dlm: 動的線型モデルに関するベイズ分析のための R パッケージ

Giovanni Petris
アーカンソー大学、ファイエットビル AR

2009-01-14

1 動的線型モデルの定義と操作

パッケージ dlm の焦点は、線型状態空間モデル ([H, WH] を参照) としても知られる動的線型モデル (Dynamic Linear Model; DLM) に関するベイズ分析にある。このパッケージには、DLM のパラメータに関する最尤推定やカルマン・フィルタリングのための関数も含まれる。カルマン・フィルタリングや尤度の評価および状態ベクトルからの標本抽出に用いられるアルゴリズムは、関連する分散行列の特異値分解 (SVD) に基づいており ([ZL] を参照)、他のアルゴリズムより数値安定性が改善されている。

1.1 モデル

DLM は次の式で特定化される。

$$\begin{cases} y_t = F_t\theta_t + v_t, & v_t \sim \mathcal{N}(0, V_t) \\ \theta_t = G_t\theta_{t-1} + w_t, & w_t \sim \mathcal{N}(0, W_t) \end{cases}$$

ここで $t = 1, \ldots, n$ である。また、θ_0 に関する**事前**(prior) 分布は次の通りである。

$$\theta_0 \sim \mathcal{N}(m_0, C_0)$$

ここで y_t は m 次元のベクトルで時点 t における観測値を表しており、一方 θ_t は一般に観測不可能な p 次元のベクトルで時点 t におけるシステムの状態を表す。v_t は観測誤差であり、w_t は遷移誤差である。行列 F_t と G_t は各々 $m \times p$ と $p \times p$ の行列であり、一方 V_t と W_t は適切な次元の分散行列である。

1.2 dlm を用いた DLM の定義

最も簡単な DLM の 1 つはランダムウォーク・プラス・ノイズモデルであり、このモデルは 1 次の多項式モデルとも呼ばれる。このモデルを用いて一変量の観測値と 1 次元の状態ベクトルをモデル化する場合、モデルは次の式で記述される。

$$\begin{cases} y_t = \theta_t + v_t, & v_t \sim \mathcal{N}(0, V) \\ \theta_t = \theta_{t-1} + w_t, & w_t \sim \mathcal{N}(0, W) \end{cases}$$

このモデルは**固定的**(constant) である。すなわち、動的特性を定義する様々な行列が時不変となる。更に、$F_t = G_t = [1]$ である。このモデルにおけるパラメータは、観測分散 V と遷移分散 W のみである。通常これ

らは最尤法かベイズ法を用いて利用可能なデータから推定される。パッケージ dlm において固定的な DLM は要素 FF, V, GG, W を持つリストとして表現され、そのクラスは"dlm"となる。$V = 0.8$ かつ $W = 0.1$ としたランダムウォーク・プラス・ノイズモデルは、R では次のように定義できる。

```
> dlm(FF = 1, V = 0.8, GG = 1, W = 0.1, m0 = 0, C0 = 100)
```

事前分布 θ_0 に関して、その平均と分散を規定する必要があることに注意しよう。これは、モデルの定義に不可欠な部分である。同じモデルは、次のような別の方法でも定義できる。

```
> dlmModPoly(order = 1, dV = 0.8, dW = 0.1, C0 = 100)
```

この関数は、m0, C0, dV, dW に対してデフォルト値を設定する (最後の二つの dV, dW は、各々 V と W の**対角**(diagonal) 要素を規定するために用いられる)。実際にはこの関数は多項式モデルの**次数**(order) に対してもデフォルト値を設定するので、この関数を手軽に使う場合、利用者はこれらのデフォルト値を知っておくべきである。特に dV と dW に対するデフォルト値はモデルの特定化を完全にするためだけに存在しているので、より正確に考慮して設定されるべきである。

