

Previsão e simulação

Após a estimação do modelo, é possível prever os valores futuros das variáveis observadas através da função `forecast`:

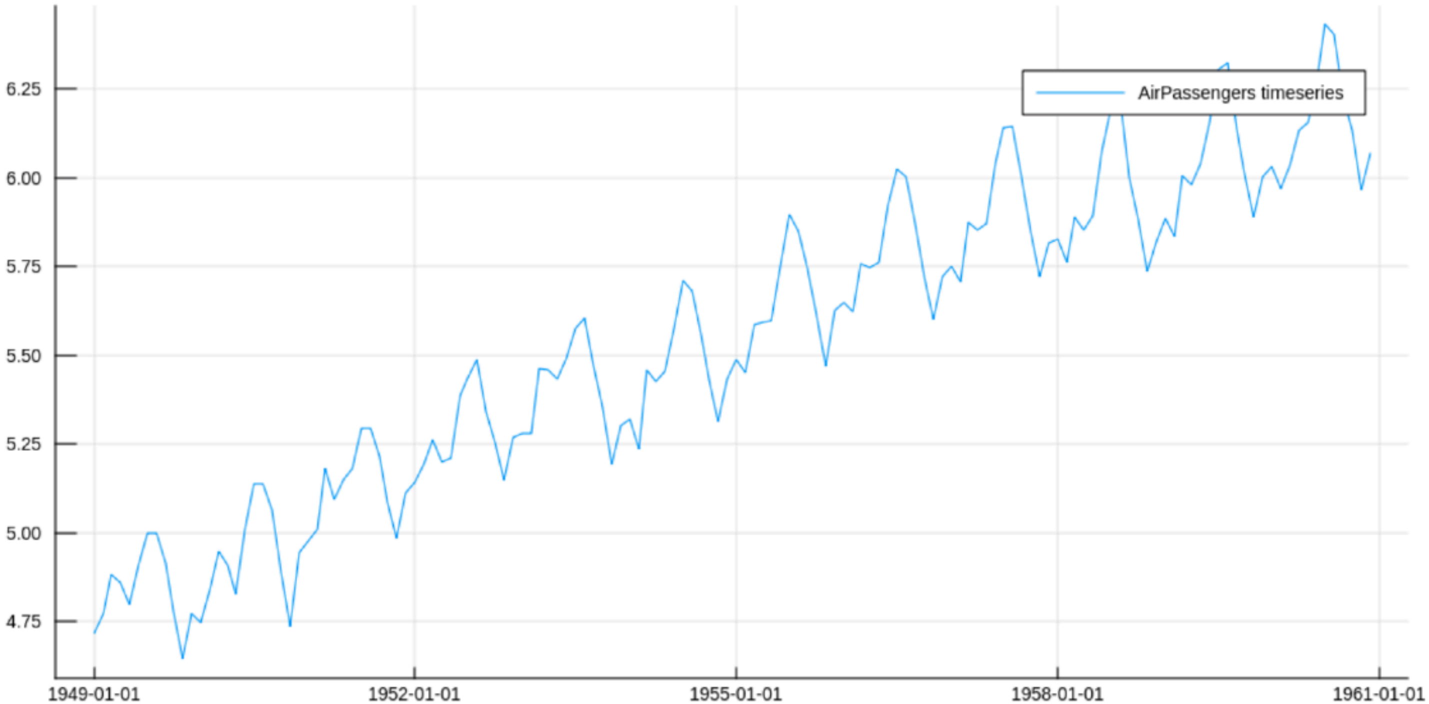
```
pred, dist = forecast(ss, N)
```

Alternativamente, outra ferramenta poderosa é a simulação de cenários futuros utilizando Monte Carlo, que é realizada através da função `simulate`:

```
scenarios = simulate(ss, N, S)
```

