拡散過程モデルのパラメータ推定法について

上原 悠槙
統計数理研究所

平成 30 年 11 月 20 日

概 要

本資料では, 高頻度離散観測に基づく拡散過程モデルのパラメータ推定法について解説する. まず導入として, 代表的なパラメータ推定法である最尤推定法及びベイズ推定法について概説を行う. その後, 拡散過程モデルにおけるパラメータ推定法について述べ, YUIMA パッケージ上での実践について説明する.

最尤推定法

確率変数 $X_1, X_2, \ldots, X_n$ が p 次元パラメータ $\theta \in \Theta$ により特徴付けられた同時密度関数 $p(X_1, X_2, \ldots, X_n, \theta)$ を持つとする. ここで Θ はパラメータ空間を表し, パラメータの真の値 θ_0 は Θ 内に存在すると仮定する. 実現値 $X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n$ が得られた時, パラメータ θ の関数

$$L_n(\theta) = p(x_1, x_2, \ldots, x_n, \theta)$$

を**尤度関数**と呼ぶ. すなわち尤度関数は, パラメータ θ ごとに現在起きている事象の確率を与える関数である. **最尤推定法**とは尤度関数 $L_n(\theta)$ を最大化する $\hat{\theta}_n := \hat{\theta}(x_1, x_2, \ldots, x_n) \in \Theta$ をパラメータ θ の推定値とする手法である. 対数関数の単調性から, 最尤推定量 $\hat{\theta}_n$ は対数尤度関数 $\log L_n(\theta)$ を最大化する $\tilde{\theta}_n \in \Theta$ と等しい.

簡単のため, 確率変数 $X_1, X_2, \ldots, X_n$ が独立同分布の場合を考える. この時, 対数尤度関数 $\log L_n(\theta)$ は X_1 の密度関数 $p(x, \theta)$ を用いて

$$\log L_n(\theta) = \sum_{j=1}^{n} \log p(x_j, \theta) \tag{1}$$

と書きかえることができる. ここで大数の法則により, 確率 1 で収束

$$\frac{1}{n}\left(\log L_n(\theta_0) - \log L_n(\theta)\right) \to \int \log\left[\frac{p(x, \theta_0)}{p(x, \theta)}\right] p(x, \theta_0) dx$$

が成り立つ. 右辺は, Kullback-Leibler 情報量と呼ばれる分布間の近さを測る尺度であり, これを $\mathrm{KL}(p(x, \theta_0); p(x, \theta))$ と書く. この時, 不等式

$$\mathrm{KL}(p(x, \theta_0); p(x, \theta)) \geq 0$$

が成り立つことが知られており, その等号成立条件はほとんどいたるところ $p(x, \theta_0) = p(x, \theta)$ である. 識別性条件 「$\theta_0 \neq \theta \Rightarrow p(x, \theta_0) \neq p(x, \theta)$」を仮定すると上の条件から, 真値 θ_0 が $\mathrm{KL}(p(x, \theta_0); p(x, \theta))$ を最小化する唯一の点であることがわかる. これにより, 最尤推定量 $\hat{\theta}_n$ が 真値 θ_0 へ収束することが予想できる. この予想は実際に正

しく, 適切な条件の下で $\hat{\theta}_n$ は θ_0 へ確率収束する (最尤推定量の一致性と呼ばれる). 加えて, 誤差ベクトルをスケーリングした $\sqrt{n}(\hat{\theta}_n - \theta_0)$ の漸近正規性, すなわち,

$$\sqrt{n}(\hat{\theta}_n - \theta_0) \xrightarrow{\mathcal{L}} N(0, I(\theta_0)^{-1}),$$

も知られている[1]. ここで $I(\theta_0) \in \mathbb{R}^p \otimes \mathbb{R}^p$ はフィッシャー情報行列

$$I(\theta_0) = \int (\partial_\theta \log p)^\top (\partial_\theta \log p)(x, \theta_0) p(x, \theta_0) dx$$

である ($\mathbb{R}^p \otimes \mathbb{R}^p$ は $p \times p$ の成分が実数値の行列全体を表し, $\top$ は行列 (ベクトル) の転置を表す). この漸近正規性により, パラメータの信頼区間や統計的仮説検定を構成することができる.

上では, 観測の独立同分布性の仮定の下で議論を行ったが, 確率微分方程式からの離散観測を想定するとき, その仮定は不適合であるため (1) のような対数尤度関数の書き換えはできない. しかし, $X_1, X_2, \ldots, X_n$ がマルコフ性, すなわち, 任意の $j \in \{1, \ldots, n-1\}$ に対して, 条件付き確率に関する関係式

$$P(X_{j+1} = x_{j+1} | X_1 = x_1, X_2 = x_2, \ldots, X_j = x_j) = P(X_{j+1} = x_{j+1} | X_j = x_j)$$

を満たす時, 対数尤度関数 $\log L_n(\theta)$ は推移確率 $p(x_j | x_{j-1}, \theta)$ を用いて

$$\log L_n(\theta) = \log p(x_1, \theta) + \sum_{j=1}^{n} p(x_j | x_{j-1}, \theta) \tag{2}$$

と書きかえることができる. 拡散過程は適当な正則条件の下, Markov 性を持つことが知られているため, 後にこの分解を元に議論を行う.

ベイズ推定

前節と同様に, 確率変数 $X_1, X_2, \ldots, X_n$ からの 実現値 $X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n$ が得られているとする. 最尤推定法では, パラメータ θ は非確率的な量として捉えていたが, ベイズ推定法ではパラメータをある分布に従う確率変数とみなす. これにより, 解析に先立ってパラメータの事前知識を推定に取り入れることができ, それに基づくパラメータの事前分布を π と書く (パラメータ空間 Θ 上の分布). また, 実現値 $X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n$ に対する尤度関数をここでは, $p(x_1, x_2, \ldots, x_n | \theta)$ と書く. ベイズ推定においては, θ の事後分布 $p(\theta | x_1, x_2, \ldots, x_n)$ が重要な役割を果たす. その定義から, 事後分布は観測が与えられた際の θ の分布であることがわかる. ここで, ベイズの定理を用いることで事後分布は,

$$\begin{aligned} &p(\theta | x_1, x_2, \ldots, x_n) \\ &= \left[\int_\Theta p(x_1, x_2, \ldots, x_n | \theta) \pi(\theta) d\theta \right]^{-1} p(x_1, x_2, \ldots, x_n | \theta) \pi(\theta), \end{aligned} \tag{3}$$

と書き表すことができ, 右辺は観測と事前分布のみで表現されている. この事後分布による平均 (**事後平均**), すなわち

$$\begin{aligned} &\int_\Theta \theta p(\theta | x_1, x_2, \ldots, x_n) d\theta \\ &= \left[\int_\Theta p(x_1, x_2, \ldots, x_n | \theta) \pi(\theta) d\theta \right]^{-1} \int_\Theta \theta p(x_1, x_2, \ldots, x_n | \theta) \pi(\theta) d\theta, \end{aligned}$$

がしばしばパラメータ θ の推定値として用いられる. 事後平均も最尤推定量と同様に, 適切な条件の下で一致性や漸近正規性を満たすことが知られている.

---

[1] $\xrightarrow{\mathcal{L}}$ は分布収束を意味する. ここで, d-次元確率変数列 (X_n) と X について, (X_n) が X に分布収束するとは, $P(X_n \leq x) \to F(x) := P(X \leq x)$ が $F(x)$ の任意の連続点 $x \in \mathbb{R}^d$ について成り立つことを意味する.

拡散過程のパラメータ推定法

以下では, d-次元拡散過程モデル

$$dX(t) = a(X(t), \alpha)dt + b(X(t), \beta)dB(t), \quad X_0 = x_0, \tag{4}$$

の離散観測 $\mathbb{X}_n = (X(t_j^n))_{j=1}^n$ に基づくパラメータ $\theta := (\alpha, \beta) \in \Theta_\alpha \times \Theta_\beta =: \Theta$ の推定法を概説する. ここで, 真の値 θ_0 は Θ に含まれているものとする. また, $t_j^n = j\Delta_n$ であり, 正数列 (Δ_n) は $T_n := n\Delta_n \to \infty$ および $n\Delta_n^2 \to 0$ を満たすものとする. すなわち, 観測時間 t_j^n も n に依存した形を取り, 各観測時間幅 Δ_n が n の増加に伴い, 減少する高頻度観測の状況を考えている. 以下では記法の簡便性のため, t_j と略記する. また, Θ_α, Θ_β はそれぞれドリフトパラメータと拡散パラメータのパラメータ空間を表す. 加えて, ドリフト係数 $a : \mathbb{R}^d \times \Theta_\alpha \mapsto \mathbb{R}^d$, 拡散係数 $b : \mathbb{R}^d \times \Theta_\beta \mapsto \mathbb{R}^d \otimes \mathbb{R}^d$ である.