ここで、次のようにして得られる 2 次の多項式モデルを考えよう。

```
> myMod <- dlmModPoly()
```

パッケージが提供する抽出と置換のための関数を用いると、モデルに関する個々の要素をアクセスしたり変更したりする事が自然な方法で実現できる。

```
> FF(myMod)
```

```
     [,1] [,2]
[1,]    1    0
```

```
> W(myMod)
```

```
     [,1] [,2]
[1,]    0    0
[2,]    0    1
```

```
> m0(myMod)
```

```
[1] 0 0
```

```
> V(myMod) <- 0.8
```

dlmModPoly に加えて別の関数 dlm も提供されており、この関数は一般的な DLM を生成する。これらを表 1 にまとめた。

表 1　特定のモデルを生成する関数

関数	モデル
dlmModARMA	ARMA 過程
dlmModPoly	n 次の多項式 DLM
dlmModReg	線型回帰
dlmModSeas	周期性 (季節要素)
dlmModTrig	周期性 (三角関数による表現形態)

dlmModARMA は多変量の場合も扱えるが、この関数を除く他の生成関数では、観測値は一変量の場合に限定される。より複雑な DLM を明示的に定義するには、一般的な関数 dlm を使う。

1.3　モデルの結合：和と外部和

いくつかの基本的なモデルを「追加」(addition) してゆくとより一般的なモデルを得ることができるが、この方法には異なる形態が存在する。一般的に、m 次元の観測値に対して独立な DLM が k 種類存在すると仮定し、その中の i 番目の DLM が次の方程式系によって定義されているとしよう。

$$\begin{cases} y_t^{(i)} = F_t^{(i)}\theta_t^{(i)} + v_t^{(i)}, & v_t^{(i)} \sim \mathcal{N}(0, V^{(i)}) \\ \theta_t^{(i)} = G_t^{(i)}\theta_{t-1}^{(i)} + w_t^{(i)}, & w_t^{(i)} \sim \mathcal{N}(0, W^{(i)}) \end{cases} \tag{1}$$

$m_0^{(i)}$ と $C_0^{(i)}$ は、i 番目の DLM における初期の状態に関する平均と分散である。DLM が異なると、状態ベクトルの次元も $p_1, \ldots, p_k$ と異なる可能性がある事に注意しよう。各モデルによって観測過程に関する確率的なトレンドや周期成分といった特徴が簡潔に表現できる。時点 t における実際の観測値は $y_t = y_t^{(1)} + \cdots + y_t^{(k)}$ となる。このことは、システムの状態を $\theta_t' = \left({\theta_t^{(1)}}', \ldots, {\theta_t^{(k)}}'\right)$ と定義して DLM を結合（もしくは追加）すると、次のような行列を持つ統合的なモデルが得られる事を示している。

$$F_t = \left(F_t^{(1)} \mid \ldots \mid F_t^{(k)}\right), \qquad V_t = \sum_{i=1}^{k} V_t^{(i)}$$

$$G_t = \begin{bmatrix} G_t^{(1)} & & \\ & \ddots & \\ & & G_t^{(k)} \end{bmatrix}, \qquad W_t = \begin{bmatrix} W_t^{(1)} & & \\ & \ddots & \\ & & W_t^{(k)} \end{bmatrix}$$

$$m_0' = \left({m_0^{(1)}}', \ldots, {m_0^{(k)}}'\right), \qquad C_0 = \begin{bmatrix} C_0^{(1)} & & \\ & \ddots & \\ & & C_0^{(k)} \end{bmatrix}$$

ここで説明した形のモデルの合成は、モデルの和として考えることができる。パッケージ dlm は、dlm クラスのオブジェクトに対する総称 + に関して、このような DLM の和が実行できるメソッド関数を提供している。

例えば、雑音を伴って観測された時系列を、確率的な線型トレンドと四半期毎の季節成分の和としてモデル化したい場合を想定しよう。このモデルは次のようにして設定することができる。

```
> myMod <- dlmModPoly() + dlmModSeas(4)
```

dlmModPoly や dlmModSeas を呼び出す際に、行列 V や W における非零の要素をより意味のある値に規定する事もできる（結合モデルを設定した後に変更することもできる）。

