Estimação de parâmetros

Volatilidade Estocástica

Sumário

1 Introdução

```
library(stochvol)
```

O pacote `stochvol` conta com 4 funções principais para estimarmos os parâmetros do modelo de volatilidade estocástica com uma abordagem bayesiana.

As 4 funções utilizam do algoritmo MCMC para funcionar, e na prática, são apenas variações convenientes umas das outras. São elas: `svsample`, `svtsample`, `svlsample` e `svtlsample`. Onde, a 3 últimas são equivalentes a primeira com algumas alterações (erros t, leverage e erros t + leverage, respectivamente).

1.1 `svsample`

1.1.1 Argumentos

A função `svsample` conta com os seguintes argumentos:

- `y`: dados.
- `draws`: tamanho das cadeias.
- `burnin`: números de draws a serem desconsiderados.
- `designmatrix`: matrix de design para modelar a média.
- `prior+`: onde + é o parâmetro cuja priori queremos especificar (pode ser mu, sigma, nu, rho, beta).
- `priorlatente0`: ou "Stationary" para usar como piori para h_0 a distribuição estacionária do processo latente AR(1), ou `numeric` B para considerar como priori $h_0 \sim N(\mu, B\sigma^2)$.
- `priorspec`: para utilizar outras prioris além das padrões.
- `n_chains`: número de cadeias independentes.

Além de alguns outros argumentos opcionais (como argumentos para controlar o número de CPU's para computar as cadeias).

1.1.2 Valores

Os resultados da função são:

- `para`: parâmetros
- `latent`: log-volatilidade latente instantânea
- `latent0`: log-volatilidade latente inicial
- `tau`: fator de inflação da variância latente
- `beta`: coeficientes da regressão (opcional)
- `y`: dados
- `runtime`: run time
- `priors`: parâmetros das prioris consideradas para os parâmetros dos modelos
- `summary`: estatísticas sumárias a posteriori
- `meanmodel`: informações sobre a matriz de design

1.2 Pacote vs Artigo

O modelo de volatilidade estocástica que o pacote utiliza é descrito de outra forma do que no artigo "*A note on stochastic volatility model estimation*". No artigo, os únicos parâmetros a serem estimados são β e ϕ, já no pacote os parâmetros são μ e ϕ.

Os modelos são definidos da seguinte forma:

Artigo,

$$\begin{aligned} r_t &= \beta e^{h_t/2}\epsilon_t \\ h_t &= \phi h_{t-1} + \omega_t \end{aligned}$$

Pacote,

$$\begin{aligned} r_t &= e^{h_t/2}\epsilon_t \\ h_t &= \mu + \phi(h_{t-1} - \mu) + \omega_t \end{aligned}$$

Podemos escrever β em função de μ (*level* do processo AR(1)) e ϕ (*persistence* do processo AR(1)), da seguinte forma:

$$\beta = e^{\mu(1-\phi)/2} \implies \mu = \frac{2\ln\beta}{1-\phi}$$

2 Aplicação

2.1 Simulação

Vamos simular dados de um modelo de volatilidade estocástica através da função pronta do modelo e da função que eu criei, para comparar os resultados.

Vou simular 4 modelos com as seguintes especificações:

- Modelo 1: $\mu = -10$, $\phi = 0.99$, $sigma = 0.2$
- Modelo 2: $\mu = -15$, $\phi = 0.98$, $sigma = 0.6$
- Modelo 3: $\mu = -10$, $\phi = 0.98$, $sigma = 0.5$
- Modelo 4: $\mu = -15$, $\phi = 0.99$, $sigma = 0.6$

obs: os parâmetros $\nu = \infty$ (graus de liberdade correspondendo uma normal padrão), e $rho = 0$ correspondem ao modelo SV básico.