Aplicações

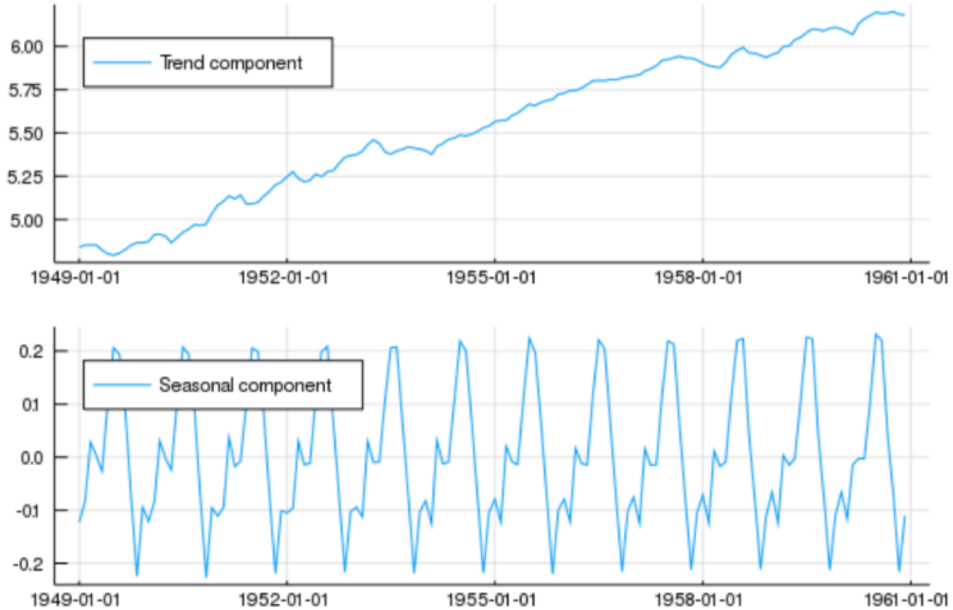
```julia
using CSV, StateSpaceModels, Plots, Statistics, Dates

# Load the AirPassengers dataset
AP = CSV.read("AirPassengers.csv")

# Take the log of the series
logAP = log.(Array{Float64}(AP[:Passengers]))

p1 = plot(AP[:Date], logAP, label = "AirPassengers timeseries", size = (1000, 500))
```



Aplicações: extração de componentes

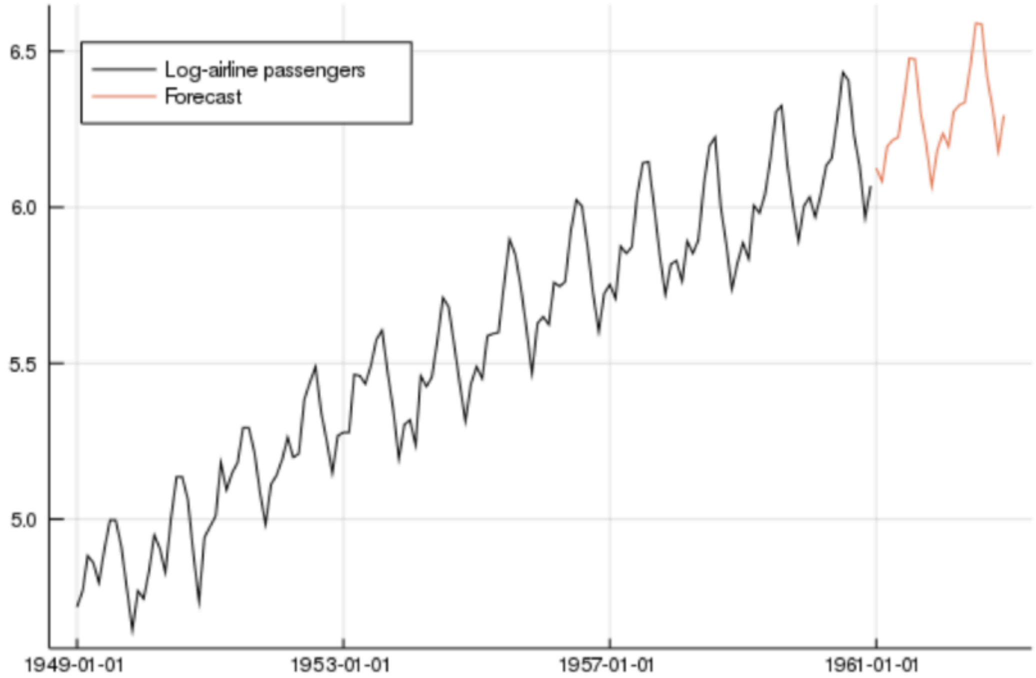
```
# Create structural model with seasonality of 12 months
model = structural(logAP, 12)
```

```
# Estimate a StateSpace structure
ss = statespace(model)

# Analyze its decomposition in trend and seasonal
p2 = plot(AP[:Date], [ss.smoother.alpha[:, 1] ss.smoother.alpha[:, 3]], layout = (2, 1),
          label = ["Trend component" "Seasonal component"], legend = :topleft)
```