(4) に関して, ドリフト係数, 拡散係数が適当な正則条件を満たしていれば確率微分方程式の解が一意に存在し, Markov 性を持つことが知られている. ゆえに, (2) から対数尤度関数 $\log L_n(\theta) = \log p(X(t_0), X(t_1), X(t_2), \ldots, X(t_n), \theta)$ は

$$\log L_n(\theta) = \log p(X(t_0), \theta) + \sum_{j=1}^n \log p(X(t_j)|X(t_{j-1}), \theta)$$

と書き直すことができる. しかしながら, 多くの場合, 上式右辺に現れる推移確率の陽な形は得られないため, 尤度解析が困難となる. そのため, 以下では尤度関数の代替となる**疑似尤度関数**の構成について述べていく.

$\Delta_j X = X(t_j) - X(t_{j-1})$, $S(x, \beta) = b(x, \beta)b^\top(x, \beta)$, $\phi(x; \mu, \Sigma)$ を平均 μ, 分散 Σ の正規分布の密度関数とする. また, 行列 A, B について, $A^{\otimes 2} = AA^\top$, $B[A] = \mathrm{Trace}(BA^\top)$ と書く. 解過程 $X(t)$ のシミュレーションで用いたオイラー・丸山法に基づき, 観測系列 $\mathbb{X}_n$ の離散近似

$$X(t_j) \approx X(t_{j-1}) + a(X(t_{j-1}), \alpha)\Delta_n + b(X(t_{j-1}), \beta)(B(t_j) - B(t_{j-1})),$$

を考えると, ブラウン運動の定義から $B(t_j) - B(t_{j-1})$ は平均 0, 分散 Δ_n の正規分布に従うため, 推移確率の近似

$$p(X(t_j)|X(t_{j-1}), \theta) \approx \phi(X(t_j); X(t_{j-1}) + a(X(t_{j-1}), \alpha)\Delta_n, S(X(t_{j-1}), \beta)\Delta_n)$$

を得ることができる. この局所正規近似により, 対数疑似尤度関数 $\mathbb{H}_n(\theta)$ を

$$\begin{aligned}
\mathbb{H}_n(\theta) &= \sum_{j=1}^n \log \phi(X(t_j); X(t_{j-1}) + a(X(t_{j-1}), \alpha)\Delta_n, S(X(t_{j-1}), \beta)\Delta_n) \\
&= \sum_{j=1}^n \left\{ -\frac{d}{2}\log(2\pi\Delta_n) - \frac{1}{2}\log|S(X(t_{j-1}), \beta)| \right. \\
&\qquad \left. -\frac{1}{2\Delta_n} S(X(t_{j-1}), \beta)^{-1}[(\Delta_j X - a(X(t_{j-1}), \alpha)\Delta_n)^{\otimes 2}] \right\}
\end{aligned}$$

により定義する. これに基づき, $\mathbb{H}_n(\theta)$ を最大化する $\hat{\theta}_n := (\hat{\alpha}_n, \hat{\beta}_n) \in \Theta$ を**疑似最尤推定量**と定義する. また, **適応型疑似ベイズ推定量** $\bar{\theta}_n := (\bar{\alpha}_n, \bar{\beta}_n)$ を [3] に基づき以下の要領で定義する:

1. まず, 任意の初期値 $\alpha^\star \in \Theta_\alpha$ を固定し, β の疑似ベイズ推定量 $\bar{\beta}_n$ を

$$\bar{\beta}_n = \left[\int_{\Theta_\beta} \exp\{\mathbb{H}(\alpha^\star, \beta)\} \pi_1(\beta) d\beta \right]^{-1} \int_{\Theta_\beta} \beta \exp\{\mathbb{H}(\alpha^\star, \beta)\} \pi_1(\beta) d\beta,$$

により定義する. ここで, π_1 は Θ_β 上の事前分布を表す.

2. 次に $\bar{\beta}_n$ を用いて, α の疑似ベイズ推定量 $\bar{\alpha}_n$ を

$$\bar{\alpha}_n = \left[\int_{\Theta_\alpha} \exp\left\{\mathbb{H}(\alpha, \bar{\beta}_n)\right\} \pi_2(\alpha) d\alpha\right]^{-1} \int_{\Theta_\alpha} \alpha \exp\left\{\mathbb{H}(\alpha, \bar{\beta}_n)\right\} \pi_2(\alpha) d\alpha,$$

により定義する. ここで, π_2 は Θ_α 上の事前分布を表す.

以上の要領で定義された疑似最尤推定量 $\hat{\theta}_n$ と適応型疑似ベイズ推定量 $\bar{\theta}_n$ は, 解過程 X がエルゴード性を持ち, 観測が充分高頻度である場合に, 通常の最尤推定量やベイズ推定量と同様, 一致性および漸近正規性を満たすことが知られている. すなわち, $\hat{\theta}_n$ および $\bar{\theta}_n$ は真の値 $\theta_0 := (\alpha_0, \beta_0)$ へ確率収束し, さらに,

$$\begin{pmatrix} \sqrt{T_n} & O \\ O & \sqrt{n} \end{pmatrix} \begin{pmatrix} \hat{\alpha}_n - \alpha_0 \\ \hat{\beta}_n - \beta_0 \end{pmatrix} \xrightarrow{\mathcal{L}} N(0, I(\theta_0)^{-1}),$$

$$\begin{pmatrix} \sqrt{T_n} & O \\ O & \sqrt{n} \end{pmatrix} \begin{pmatrix} \bar{\alpha}_n - \alpha_0 \\ \bar{\beta}_n - \beta_0 \end{pmatrix} \xrightarrow{\mathcal{L}} N(0, I(\theta_0)^{-1}),$$

は成り立つ ([1], [2], [3]). ここで, 解過程 X の不変分布を ν とすると, 漸近分散 $I(\theta_0) = \mathrm{diag}[\Gamma_1(\theta_0), \Gamma_2(\theta_0)]$ は,

$$\Gamma_1(\theta_0)[u_1^{\otimes 2}] = \int_{\mathbb{R}^d} S(x, \beta_0)^{-1}[\partial_\alpha a(x, \alpha_0)[u_1], \partial_\alpha a(x, \alpha_0)[u_1]] \nu(dx)$$
$$\Gamma_2(\theta_0)[u_2^{\otimes 2}] = \frac{1}{2} \int_{\mathbb{R}^d} \mathrm{Trace}\left\{S^{-1}(\partial_\beta S) S^{-1}(\partial_\beta S)(x, \beta_0)[u_1^{\otimes}]\right\} \nu(dx)$$

により定義される.

YUIMA 上での実践

前節で定義した疑似最尤推定量および疑似ベイズ推定量の計算は, YUIMA 上に実装されている関数 `qmle` および `adaBayes` により容易に計算することができる. まず 2 つの関数の仕様について述べる.

qmle

関数 `qmle` は様々な確率微分方程式モデルの疑似最尤推定量を計算する関数である. これは以下の形式で定義されている.

```
qmle(yuima, start, lower, upper, joint = FALSE, rcpp =FALSE, ...)
```

ここで各引数はそれぞれ,

- `yuima`: 係数の情報やデータを包含した yuima オブジェクト.
- `start`: 推定量計算のための初期値. 構造は名前付きリスト.
- `lower`: それぞれのパラメータ空間の下限. 構造は名前付きリスト.
- `upper`: それぞれのパラメータ空間の上限. 構造は名前付きリスト.
- `joint`: 推定を段階的に行うか否か. TRUE または FALSE で入力し, FALSE の方が計算速度が速い (デフォルトは FALSE).
- `rcpp`: C++ のコードを使用するか否か. TRUE または FALSE で入力し, TRUE の方が計算速度が速い (デフォルトは FALSE).

である. また, その返り値は yuima.qmle というクラスで与えられる. このクラスは, 元の yuima オブジェクトの係数の情報や, `qmle` を使用した際のオプションなどを含んでいるが, 特に指定がなければ

- 疑似最尤推定量の値および標準誤差.
- 疑似最尤推定量をプラグインした -2 倍の対数疑似尤度関数の値.

の 2 つが出力される. クラス内の情報の抽出法は, 後に説明する.

adaBayes

関数 `adaBayes` は拡散過程モデルの適応型ベイズ推定量を計算する関数である. これは以下の形式で定義されている.

```
adaBayes(yuima, start, prior, lower, upper, rcpp = TRUE,
         method = "mcmc", mcmc = 1000,...)
```

ここで各引数はそれぞれ,

- `yuima`: 係数の情報やデータを包含した yuima オブジェクト.
- `start`: 推定量計算のための初期値. 構造は名前付きリスト.
- `prior`: パラメータの事前分布. 構造は名前付きリスト.
- `lower`: それぞれのパラメータ空間の下限. 構造は名前付きリスト.
- `upper`: それぞれのパラメータ空間の上限. 構造は名前付きリスト.
- `rcpp`: C++ のコードを使用するか否か. TRUE または FALSE で入力し, TRUE の方が計算速度が速い (デフォルトは FALSE).
- `method`: 積分計算で何を用いるか. `mcmc`: または `nomcmc` で入力し, デフォルトは `mcmc`. また, `nomcmc` を用いる際は `cubature` が必要 (`library(cubature)` でインストールできる).
- `mcmc`: `method` で `mcmc` を選んだ場合に指定する必要があり, MCMC の繰り返し回数を表す. デフォルトは 1000.