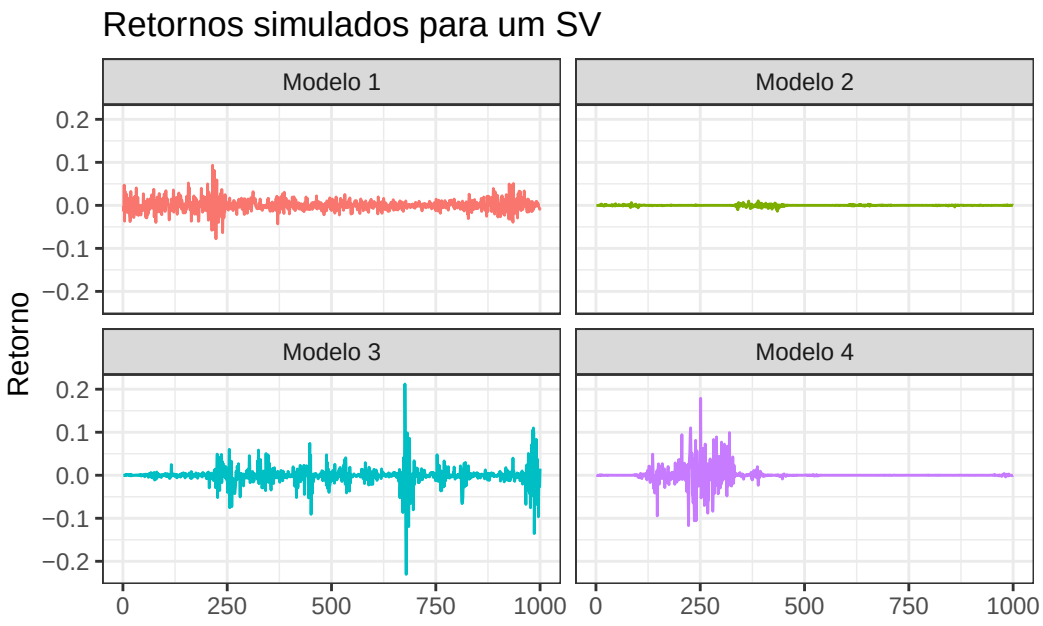
obs2: A função que transforma β em μ e vice-versa é:

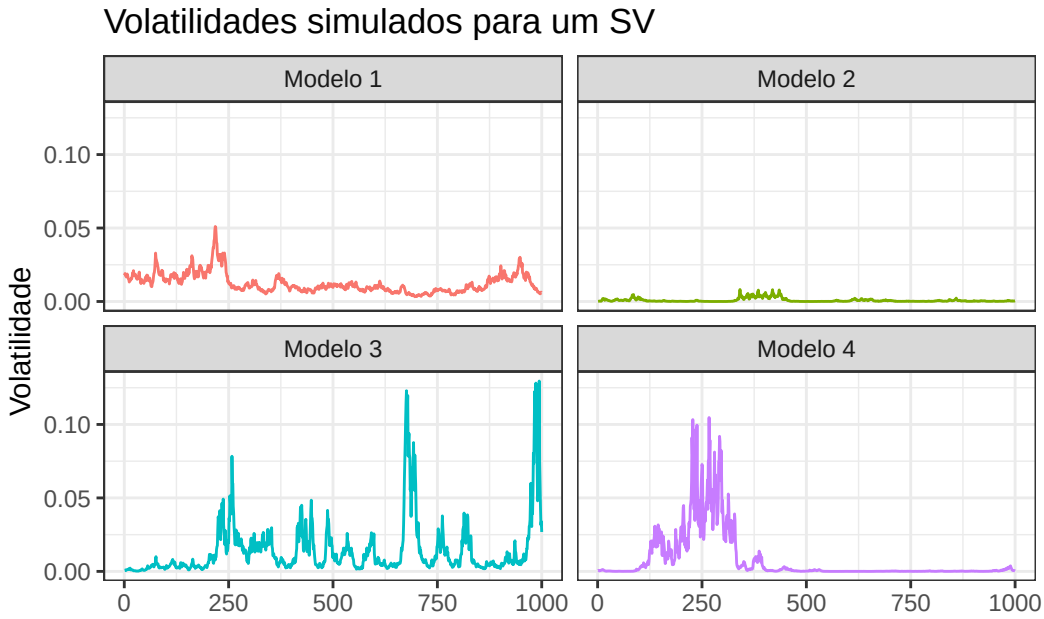
```
beta_mu <- function(x, phi, to_mu=TRUE) {
  # Caso to_mu=TRUE a função transforma um valor beta em mu
  # Caso FALSE a função transforma um valor mu em beta
  if (to_mu) {
    mu <- (2*log(x)) / (1 - phi)
    return(mu)
  }

  beta <- exp((x * (1 - phi))/2)
  return(beta)
}
```

A simulação das séries se encontra abaixo:

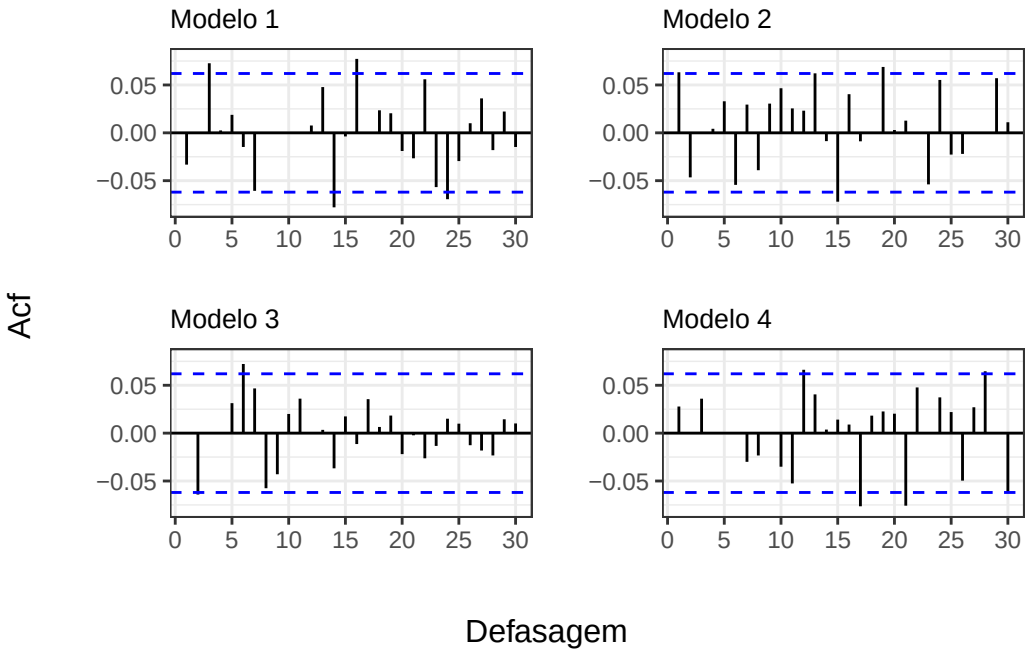
```
set.seed(236106)

amostra1 <- svsim(1000, mu = -10, phi = .99, sigma =  0.2)

amostra2 <- svsim(1000, mu = -15, phi = .98, sigma =  0.6)

amostra3 <- svsim(1000, mu = -10, phi = .98, sigma =  0.5)

amostra4 <- svsim(1000, mu = -15, phi = .99, sigma =  0.6)
```

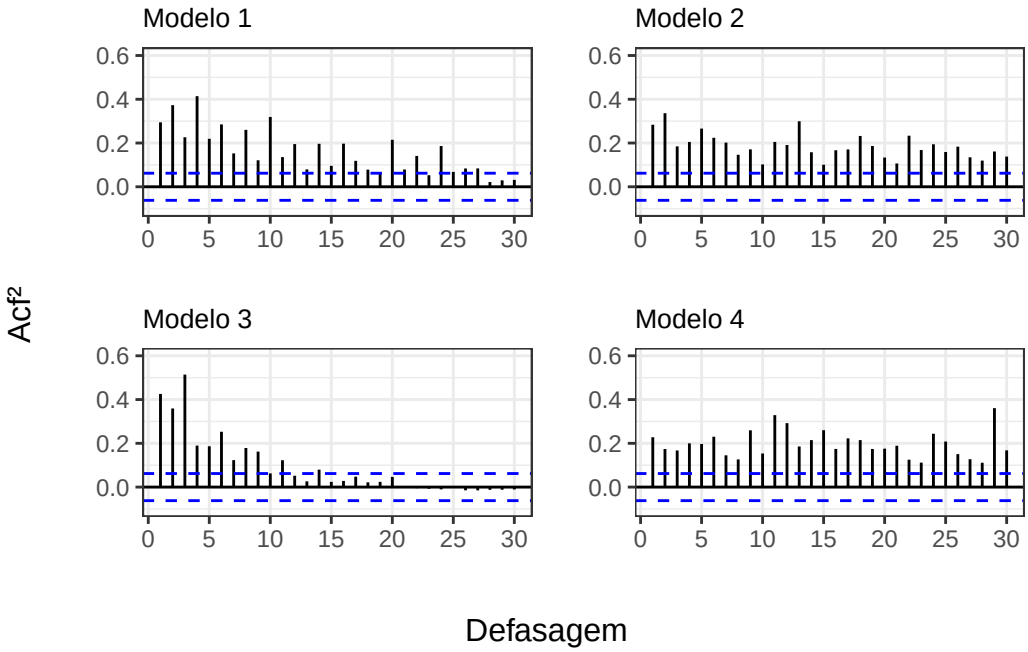
Os gráficos dos valores simulados e suas volatilidades estão apresentados abaixo:





As funções de autocorrelações e seus quadrados são mostradas abaixo:





2.2 Estimação

Para fazer estimação, irei considerar as prioris padrões.