DLM の結合にはもうひとつ別の自然な方法があり、この方法は「外部」(outer) 和に類似している。改めて (1) 式の k 種類のモデルを考えるが、今度は各モデルにおける観測値の次元が $m_1, \ldots, m_k$ のように異なる可能性がある場合を想定する。全ての $y_t^{(i)}$ を含む多変量のモデルを得る明白な方法は、各モデルが独立であると考え、$y_t' = \left({y_t^{(1)}}', \ldots, {y_t^{(k)}}'\right)$ と設定することである。各 $y_t^{(i)}$ は、それ自体が確率ベクトルとなり得ることに注意しよう。これは、前述の場合の様に状態ベクトルを $\theta_t' = \left({\theta_t^{(1)}}', \ldots, {\theta_t^{(k)}}'\right)$ として新しい DLM を定義する事に対応しており、この場合の各種の行列は次のようになる。

$$F_t = \begin{bmatrix} F_t^{(1)} & & \\ & \ddots & \\ & & F_t^{(k)} \end{bmatrix}, \qquad V_t = \begin{bmatrix} V_t^{(1)} & & \\ & \ddots & \\ & & V_t^{(k)} \end{bmatrix}$$

$$G_t = \begin{bmatrix} G_t^{(1)} & & \\ & \ddots & \\ & & G_t^{(k)} \end{bmatrix}, \qquad W_t = \begin{bmatrix} W_t^{(1)} & & \\ & \ddots & \\ & & W_t^{(k)} \end{bmatrix}$$

$$m_0' = \left({m_0^{(1)}}', \ldots, {m_0^{(k)}}'\right), \qquad C_0 = \begin{bmatrix} C_0^{(1)} & & \\ & \ddots & \\ & & C_0^{(k)} \end{bmatrix}$$

例えば二つの時系列があると仮定しよう。両系列共に雑音を伴って観測されるが、最初の系列は確率的な線型トレンドであり、後の系列はランダムウォーク・プラス・四半期毎の季節成分であるとする。この二つの系列が独立であると仮定すれば共用の DLM が考えられ、R では次のように設定ができる。

```
> dlmModPoly(dV = 0.2, dW = c(0, 0.5)) %+%
+     (dlmModSeas(4, dV = 0, dW = c(0, 0, 0.35)) +
+      dlmModPoly(1, dV = 0.1, dW = 0.03))
```

1.4 時変のモデル

時変の DLM では、F_t, V_t, G_t, W_t における要素の少なくとも 1 つが時間的に変化する。このような要素は、時系列、もしくはより一般的には数値ベクトルと考えることができる。モデルの行列における時変要素をすべてまとめると、多変量時系列、もしくは数値行列に格納することができる。この考え方は、時変 DLM の内部表現に関する基礎をなしている。dlm クラスのオブジェクト m には、JFF, JV, JGG, JW, X という名前の要素が含まれる可能性がある。最初の四つは各々 FF, V, GG, W と同じ次元の整数行列であるが、X は $n \times m$ の行列となる（ここで n はデータの観測数である）。FF の (i, j) 要素が時不変であれば JFF における対応する要素は 0 になるが、時変の場合には k となり、$F_t[i, j]$ の異なる時点における値が X[,k] にベクトルとして格納される。時変の場合、モデルのオブジェクトにおける FF[i,j] の値は、実際には使用されない。DLM の残りの要素に関しても同様である。例えば動的な線型回帰は、次のようにモデル化ができる。

$$\begin{cases} y_t = \alpha_t + x_t\beta_t + v_t & v_t \sim \mathcal{N}(0, V_t) \\ \alpha_t = \alpha_{t-1} + w_{\alpha,t}, & w_{\alpha,t} \sim \mathcal{N}(0, W_{\alpha,t}) \\ \beta_t = \beta_{t-1} + w_{\beta,t}, & w_{\beta,t} \sim \mathcal{N}(0, W_{\beta,t}) \end{cases}$$

ここで、システムの状態を $\theta_t = (\alpha_t,\ \beta_t)'$ とすると $F_t = [1,\ x_t]$ となる（β_t は時点 t における回帰係数、x_t は時点 t における共変量）。このような DLM は、R では以下のように設定ができる。

```
> u <- rnorm(25)
> myMod <- dlmModReg(u, dV = 14.5)
> myMod$JFF
```

```
     [,1] [,2]
[1,]    0    1
```

```
> head(myMod$X)
```

```
       [,1]
[1,]  0.639
[2,]  0.947
[3,]  3.128
[4,]  2.733
[5,] -0.480
[6,]  0.156
```