である.

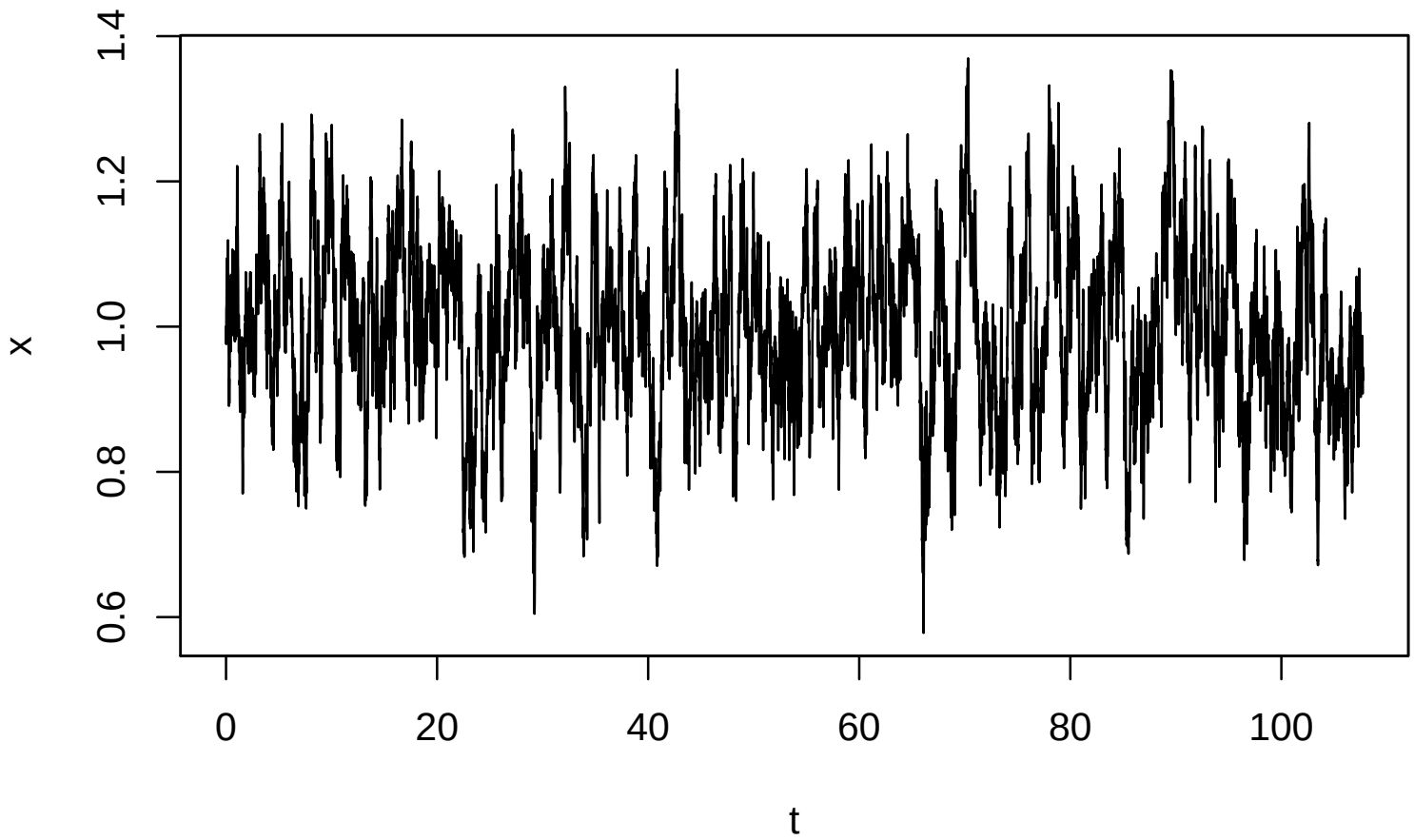
シミュレーション

一例として, オルンシュタイン・ウーレンベックモデルのパラメータ推定を考えてみよう.

$$dX(t) = -\alpha_1(X(t) - \alpha_2)dt + \beta_1 dB(t), X_0 = 1.$$

ここで, パラメータの真値を $(\alpha_{1,0}, \alpha_{2,0}, \beta_{1,0}) = (3, 1, 0.3)$ とし, $n = 10000, h_n = 5n^{-2/3}$ とする. またパラメータ空間をそれぞれ, $\Theta_\alpha = [0.1, 5] \times [0.5, 2]$, $\Theta_\beta = [0.1, 2.5]$ とする. この時モデルの設定やデータの生成は, 関数 `setModel` や `simulate` を用いて下記のコードで実行できる.

```
mod.ou <- setModel(drift="-alpha1*(x-alpha2)",diffusion="beta")
n <- 10000
Terminal <- 5*n^{1/3}
samp <- setSampling(Initial = 0, Terminal = Terminal, n = n)
ou <- setYuima(mod=mod.ou,sampling = samp)
set.seed(123)
x0 <- 1
param <- list(alpha1 = 3, alpha2 = 1, beta = 0.3)
obj <- simulate(ou,xinit=x0,true.parameter=param)
plot(obj)
```



この時, パラメータ $(\alpha_1, \alpha_2, \beta_1)$ を関数 `qmle` および `adaBayes` を用いて推定する. コードは以下の通り:

```
### パラメータの設定
start <- list(alpha1 = 2, alpha2 = 1.3, beta = 1.6)
lower <- list(alpha1 = 0.1, alpha2 = 0.5, beta = 0.1)
upper <- list(alpha1 = 5, alpha2 = 2, beta = 2.5)

### 事前分布の設定
prior <- list(alpha1 = list(measure.type = "code",
                            df = "dunif(alpha1, 0.1, 5)"),
              alpha2 = list(measure.type = "code",
                            df = "dunif(alpha1, 0.5, 2)"),
              beta = list(measure.type = "code",
                          df = "dunif(alpha1, 0.1, 2.5)"))

### 疑似最尤推定量の計算
res <- qmle(obj,start = start, lower = lower,
            upper = upper, joint = TRUE, rcpp = TRUE)

### 適応型疑似ベイズ推定量の計算
bres <- adaBayes(obj,start = start, lower = lower,
```

```
                  upper = upper, rcpp = TRUE, method =
                    "nomcmc")
```

出力は summary 関数を用いることで見ることができる.

```
### 出力
summary(res)
```

```
## Quasi-Maximum likelihood estimation
##
## Call:
## qmle(yuima = obj, start = start, lower = lower, upper = upper,
##     joint = TRUE, rcpp = TRUE)
##
## Coefficients:
##         Estimate  Std. Error
## beta   0.3000204 0.002121366
## alpha1 3.0451871 0.236139708
## alpha2 0.9924896 0.009492667
##
## -2 log L: -41007.26
```

```
summary(bres)
```

```
## Maximum likelihood estimation
##
## Call:
## adaBayes(yuima = obj, start = start, lower = lower, upper = upper,
##     method = "nomcmc", rcpp = TRUE)
##
## Coefficients:
##         Estimate  Std. Error
## beta   0.3000572 0.002120743
## alpha1 3.0279170 0.236168742
## alpha2 0.9924783 0.009547987
##
## -2 log L:
```