- $\mu \sim N(0, 100)$ (priori não-informativa)
- $\frac{\phi+1}{2} \sim Beta(5, 1.5)$
- $\sigma^2 \sim \sigma_0 \cdot \chi^2_1$ (onde σ_0 é o sigma a priori, por *default* $= 1$).

E estimei com os argumentos default:

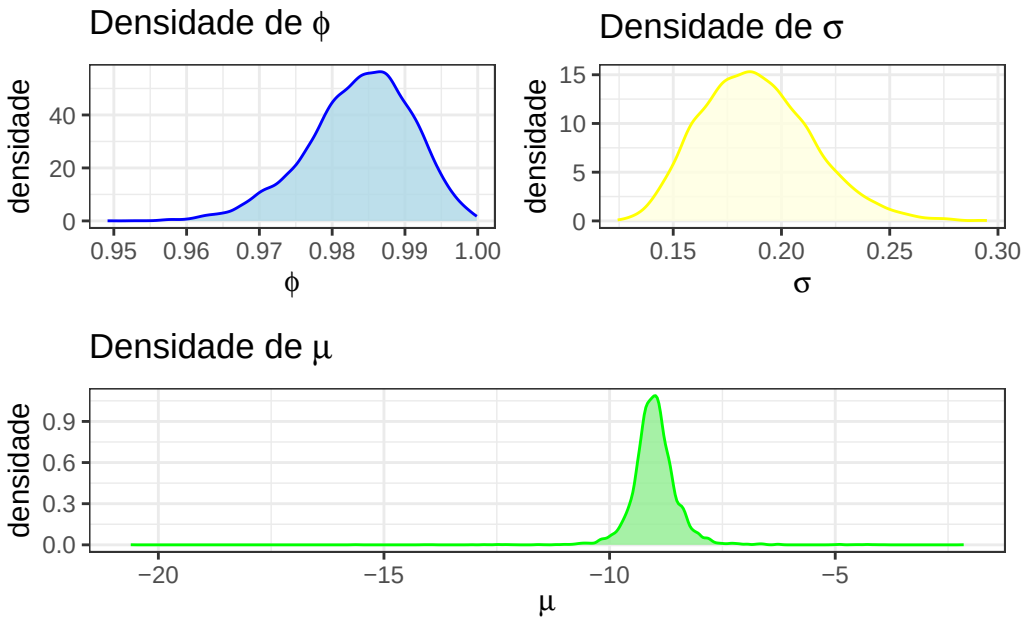
```
# Estimações dos parâmetros
draws1 <- svsample(amostra1$y, draws = 10000)
draws2 <- svsample(amostra2$y, draws = 10000)
draws3 <- svsample(amostra3$y, draws = 10000)
draws4 <- svsample(amostra4$y, draws = 10000)
```

Os coeficientes estimados para o primeiro modelo foram:

```
draws1$summary$para[,1]
```

```
        mu         phi       sigma    exp(mu/2)     sigma^2
-9.00075279  0.98377485  0.18850594  0.01160447  0.03618145
```

Podemos, também, graficar as densidades estimadas para cada parâmetro de interesse:



Os resultados obtidos estão apresentados na tabela abaixo, onde podemos compará-los com os verdadeiros valores dos parâmetros.

Table 1: Valores estimados para cada modelo.

Modelos	μ	ϕ	σ
1	-9.0008	0.9838	0.1885
2	-16.1648	0.9778	0.6266
3	-9.7617	0.9652	0.6012
4	-15.2011	0.9930	0.5808

Table 2: Valores verdadeiros dos parâmetros, por modelo.