現時点では、時変 DLM の外部和は実行できないようになっている。

2 最尤推定

DLM を定義する行列に未知パラメータが含まれる場合がよくある。パッケージ dlm は主にベイズ推定のために開発されたが、最尤法を用いて未知パラメータを推定する事が可能な方法も提供されている。関数 dlmMLE は、実質的には optim を呼び出すラッパ関数である。この関数の引数には、データや最適化アルゴリズムのための初期値に加え、特定の値の未知パラメータベクトルから DLM を「構築」(build) する様な関数も必要となる。dlmMLE の使用法について、2, 3 の簡単な例とともに説明を行う。

ここでナイル川のデータセットを考えよう。合理的なモデルになりうるのはランダムウォーク・プラス・ノイズモデルであるるが、そのシステム分散と観測分散は未知である。このモデルは、以下に定義される関数を用いて構築することができる（分散が正の値を取ることを保証するために、対数スケールでパラメータ化している）。

```
> buildFun <- function(x) {
+     dlmModPoly(1, dV = exp(x[1]), dW = exp(x[2]))
+ }
```

任意の点 (ここでは $(0, 0)$) から最適化を始めた結果、パラメータの MLE は次のようになることがわかる。

```
> fit <- dlmMLE(Nile, parm = c(0,0), build = buildFun)
> fit$conv
```

```
[1] 0
```

```
> dlmNile <- buildFun(fit$par)
> V(dlmNile)

      [,1]
[1,] 15100

> W(dlmNile)

     [,1]
[1,] 1468
```

比較のために、StructTS を用いて得られた分散の推定値は次のようになる。

```
> StructTS(Nile, "level")

Call:
StructTS(x = Nile, type = "level")


Variances:
  level  epsilon
   1469    15099
```

あまり標準的な例ではないが、ナイル川の流量に関して、1898 年におけるアスワンダムの建設に伴う急変を考慮する場合を想定しよう。この様な要望は、1899 年にシステム分散を増加させることで実現可能となる(以下では 1 より大きな乗数を適用している)。

```
> buildFun <- function(x) {
+     m <- dlmModPoly(1, dV = exp(x[1]))
+     m$JW <- matrix(1)
+     m$X <- matrix(exp(x[2]), nc = 1, nr = length(Nile))
+     j <- which(time(Nile) == 1899)
+     m$X[j,1] <- m$X[j,1] * (1 + exp(x[3]))
+     return(m)
+ }
> fit <- dlmMLE(Nile, parm = c(0,0,0), build = buildFun)
> fit$conv

[1] 0

> dlmNileJump <- buildFun(fit$par)
> V(dlmNileJump)

      [,1]
[1,] 16300

> dlmNileJump$X[c(1, which(time(Nile) == 1899)), 1]
```

```
[1] 2.79e-02 6.05e+04
```

結論として 1899 年における急変を考慮すれば、その年の前後の期間における系列のレベルは、本質的には一定であるといえる。

3 フィルタリング、平滑化、予測

DLM のパラメータが全て既知の場合、状態と観測値の同時分布がガウス型になるという事実のおかげで、データが観測された下における、状態や将来の観測値に関する条件付き分布を導出する事がかなり容易になる。$i \leq j$ となる任意の整数の組 (i, j) に関して、i 番目から j 番目（を含む）までの観測値 $y_i, \ldots, y_j$ を $y_{i:j}$ で表そう。同様の記法は状態に関しても用いられる。時点 t における**フィルタリング**(filtering) 分布は、$y_{1:t}$ が与えられた下における θ_t の条件付き分布となる。時点 t における**平滑化**(smoothing) 分布は、$y_{1:t}$ が与えられた下における $\theta_{0:t}$ の条件付き分布となる。もしくは言葉の乱用になるが無害なので、その周辺分布 (つまり、$y_{1:t}$ が与えられた下における θ_s の条件付き分布 $(s \leq t)$) も全て平滑化分布と呼ばれる。明らかに $s = t$ では、この周辺分布はフィルタリング分布に一致する。時点 t までのデータが与えられた下での将来の状態や観測値に関する条件付き分布は、**予測**(forecast/predictive) 分布と呼ばれる。著名なカルマン・フィルタアルゴリズムやその拡張に基づく再帰的なアルゴリズムを用いれば、フィルタリング分布や平滑化分布の計算が可能になる。フィルタリング分布の平均と分散を「事前分布」(prior) の平均と分散 — m_0 と C_0 — として用いれば、モデルの定義から再帰的に予測分布が容易に導出できる。これは通常の逐次ベイズ更新であり、時点 t における事後分布は時点 t より後の観測値等に関する事前分布の役割を果たす。周辺分布もしくは条件付き分布 (特にフィルタリング分布、平滑化分布、及び予測分布) は全てガウス型となることに注意しよう（ガウス分布は平均と分散だけで同定される)。