上記で述べたように, qmle と adaBayes の返り値 (上のコードでは res, bres にあたる) は引数の yuima オブジェクトやオプションの情報を格納している. この内容をすべて見るには, str 関数を用いれば良い (yuima オブジェクトについても同様).

```
str(res)
```

```
## Formal class 'yuima.qmle' [package "yuima"] with 10 slots
##   ..@ model    :Formal class 'yuima.model' [package "yuima"] with 16 slots
##   .. .. ..@ drift          :  expression((-alpha1 * (x - alpha2)))
##   .. .. ..@ diffusion      :List of 1
##   .. .. .. ..$ :  expression((beta))
##   .. .. ..@ hurst          : num 0.5
##   .. .. ..@ jump.coeff     : list()
##   .. .. ..@ measure        : list()
```

```
##   .. .. ..@ measure.type   : chr(0)
##   .. .. ..@ parameter      :Formal class 'model.parameter' [package "yuima"]
                               with 7 slots
##   .. .. .. .. ..@ all      : chr [1:3] "alpha1" "alpha2" "beta"
##   .. .. .. .. ..@ common   : chr(0)
##   .. .. .. .. ..@ diffusion: chr "beta"
##   .. .. .. .. ..@ drift    : chr [1:2] "alpha1" "alpha2"
##   .. .. .. .. ..@ jump     : chr(0)
##   .. .. .. .. ..@ measure  : chr(0)
##   .. .. .. .. ..@ xinit    : chr(0)
##   .. .. ..@ state.variable : chr "x"
##   .. .. ..@ jump.variable  : chr(0)
##   .. .. ..@ time.variable  : chr "t"
##   .. .. ..@ noise.number   : num 1
##   .. .. ..@ equation.number: int 1
##   .. .. ..@ dimension      : int [1:6] 3 0 1 2 0 0
##   .. .. ..@ solve.variable : chr "x"
##   .. .. ..@ xinit          :  expression(1)
##   .. .. ..@ J.flag         : logi FALSE
##   ..@ call     : language qmle(yuima = obj, start = start, lower = lower,
                     upper = upper, joint = TRUE, rcpp = TRUE)
##   ..@ coef     : Named num [1:3] 0.3 3.045 0.992
##   .. ..- attr(*, "names")= chr [1:3] "beta" "alpha1" "alpha2"
##   ..@ fullcoef : Named num [1:3] 0.3 3.045 0.992
##   .. ..- attr(*, "names")= chr [1:3] "beta" "alpha1" "alpha2"
##   ..@ vcov     : num [1:3, 1:3] 4.50e-06 -3.80e-08 1.62e-10 -3.80e-08 5.58e-02 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:3] "beta" "alpha1" "alpha2"
##   .. .. ..$ : chr [1:3] "beta" "alpha1" "alpha2"
##   ..@ min      : num -20504
##   ..@ details  :List of 2
##   .. ..$ par    : Named num [1:3] 0.3 3.045 0.992
##   .. .. ..- attr(*, "names")= chr [1:3] "beta" "alpha1" "alpha2"
##   .. ..$ hessian: num [1:3, 1:3] 2.22e+05 1.51e-01 -4.13e-01 1.51e-01 1.79e+01 ...
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:3] "beta" "alpha1" "alpha2"
##   .. .. .. ..$ : chr [1:3] "beta" "alpha1" "alpha2"
##   ..@ minuslogl:function (yuima, param, print = FALSE, env, rcpp = FALSE)
##   ..@ nobs     : int 10000
##   ..@ method   : chr "BFGS"
```

```
str(bres)
```

```
## Formal class 'mle' [package "stats4"] with 9 slots
##   ..@ call     : language adaBayes(yuima = obj, start = start, lower = lower,
                                    upper = upper, method = "nomcmc", rcpp = TRUE)
##   ..@ coef     : Named num [1:3] 0.3 3.028 0.992
##   .. ..- attr(*, "names")= chr [1:3] "beta" "alpha1" "alpha2"
##   ..@ fullcoef : Named num [1:3] 0.3 3.028 0.992
##   .. ..- attr(*, "names")= chr [1:3] "beta" "alpha1" "alpha2"
##   ..@ vcov     : num [1:3, 1:3] 4.50e-06 0.00 0.00 0.00 5.58e-02 ...
##   .. ..- attr(*, "dimnames")=List of 2
```

```
##   .. .. ..$ : chr [1:3] "beta" "alpha1" "alpha2"
##   .. .. ..$ : chr [1:3] "beta" "alpha1" "alpha2"
##   ..@ min      : num(0)
##   ..@ details  :List of 2
##   .. ..$ par    : Named num [1:3] 0.3 3.028 0.992
##   .. .. ..- attr(*, "names")= chr [1:3] "beta" "alpha1" "alpha2"
##   .. ..$ hessian: num [1:3, 1:3] 222343.2 0 0 0 17.9 ...
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:3] "beta" "alpha1" "alpha2"
##   .. .. .. ..$ : chr [1:3] "beta" "alpha1" "alpha2"
##   ..@ minuslogl:function ()
##   ..@ nobs     : int(0)
##   ..@ method   : chr "nomcmc"
```

また, 各項目を抜き出して表示する場合は, ...@-とすれば良い. 特に, パラメータの推定値は ...@coef により取り出すことができる.

```
res@coef
```

```
##      beta    alpha1    alpha2
## 0.3000204 3.0451871 0.9924896
```

```
res@model@drift
```

```
## expression((-alpha1 * (x - alpha2)))
```

```
bres@coef
```

```
##      beta    alpha1    alpha2
## 0.3000572 3.0279170 0.9924783
```

実データへのフィッティング

本節では, 実際に qmle を用いて実データへのフィッティングを行う. 解析対象のデータは, YUIMA パッケージに内蔵されている SPX500 の 2012 年 7 月 9 日から 2015 年 4 月 1 日までの日次データ (671 個) を用いる. 最初の 600 個を用いて推定量を構成し, 残りの 71 個を用いてそのパフォーマンスを確認する. SPX500 データは, 下記のコードを入力すれば Data としてダウンロードされる.

```
data(LogSPX)
```

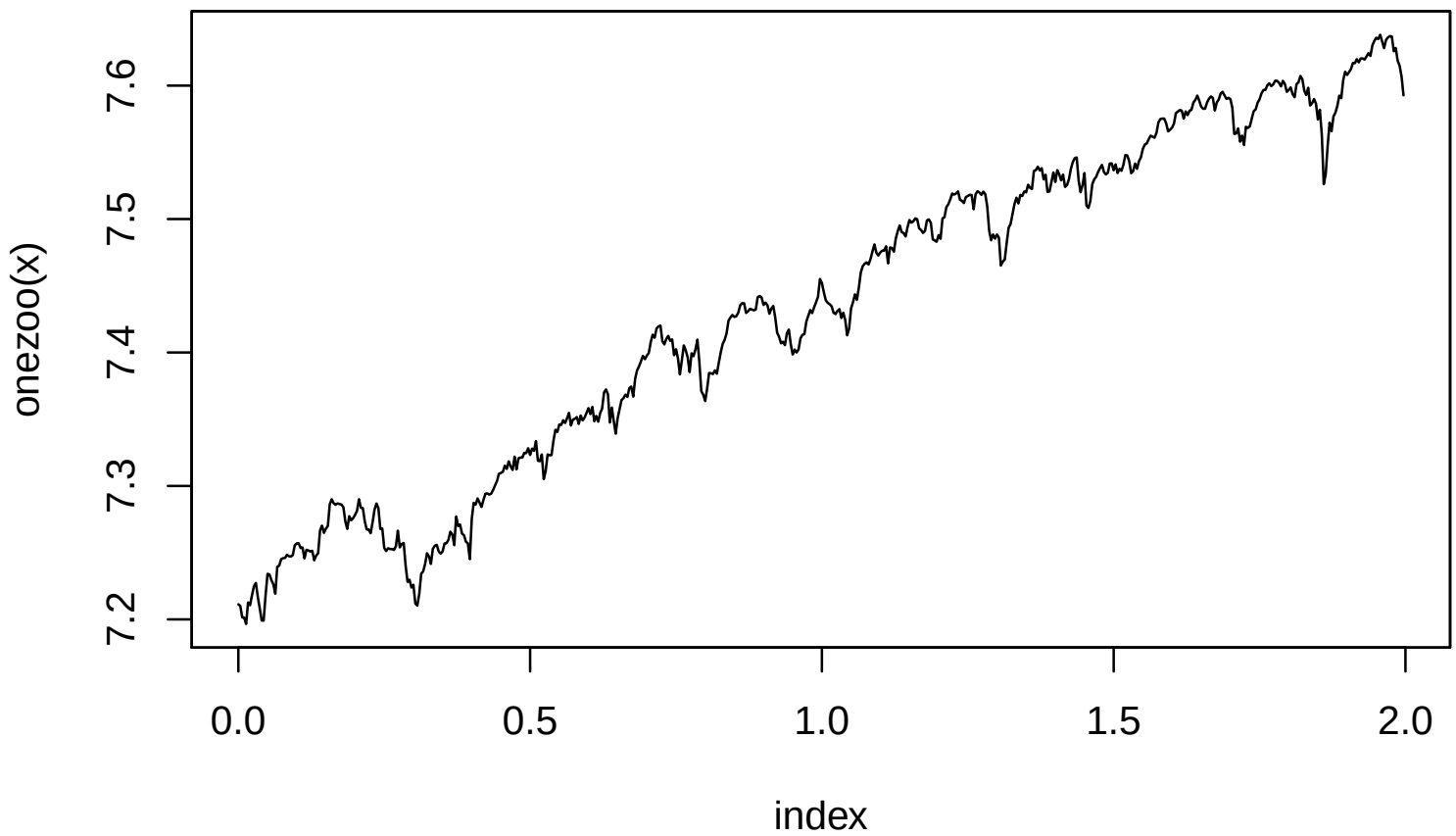
ここでは, SPX500 の日毎の対数価格 logDayprice を用いる.

```
str(Data)
x <- Data$logdayprice[1:600]
```

```
## List of 3
##  $ allObs     : num [1:53009] 0.00 -7.37e-05 -6.94e-04 -3.13e-03 -3.85e-03 ...
##  $ obsinday   : int 79
##  $ logdayprice: num [1:671] 7.21 7.21 7.2 7.2 7.2 ...
```

YUIMA 上で解析するためにデータオブジェクトを関数 setData と setYuima を利用して構築する. setData 内の delta を設定することで, ベクトルデータ である logdayprice に時間を紐づけできる.

```
y <- setYuima(data=setData(x,delta = 1/300))
plot(y)
```