Modelos	μ	ϕ	σ
1	-10	0.99	0.2
2	-15	0.98	0.6
3	-10	0.98	0.5
4	-15	0.99	0.6

Podemos ainda melhorar a estimação colocando algumas informações a priori. Como feito no artigo do professor Maurício, podemos considerar como valor inicial para β a média incondicional da série simulada, e comparar a estimação com aquela feita sem essa informação prévia.

O único problema que encontrei é que, para transformarmos de β para μ, precisamos do valor de ϕ. Talvez uma alternativa seja estimar o valor de ϕ sem considerar nenhuma priori específica, e depois usar essa estimativa para encontrar μ, dado β.

2.3 Predição

A predição das observações é feita através da função `predict`. Caso estejamos interessados em fazer um *Rolling windows*, podemos utilizar a função `svlsample_roll`.

2.3.1 Exemplo

- Fazer a previsão do modelo 1 para 3 passos a frente:

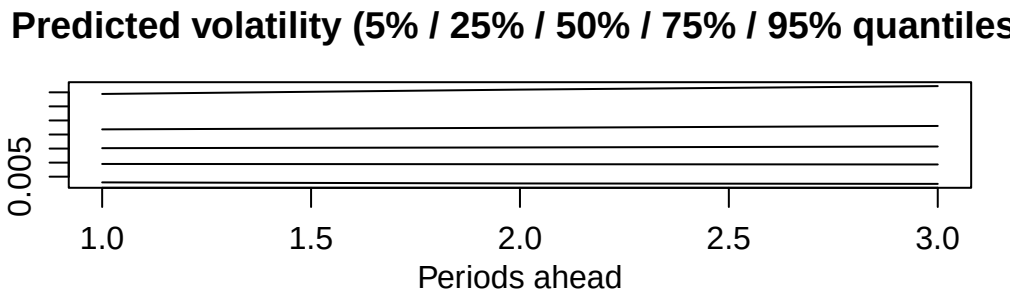
```
fore <- predict(draws1, 3)
summary(predlatent(fore))
```

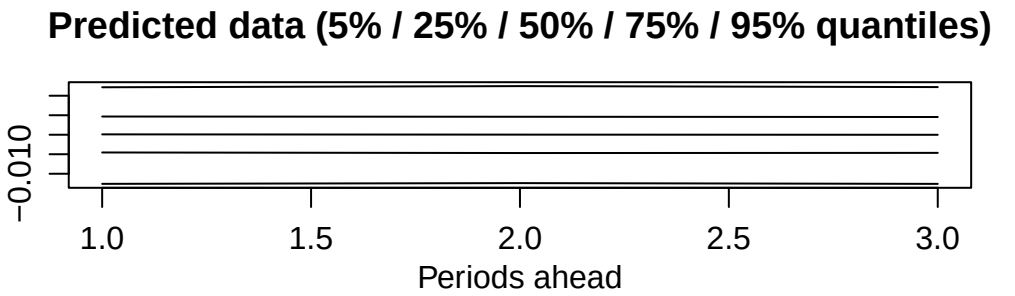
```
Iterations = 1:10000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 10000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

         Mean     SD Naive SE Time-series SE
h_1001 -9.912 0.5244 0.005244        0.01089
h_1002 -9.900 0.5474 0.005474        0.01039
h_1003 -9.886 0.5716 0.005716        0.01028

2. Quantiles for each variable:

         2.5%    25%    50%    75%  97.5%
h_1001 -10.91 -10.26 -9.916 -9.567 -8.863
h_1002 -10.96 -10.27 -9.901 -9.541 -8.797
h_1003 -11.01 -10.28 -9.883 -9.510 -8.764
```

```
summary(predy(fore))
```

```
Iterations = 1:10000
Thinning interval = 1
Number of chains = 1
Sample size per chain = 10000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

             Mean       SD  Naive SE Time-series SE
y_1001 -4.831e-06 0.007586 7.586e-05      7.586e-05
y_1002 -1.140e-05 0.007645 7.645e-05      7.645e-05
y_1003 -6.915e-05 0.007664 7.664e-05      7.664e-05

2. Quantiles for each variable:

           2.5%       25%        50%      75%   97.5%
y_1001 -0.01537 -0.004548  9.267e-05 0.004668 0.01506
y_1002 -0.01545 -0.004693  2.745e-05 0.004605 0.01548
y_1003 -0.01582 -0.004647 -1.162e-05 0.004546 0.01526
```

```
plot(fore)
```






```
plot(draws1, forecast = fore)
```