3.1 フィルタリング

改めて、ナイル川のデータにあてはめを行った 6 ページの最後のモデルを考えよう。推定されたパラメータを既知であるかのように扱うと、`dlmFilter` を用いてフィルタリング分布が計算できる。データセットにおける観測数を n とすると、`dlmFilter` はデータから計算可能なフィルタリング分布に関する平均と分散を $n + 1$ 個分返す。このフィルタリング分布は、$y_{1:t}$ が与えられた下における θ_t の分布 $(t = 0, 1, \ldots, n)$ のことであり、$t = 0$ は事前分布 θ_0 に対応するのが慣例となっている。

```
> nileJumpFilt <- dlmFilter(Nile, dlmNileJump)
> plot(Nile, type = 'o', col = "seagreen")
> lines(dropFirst(nileJumpFilt$m), type = 'o',
+       pch = 20, col = "brown")
```

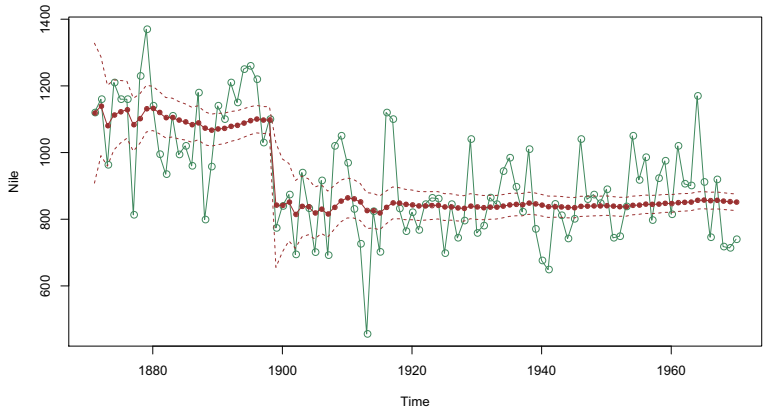

図 1　ナイル川のデータとフィルタ化レベル

フィルタリング分布の分散 $C_0, \ldots, C_n$ は、特異値分解 (SVD) の形で返される。非負定値対称行列 Σ の SVD は $\Sigma = UD^2U'$ となる。ここで U は直交行列であり、D は対角行列である。$C_0, \ldots, C_n$ の SVD における U や D に対応する行列は、`dlmFilter` から返さるリストの中の `U.C` や `D.C` という要素として各々見つけることができる。`U.C` は行列のリストとなるが、SVD の D の部分は対角行列なので、`D.C` は行列 D の対角成分を各列に格納していった行列となる。この様な分解をしておくと、平滑化の様な計算を更に行う事に関心がある場合、役に立つ。しかしながら、フィルタリングの分散それ自体が関心の対象となる場合もあるので、`dlm` ではユーティリティ関数 `dlmSvd2var` が提供されており、SVD から分散を再構築することができる。以下で説明する様に、このような分散は例えばフィルタリングにおける確率区間を計算するために用いることもできる。

```
> attach(nileJumpFilt)
> v <- unlist(dlmSvd2var(U.C, D.C))
> pl <- dropFirst(m) + qnorm(0.05, sd = sqrt(v[-1]))
> pu <- dropFirst(m) + qnorm(0.95, sd = sqrt(v[-1]))
> detach()
> lines(pl, lty = 2, col = "brown")
> lines(pu, lty = 2, col = "brown")
```

`dlmFilter` はフィルタリング分布に関する平均と分散に加え、状態の一期先予測分布 ($y_{1:t-1}$ が与えられた下での θ_t の分布) に関する平均と分散や、観測値の一期先予測分布 ($y_{1:t-1}$ が与えられた下での y_t の分布) に関する平均も返す (ここで $t = 1, \ldots, n$)。この場合も分散は SVD の形で返される。