上記のデータについて, ここではブラック・ショールズモデルの当てはめを考える. ブラック・ショールズモデルとはドリフト係数が $a(x,\mu) = \mu x$, 拡散係数が $b(x,\sigma) = \sigma x$ の拡散過程モデル

$$dX(t) = \mu X(t)dt + \sigma X(t)dB(t)$$

であった. YUIMA 上では, これまで見てきたように setModel 関数を用いてこのモデルを構成できる. また, setYuima 関数を利用することで, このモデルに先ほど入手したデータを格納することができる. str を用いてその中身を見てみると, @data の欄にデータが格納されていることがわかる.

```
SPX.mod <- setModel(drift = c("mu*x"),
                    diffusion = matrix(c("sigma*x"),1,1))
SPX.mod.data <- setYuima(model = SPX.mod, data = y@data)
str(SPX.mod.data)
```

```
## Formal class 'yuima' [package "yuima"] with 5 slots
##   ..@ data          :Formal class 'yuima.data' [package "yuima"] with 2 slots
##   .. .. ..@ original.data: num [1:600] 7.21 7.21 7.2 7.2 7.2 ...
##   .. .. ..@ zoo.data     :List of 1
##   .. .. .. ..$ :'zoo' series from 0 to 1.99666666666667
##   Data: num [1:600] 7.21 7.21 7.2 7.2 7.2 ...
##   Index:  num [1:600] 0 0.00333 0.00667 0.01 0.01333 ...
##   ..@ model         :Formal class 'yuima.model' [package "yuima"] with 16 slots
##   .. .. ..@ drift          :  expression((mu * x))
##   .. .. ..@ diffusion      :List of 1
##   .. .. .. ..$ :  expression((sigma * x))
##   .. .. ..@ hurst          : num 0.5
```

```
##   .. .. ..@ jump.coeff     : list()
##   .. .. ..@ measure        : list()
##   .. .. ..@ measure.type   : chr(0)
##   .. .. ..@ parameter      :Formal class 'model.parameter' [package "yuima"]
                                with 7 slots
##   .. .. .. .. ..@ all      : chr [1:2] "mu" "sigma"
##   .. .. .. .. ..@ common   : chr(0)
##   .. .. .. .. ..@ diffusion: chr "sigma"
##   .. .. .. .. ..@ drift    : chr "mu"
##   .. .. .. .. ..@ jump     : chr(0)
##   .. .. .. .. ..@ measure  : chr(0)
##   .. .. .. .. ..@ xinit    : chr(0)
##   .. .. ..@ state.variable : chr "x"
##   .. .. ..@ jump.variable  : chr(0)
##   .. .. ..@ time.variable  : chr "t"
##   .. .. ..@ noise.number   : int 1
##   .. .. ..@ equation.number: int 1
##   .. .. ..@ dimension      : int [1:6] 2 0 1 1 0 0
##   .. .. ..@ solve.variable : chr "x"
##   .. .. ..@ xinit          :  expression((0))
##   .. .. ..@ J.flag         : logi FALSE
##   ..@ sampling      :Formal class 'yuima.sampling' [package "yuima"] with 11 slots
##   .. .. ..@ Initial      : num 0
##   .. .. ..@ Terminal     : num 2
##   .. .. ..@ n            : int 600
##   .. .. ..@ delta        : num 0.00333
##   .. .. ..@ grid         :List of 1
##   .. .. .. ..$ : num [1:600] 0 0.00333 0.00667 0.01 0.01333 ...
##   .. .. ..@ random       : logi FALSE
##   .. .. ..@ regular      : logi TRUE
##   .. .. ..@ sdelta       : num(0)
##   .. .. ..@ sgrid        : num(0)
##   .. .. ..@ oindex       : num(0)
##   .. .. ..@ interpolation: chr "pt"
##   ..@ characteristic:Formal class 'yuima.characteristic' [package "yuima"]
                         with 2 slots
##   .. .. ..@ equation.number: int 1
##   .. .. ..@ time.scale     : num 1
##   ..@ functional    :Formal class 'yuima.functional' [package "yuima"]
                          with 4 slots
##   .. .. ..@ F    : NULL
##   .. .. ..@ f    : list()
##   .. .. ..@ xinit: num(0)
##   .. .. ..@ e    : num(0)
```

以上で関数 qmle を適用する準備が整った. ...@coef により, qmle の出力からパラメータの推定値が抽出できることを思い出すと, その推定値は以下で与えられる.

```
SPX.fit <- qmle(SPX.mod.data,
                lower=list(mu=0.01, sigma=0.01),
                start=list(mu=1, sigma=1),
```

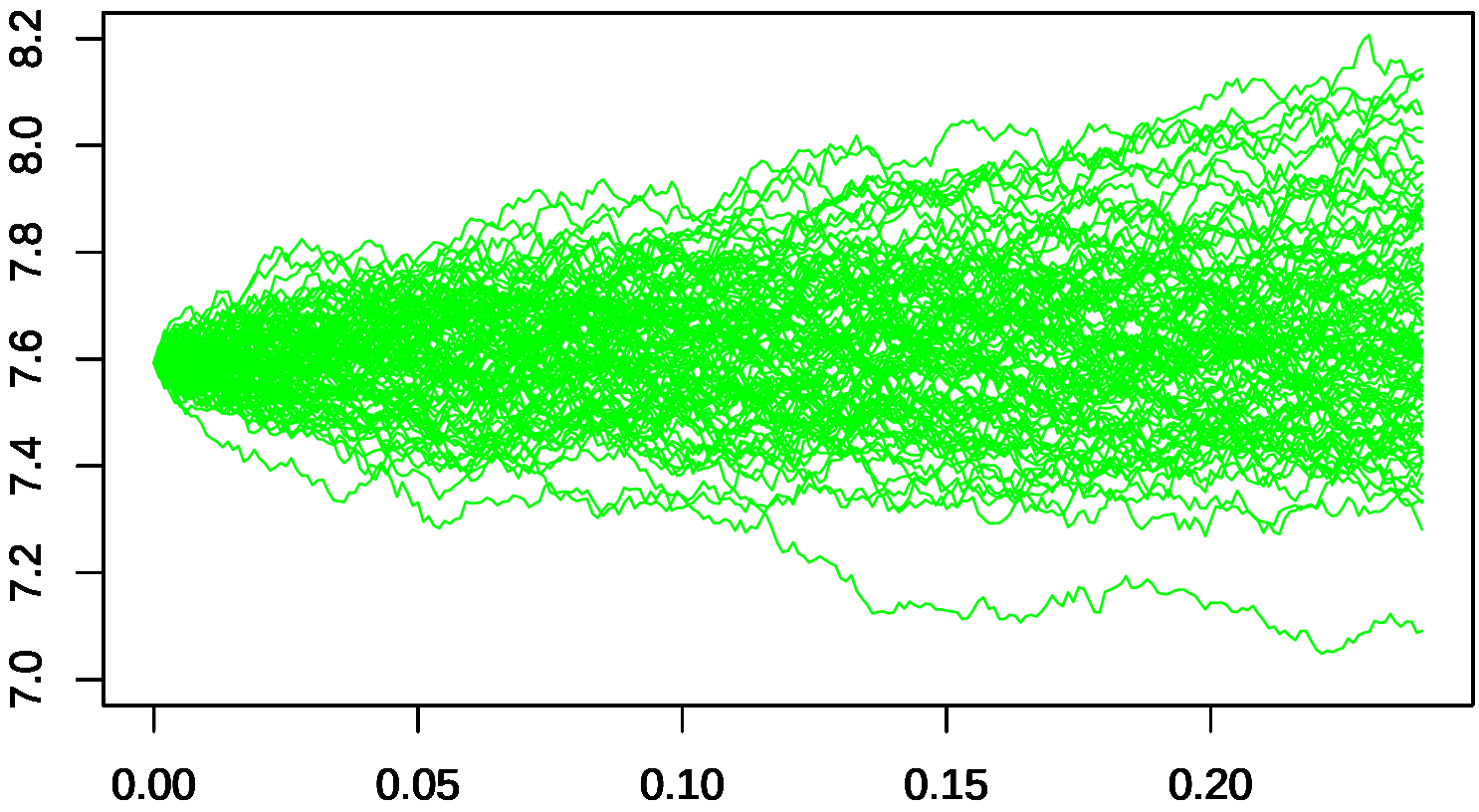
```
                upper=list(mu=10, sigma=10),
                method="L-BFGS-B")
SPX.fit@coef
```

```
##      sigma         mu
## 0.05846272 0.02593877
```

以下では, この推定値を使ったパスの予測を行ってみよう.

次に, 上で算出したパラメータ μ, σ の推定値を用いて yuima オブジェクトを再構成し, 100 本のサンプルパスを simulate 関数を用いて発生させる. 具体的には, simulate 関数の true.parameter で推定値 (SPX.fit@coef) を代入し, サンプルパスを発生させればよい.