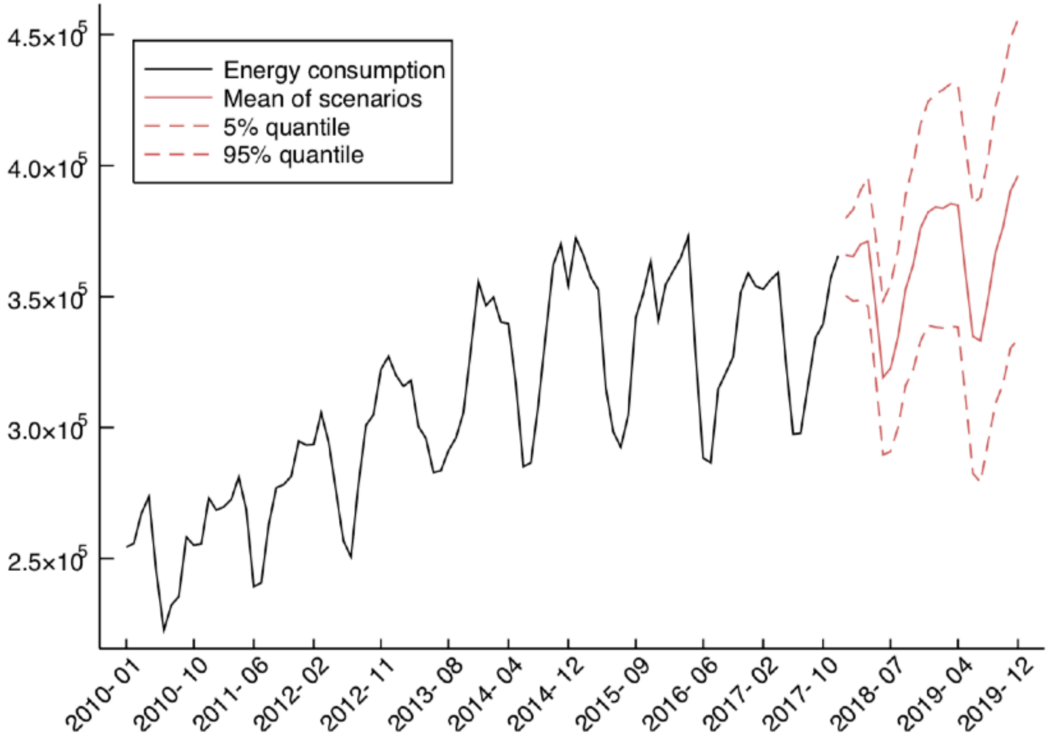
Aplicações: previsão

```
# Forecast 24 months ahead
N = 24
pred, dist = forecast(ss, N)

# Define forecasting dates
firstdate = AP[:Date][end] + Month(1)
newdates = collect(firstdate:Month(1):firstdate + Month(N - 1))

p3 = plot!(p1, newdates, pred, label = "Forecast")
```



Aplicações: simulação

```
# Specify the state-space model
model = structural(consumption, 12; X = temperature)
# Estimate the state-space model
ss = statespace(model; filter_type = SquareRootFilter)
# Number of months ahead to be simulated
N = 24
# Number of scenarios to be simulated
S = 1000
# Perform simulation
sim = simulate(ss, N, S)
```



Aplicações: rastreamento de veículos

Vamos supor agora que y_t é um vetor 2x1 de observações ruidosas da posição de um veículo sobre um plano de duas dimensões no instante t.