3.2　平滑化

関数 `dlmSmooth` は平滑化分布の平均と分散を計算する。この関数の引数には、データベクトル（もしくは行列）と特定の `dlm` クラスのオブジェクトを一緒に与えるか、もしくは、`dlmFilter` によって生成された「フィルタ化された DLM」(filtered DLM) オブジェクトを与える事ができる。以下の `R` のコードでは、ナイル川のデータの例でこの関数の使用法を示している。

```
> nileJumpSmooth <- dlmSmooth(nileJumpFilt)
> plot(Nile, type = 'o', col = "seagreen")
> attach(nileJumpSmooth)
> lines(dropFirst(s), type = 'o', pch = 20, col = "brown")
> v <- unlist(dlmSvd2var(U.S, D.S))
> pl <- dropFirst(s) + qnorm(0.05, sd = sqrt(v[-1]))
> pu <- dropFirst(s) + qnorm(0.95, sd = sqrt(v[-1]))
> detach()
> lines(pl, lty = 2, col = "brown")
> lines(pu, lty = 2, col = "brown")
```

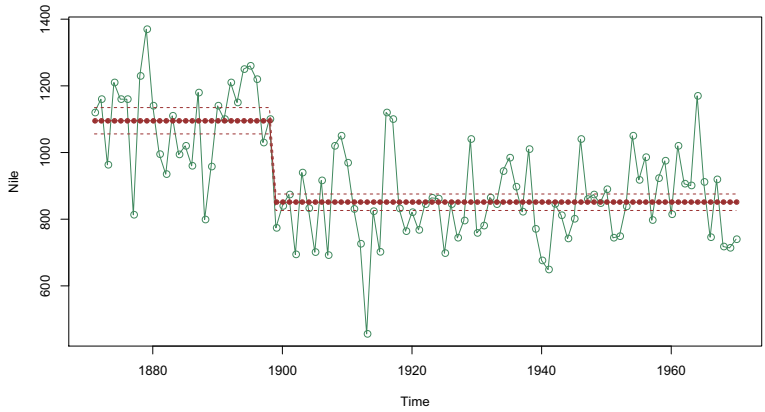

図 2　ナイル川のデータと平滑化レベル

第二の例として、イギリスのガス消費量に関するデータセットを考えよう。このデータを対数スケールにすれば、四半期毎の季節成分とローカル線型トレンド (和分ランダムウォークの形態) をあわせた DLM によって、合理的にモデル化が可能となる。まず、最尤法で未知の分散を推定する。

```
> lGas <- log(UKgas)
> dlmGas <- dlmModPoly() + dlmModSeas(4)
> buildFun <- function(x) {
+     diag(W(dlmGas))[2:3] <- exp(x[1:2])
+     V(dlmGas) <- exp(x[3])
+     return(dlmGas)
+ }
> (fit <- dlmMLE(lGas, parm = rep(0, 3), build = buildFun))$conv

[1] 0

> dlmGas <- buildFun(fit$par)
> drop(V(dlmGas))
```

```
[1] 0.00182
```

```
> diag(W(dlmGas))[2:3]
```

```
[1] 7.90e-06 3.31e-03
```

あてはめを行ったモデルに基づき、状態の平滑化推定値を計算することができる。この値を用いて、測定誤差がある中で得たデータを、なだらかなトレンド成分と確率的な季節成分に分解する事ができる。

```
> gasSmooth <- dlmSmooth(lGas, mod = dlmGas)
> x <- cbind(lGas, dropFirst(gasSmooth$s[,c(1,3)]))
> colnames(x) <- c("Gas", "Trend", "Seasonal")
> plot(x, type = 'o', main = "UK Gas Consumption")
```