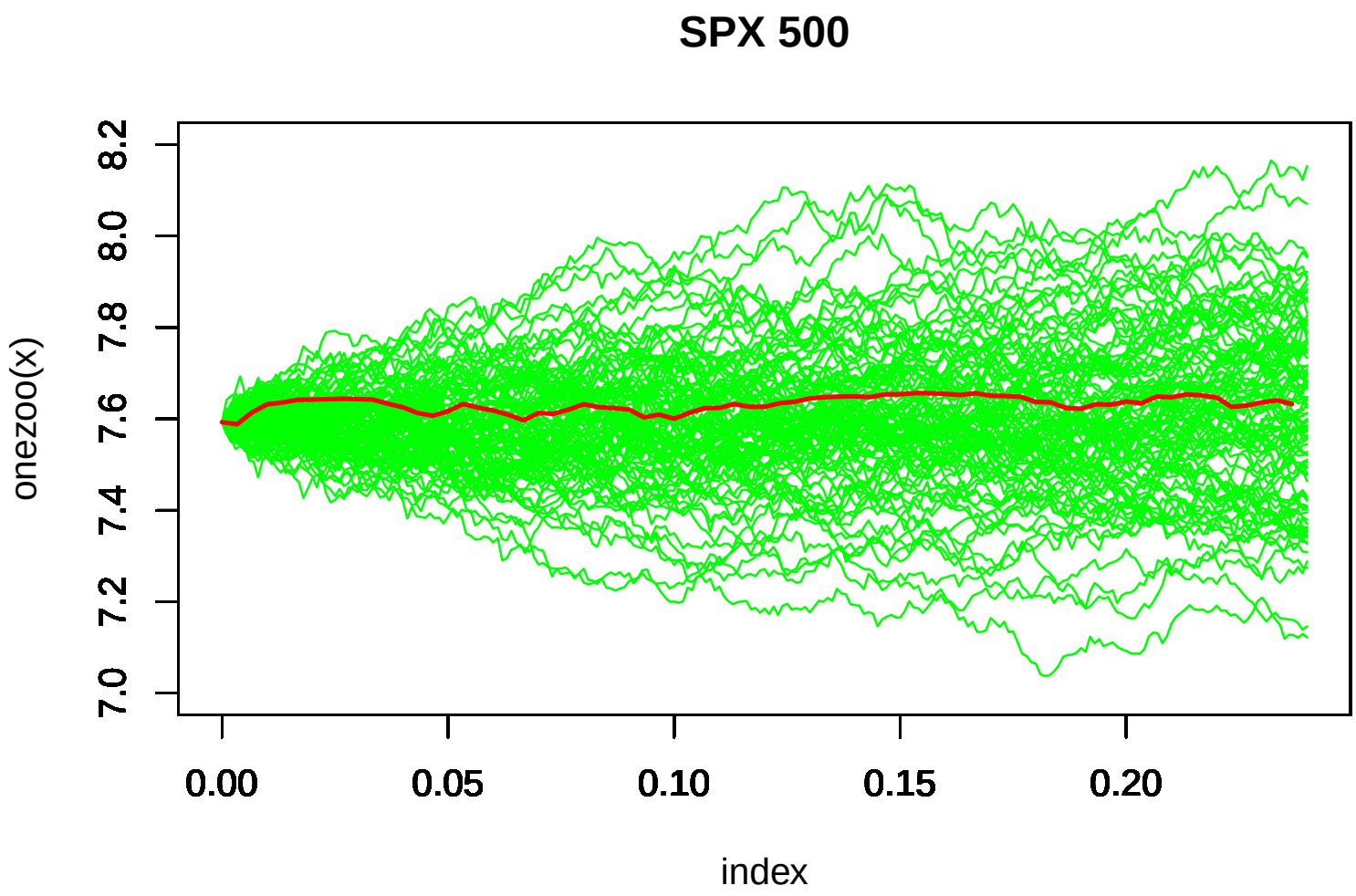
```
SPX.mod.pred <- setYuima(model=SPX.mod,
                         sampling =
                           setSampling(Terminal=0.24,
                                       n=240))
for(i in 1:100){
  plot(simulate(SPX.mod.pred,
                true.parameter =
                 list(sigma=SPX.fit@coef[1],
                      mu=SPX.fit@coef[2]),
                      xinit = Data$logdayprice[600]),
                 ylim=c(7,8.2),xlim=c(0,0.24),
                 par(new=TRUE), col="green",
                 xlab=c(""), ylab=c(""))
}
```



このサンプルパスに実際のデータを重ねてみると, 下図のようになる.

```
SPX.y <- setYuima(data=setData(Data$logdayprice[600:671],
                               delta=1/300))
plot(SPX.y,
     ylim=c(7,8.2),xlim=c(0,0.24),
```

```
      par(new=TRUE),col="red",
      lwd=2, main="SPX500"
)
```



補足: quantmod を用いたデータの入手およびパラメータ推定

下準備として, 金融関連のデータを入手可能な quantmod パッケージをインストールする.

```
install.packages("quantmod")
require(quantmod)
```

このパッケージ内の関数 getSymbols を用いることで, 株式の日次データの取得ができる.

```
getSymbols("IBM", from = "2016-01-01", to = "2017-12-31")
```

上例でわかるように期間は, from および to で指定することができる. ここで入手したデータは, xts オブジェクトと呼ばれるものに変換されているが, YUIMA 上で解析するためにデータオブジェクトを関数 setData と setYuima を利用して構築する.

```
x <- setYuima(data=setData(IBM$IBM.Close))
str(x@data) ###  xのデータスロットにIBM.Closeデータが確認できる
```

```
## Formal class 'yuima.data' [package "yuima"] with 2 slots
##   ..@ original.data:An 'xts' object on 2016-01-04/2017-12-29 containing:
##   Data: num [1:503, 1] 136 136 135 133 132 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr "IBM.Close"
##   Indexed by objects of class: [Date] TZ: UTC
##   xts Attributes:
## List of 2
##   .. ..$ src    : chr "yahoo"
##   .. ..$ updated: POSIXct[1:1], format: "2018-11-05 18:20:43"
##   ..@ zoo.data     :List of 1
##   .. ..$ IBM.Close:'zoo' series from 2016-01-04 to 2017-12-29
##   Data: num [1:503] 136 136 135 133 132 ...
##   Index:  Date[1:503], format: "2016-01-04" ...
```

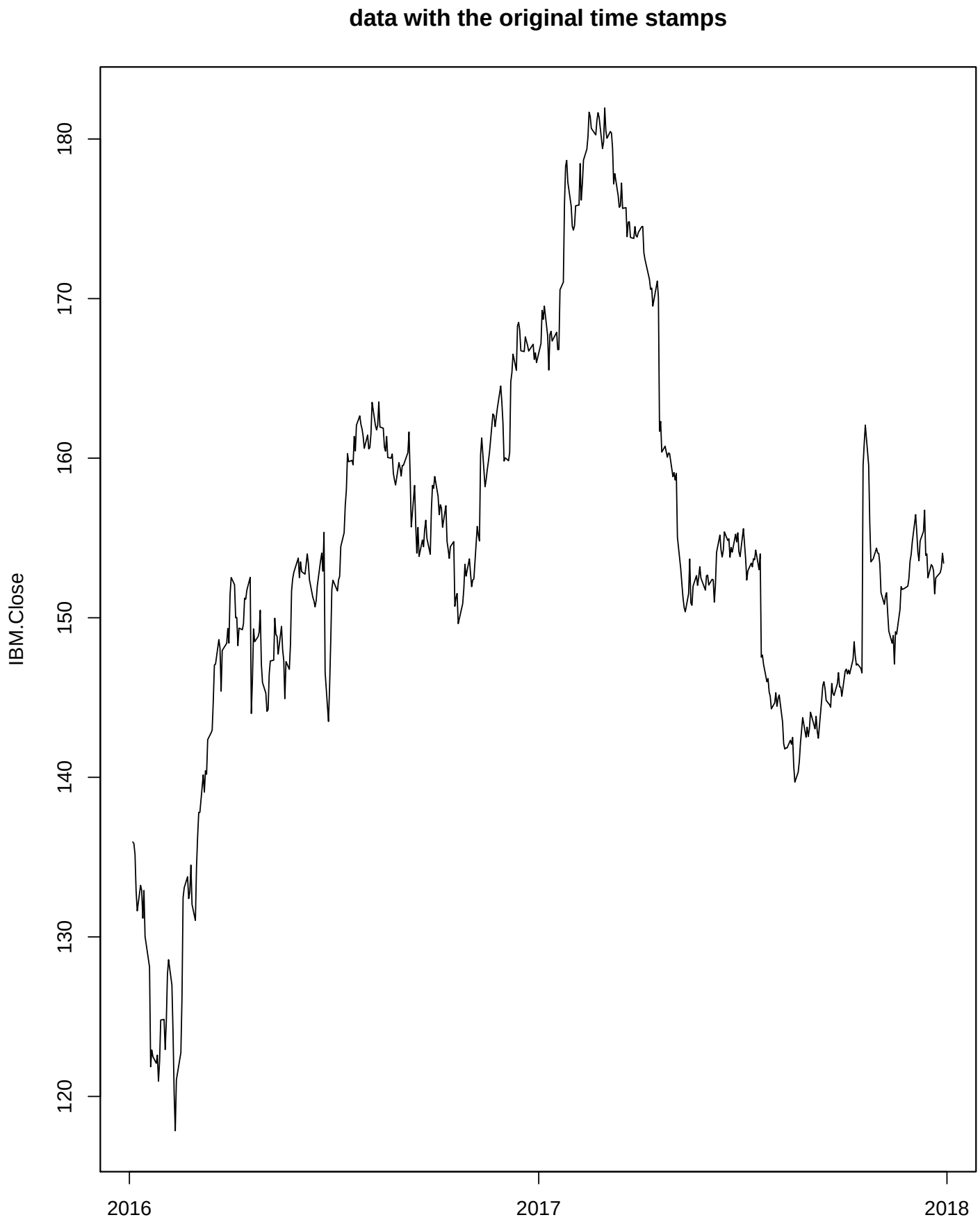
上記の Index の欄を見ると, 各時点データは年月日により紐づけられていることがわかる. 拡散過程によるモデリングを行うため, 下のように Index を変更する.

```
### 時間の単位として 1 年を 1 単位時間とする。
###  delta = 1/252 = 0.00397, 252 は 1 年のワーキングデーの数
y <- setYuima(data=setData(IBM$IBM.Close, delta=1/252))
str(y@data) ### 1 ステップの時間が 0.00397 となった。
```