Definimos algumas equações dinâmicas básicas:

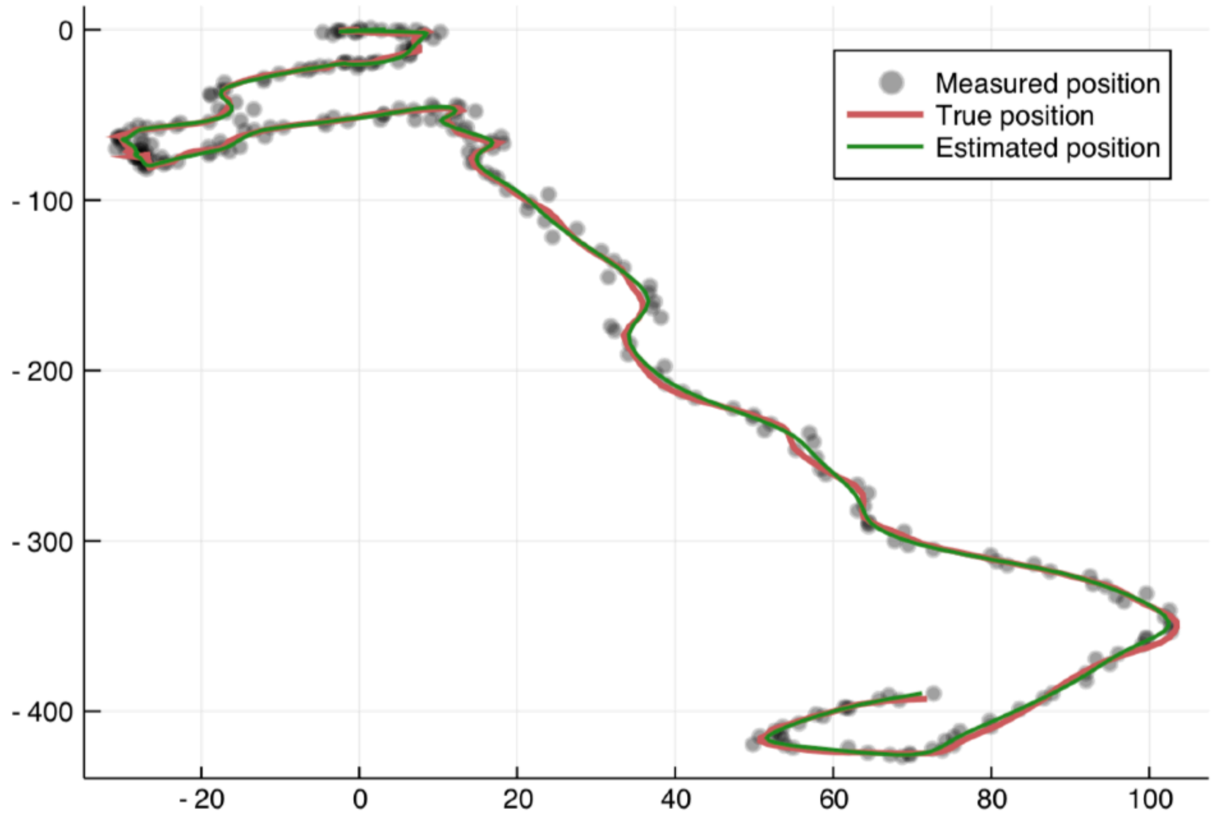
$$x_{t+1}^{(d)} = x_t^{(d)} + \left(1 - \frac{\rho\Delta_t}{2}\right)\Delta_t \dot{x}_t^{(d)} + \frac{\Delta_t^2}{2}\eta_t^{(d)},$$
$$\dot{x}_{t+1}^{(d)} = (1-\rho)\dot{x}_t^{(d)} + \Delta_t \eta_t^{(d)}$$

Isso pode ser colocado na forma de espaço de estados da seguinte forma:

$$y_t = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \alpha_{t+1} + \varepsilon_t,$$

$$\alpha_{t+1} = \begin{bmatrix} 1 & (1-\frac{\rho\Delta_t}{2})\Delta_t & 0 & 0 \\ 0 & (1-\rho) & 0 & 0 \\ 0 & 0 & 1 & (1-\frac{\rho\Delta_t}{2}) \\ 0 & 0 & 0 & (1-\rho) \end{bmatrix} \alpha_t + \begin{bmatrix} \frac{\Delta_t^2}{2} & 0 \\ \Delta_t & 0 \\ 0 & \frac{\Delta_t^2}{2} \\ 0 & \Delta_t \end{bmatrix} \eta_t$$

Aplicações: rastreamento de veículos



Obrigado!

Raphael Saavedra — raphael.saavedra93@gmail.com