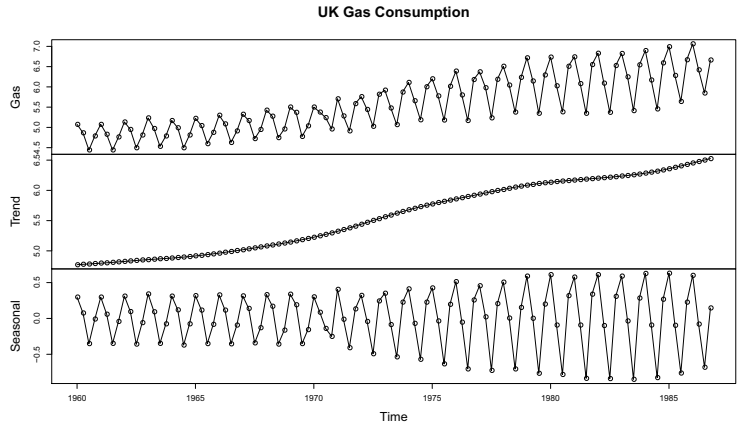

図 3　ガス消費量と「トレンド + 季節」(trend+seasonal) による分解

3.3 予測

状態や観測値の予測分布に関する平均と分散は、以下のコードで説明するように関数 dlmForecast によって得られる。将来の状態や観測値に関する平均と分散はリストとして返され、その要素は a, R, f, Q となる。

```
> gasFilt <- dlmFilter(lGas, mod = dlmGas)
> gasFore <- dlmForecast(gasFilt, nAhead = 20)
> sqrtR <- sapply(gasFore$R, function(x) sqrt(x[1,1]))
> pl <- gasFore$a[,1] + qnorm(0.05, sd = sqrtR)
> pu <- gasFore$a[,1] + qnorm(0.95, sd = sqrtR)
> x <- ts.union(window(lGas, start = c(1982, 1)),
+               window(gasSmooth$s[,1], start = c(1982, 1)),
+               gasFore$a[,1], pl, pu)
> plot(x, plot.type = "single", type = 'o', pch = c(1, 0, 20, 3, 3),
```

```
+       col = c("darkgrey", "darkgrey", "brown", "yellow", "yellow"),
+       ylab = "Log gas consumption")
> legend("bottomright", legend = c("Observed",
+                          "Smoothed (deseasonalized)",
+                          "Forecasted level", "90% probability limit"),
+         bty = 'n', pch = c(1, 0, 20, 3, 3), lty = 1,
+         col = c("darkgrey", "darkgrey", "brown", "yellow", "yellow"))
```

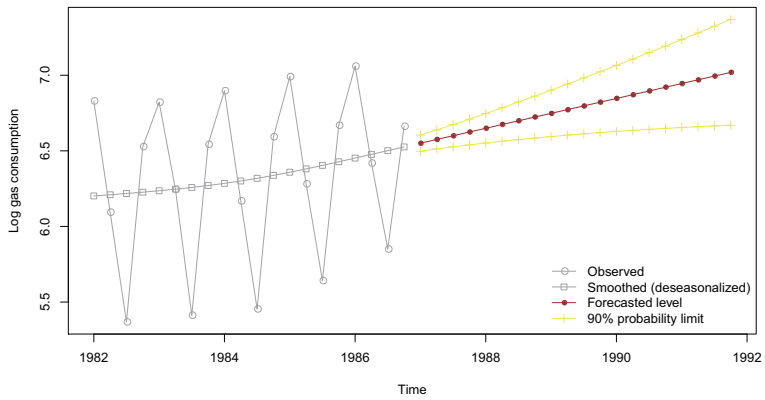

図 4　ガス消費量の予測

4　動的線型モデルのベイズ分析

DLM を定義するパラメータが全て既知であれば、ベイズ分析を実行するために必要なものは、前節で説明した平滑化や予測のための関数が全てとなる。このことは、少なくとも観測不可能な状態や将来の観測値/状態に関する事後推定値に関心がある場合には、事実である。しかし現実の世界におけるほとんどすべての応用では、DLM の特定化に未知パラメータが一つ以上は含まれ、推定を行う必要が生じる。いくつかの非常に単純なモデルや特別な事前分布が適用できる場合を除くと、未知パラメータの事後分布 (もしくは、未知パラメータと状態の同時事後分布) は単純な形とはならないので、その事後分布からの標本を生成するためにはマルコフ連鎖モンテカルロ (Markov chain Monte Carlo; MCMC) 法を用いるのが一般的なアプローチとなる[1]。MCMC の動作は、同じ種類の DLM であってもモデルや事前分布に強く依存するため、あらゆる場合にうまく機能する一般的なアルゴリズムや関数一式といったものを提供することはできない。しかしながらパッケージ `dlm` では、MCMC による事後分布のシミュレーションを容易にするために、いくつかの関数が提供されている。さらにこのパッケージでは、サンプラの出力を分析するために最低限のツール類も提供されている。