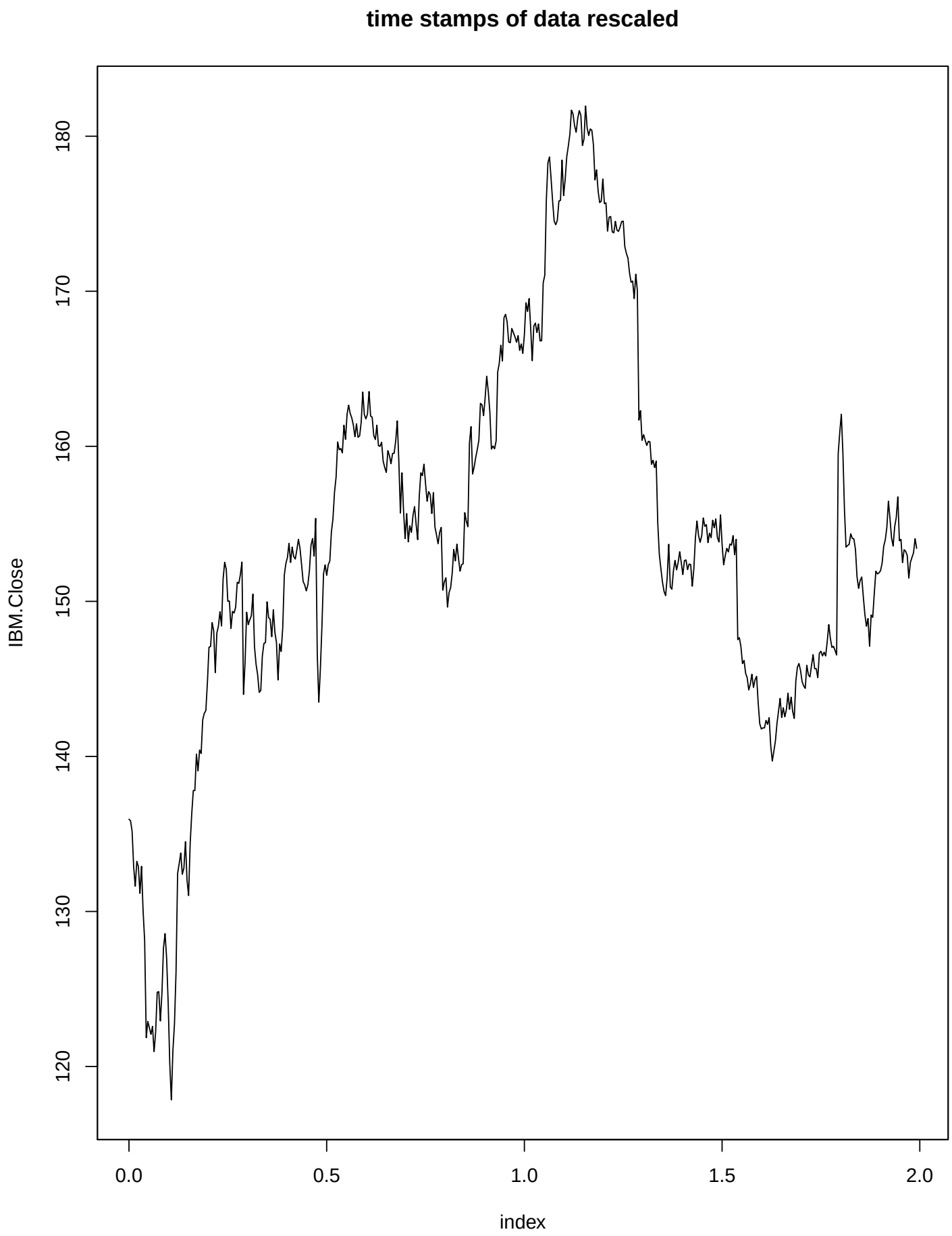
```
## Formal class 'yuima.data' [package "yuima"] with 2 slots
##   ..@ original.data:An 'xts' object on 2016-01-04/2017-12-29 containing:
##   Data: num [1:503, 1] 136 136 135 133 132 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr "IBM.Close"
##   Indexed by objects of class: [Date] TZ: UTC
##   xts Attributes:
## List of 2
##   .. ..$ src    : chr "yahoo"
##   .. ..$ updated: POSIXct[1:1], format: "2018-11-05 18:20:43"
##   ..@ zoo.data     :List of 1
##   .. ..$ IBM.Close:'zoo' series from 0 to 1.99206349206349
##   Data: num [1:503] 136 136 135 133 132 ...
##   Index:  num [1:503] 0 0.00397 0.00794 0.0119 0.01587 ...
```

ここで, 変更前と変更後のプロットを眺めてみると, 時間軸が変わっていることがわかる.

```
plot(x, main="data with the original time stamps")
```



```
plot(y, main="time stamps of data rescaled")
```



上記のデータについて, ここではブラック・ショールズモデルの当てはめを考える. ブラック・ショールズモデルとはドリフト係数が $a(x,\mu)=\mu x$, 拡散係数が $b(x,\sigma)=\sigma x$ の拡散過程モデル

$$dX(t)=\mu X(t)dt+\sigma X(t)dB(t)$$

であった. YUIMA 上では, これまで見てきたように `setModel` 関数を用いてこのモデルを構成できる. また, `setYuima` 関数を利用することで, このモデルに先ほど入手したデータを格納することができる.

```
IBM.mod <- setModel(drift=c("mu*x"),diffusion=matrix(c("sigma*x"),1,1))
IBM.mod.data <- setYuima(model=IBM.mod, data=y@data)
```

以上で関数 qmle を適用する準備が整った. ...@coef により, qmle の出力からパラメータの推定値が抽出できることを思い出すと, その推定値は以下で与えられる.

```
IBM.fit <- qmle(IBM.mod.data, lower=list(mu=0.01, sigma=0.01),
                start=list(mu=1, sigma=1), upper=list(mu=10, sigma=10),
                method="L-BFGS-B")
IBM.fit@coef
```

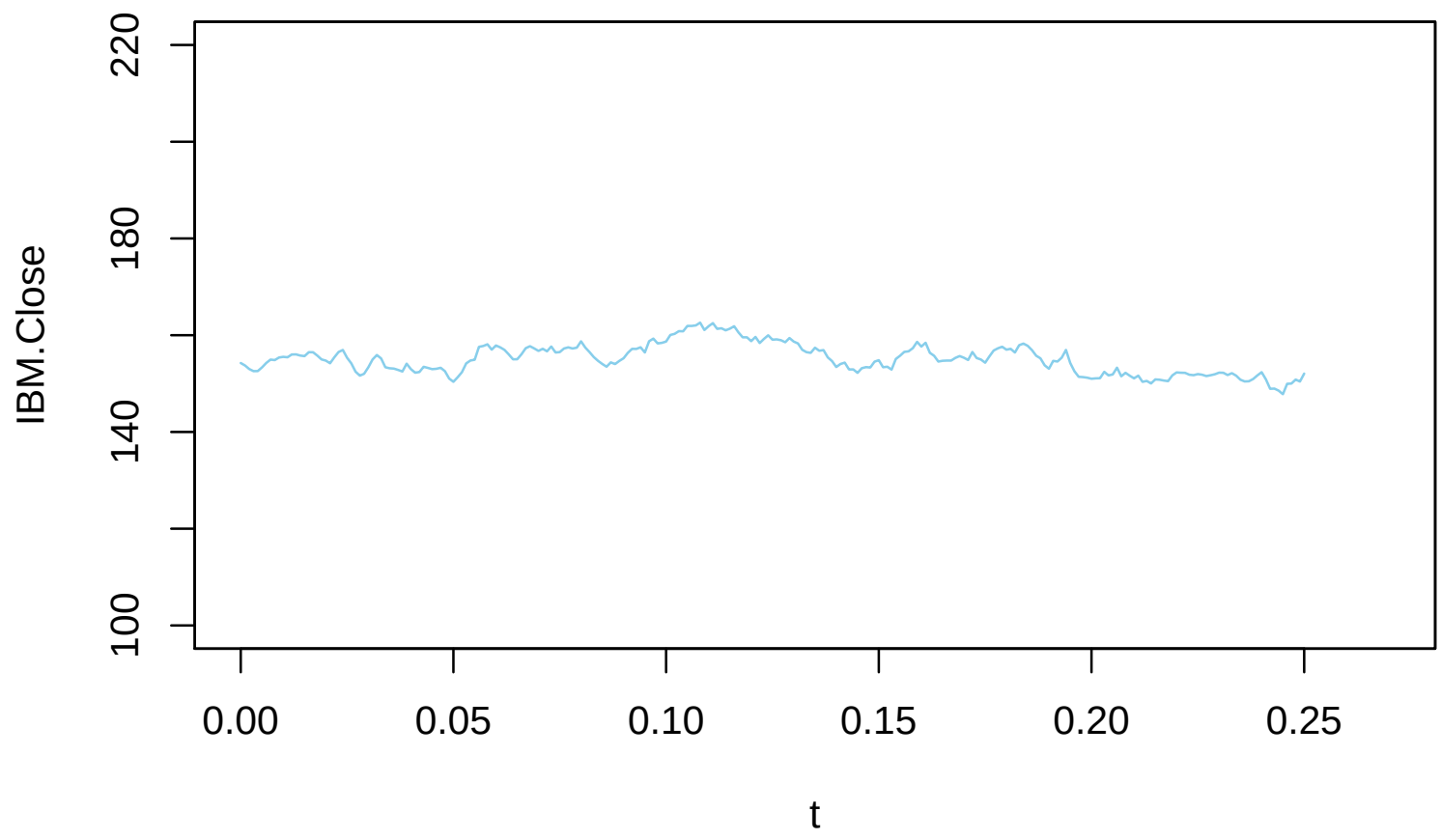
```
##       sigma          mu
## 0.18820717 0.07668582
```

以下では, この推定値を使ったパスの予測を行ってみよう. 再び, getSymbols を用いて 2018/01/02 - 2018/04/05 間の日次データを抽出する.

```
getSymbols("IBM", from = "2018-01-02", to = "2018-04-05")
IBM.first <- IBM$IBM.Close["2018-01-02"]
```

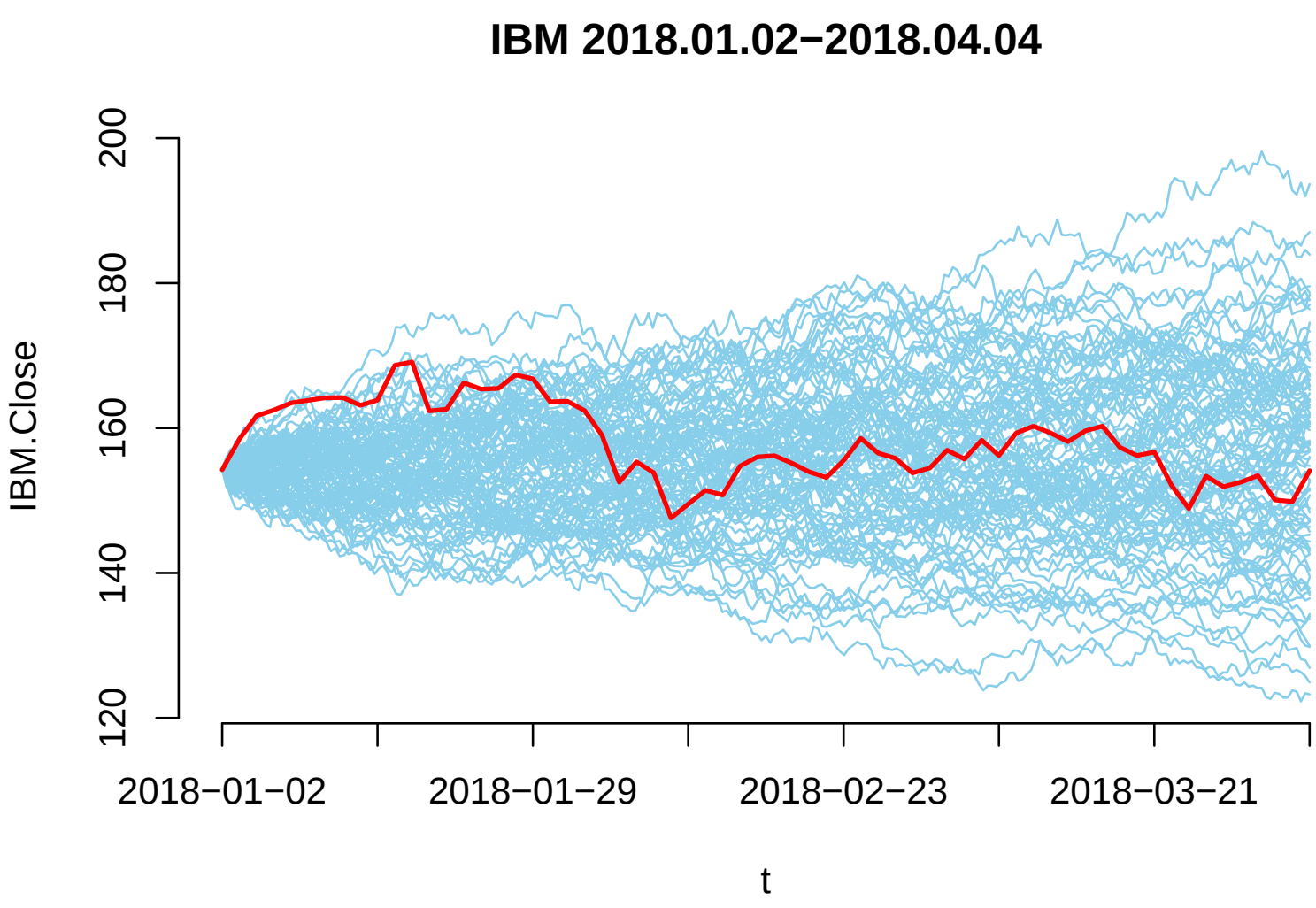
次に, 上で算出したパラメータ μ, σ の推定値を用いて yuima オブジェクトを再構成し, サンプルパスを simulate 関数を用いて発生させる. 具体的には, simulate 関数の true.parameter で推定値を代入すれば良い. また, 初期値 xinit では 2018/01/02 の終値を代入している.

```
set.seed(127)
## 0.25年後の予測モデルを構築
IBM.mod.pred <- setYuima(model=IBM.mod,
                         sampling = setSampling(Terminal=0.25, n=250))
IBM.pred <- simulate(IBM.mod.pred,
                     true.parameter = list(sigma=IBM.fit@coef[1],
                                           mu=IBM.fit@coef[2]),
                     xinit = IBM.first)
plot(IBM.pred, ylim=c(100,220), xlim=c(0,0.27), col="skyblue",
     ylab=c("IBM.Close"),xlab=c("t"))
```



1 本では傾向がわからないため, 100 本のサンプルパスを発生させ, 最後に実際のデータを重ねる.

```r
## シミュレーターの作成
mypred <- function(param){
  result <- simulate(IBM.mod.pred,
                     true.parameter = list(sigma=param[1], mu=param[2]),
                     xinit = IBM.first)
  return(get.zoo.data(result)[[1]])
}
## 100本のパスをシミュレートして結果を保存
path <- replicate(100, mypred(IBM.fit@coef))
## タイムインデックス
time.idx <- IBM.mod.pred@sampling@grid[[1]]
## シミュレートしたパスをプロット
matplot(time.idx, path, type = "l", lty = 1, col="skyblue",
        xlab = "t", ylab = "IBM.Close", axes = FALSE,
        main="IBM 2018.01.02-2018.04.04")
IBM2018.y <- setYuima(data=setData(IBM$IBM.Close, delta=1/252))
IBM2018.y <- get.zoo.data(IBM2018.y)[[1]]
## 実際のパスを上書き
lines(time(IBM2018.y),IBM2018.y, col="red",lwd=2)
## 軸の設定
idx <- seq(1, length(time(IBM)), by = 9)
axis(1, at = time(IBM2018.y)[idx], labels = time(IBM)[idx])
axis(2)
```



参考文献

[1] M. Kessler. Estimation of an ergodic diffusion from discrete observations. *Scand. J. Statist.*, 24(2):211–229, 1997.

[2] N. Yoshida. Estimation for diffusion processes from discrete observation. *Journal of Multivariate Analysis*, 41(2):220–242, 1992.

[3] N. Yoshida. Polynomial type large deviation inequalities and quasi-likelihood analysis for stochastic differential equations. *Ann. Inst. Statist. Math.*, 63(3):431–479, 2011.