4.1　前向きフィルタ後ろ向きサンプリング

パラメータと観測不可能な状態の同時事後分布から標本を得る方法の一つはギブスサンプラを動作させることであり、そこでは状態に関する全条件付分布からの抽出と、パラメータに関する全条件付分布からの

[1] 逐次モンテカルロ法は MCMC の代替手段を提供するが、ここでは説明をしない。

抽出が繰り返される。パラメータの生成はモデルに依存するが、状態に関する全条件付分布からの標本は、[CK, FS, S] によって独立に開発されたいわゆる前向きフィルタ後ろ向きサンプリング（Forward Filtering Backward Sampling; FFBS）アルゴリズムを用いた一般的な方法によって、容易に得ることができる。このアルゴリズムは、実質的にカルマン平滑化のシミュレーション版からなっている。[DK] にはこの代替アルゴリズムを見つけることができる。ギブスサンプラで状態を生成する際、モデルのパラメータは直近で生成された値に固定されることに注意しよう。このため DLM の特定化が完全になり、問題は観測値が与えられた下での状態に関する条件付き分布からの抽出に還元され、この抽出は DLM の一般的な構造に基づいて効率的に実行される。

関心の対象が状態ではなく未知パラメータのみの場合であっても、多くの場合では状態も残して事後分布を考えた方がギブスサンプラが単純になる。このことが典型的にあてはまるのは、システム方程式やシステム分散に未知パラメータが存在する場合である。これは通常、状態が得られた条件の下であればこれらの未知パラメータをデータから独立にして考えることができるためであり、結果的に全条件付分布が単純になる。この様な枠組みの中でサンプラに状態を含めることは、データ拡大法の一テクニックと考えることができる。

パッケージ `dlm` において、FFBS は関数 `dlmBSample` に実装されている。より正確に言うと、この関数はフィルタ化されたモデルから処理を開始するので、アルゴリズムにおける後ろ向きサンプリングの部分だけを実施している。`dlmBSample` の引数は実際には一つで、`dlmFiltered` オブジェクト、もしくはそう解釈できるリストとなる。

以下のコードでは、観測不可能なナイル川の「真のレベル」(true level) に関する事後分布からシミュレートした実現値を 10 回分生成し、図 5 にプロットしている。ここで、モデルにはランダムウォーク・プラス・ノイズモデルを使用した (2 節で記載したレベルの急変は未考慮)。この例ではモデルのパラメータは MLE で決定している。

```
> plot(Nile, type = 'o', col = "seagreen")
> nileFilt <- dlmFilter(Nile, dlmNile)
> for (i in 1:10) # 10 simulated "true" levels
+     lines(dropFirst(dlmBSample(nileFilt)), col = "brown")
```

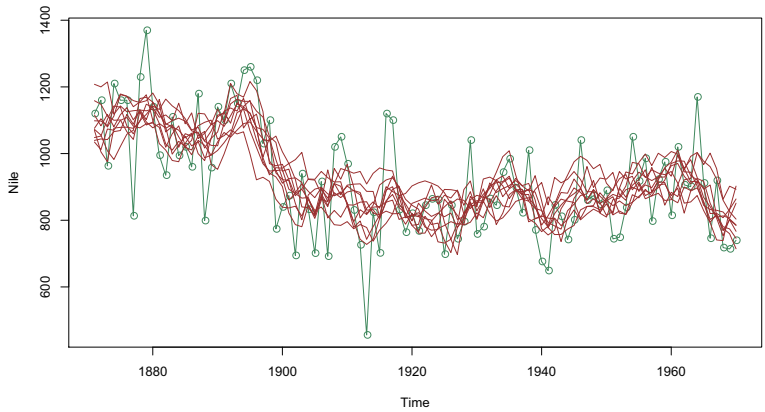

図 5　ナイル川のデータとシミュレートされた**真のレベル**(true level)

この図を見ると、レベルの急変を考慮したモデルを用いた結果とは、大分異なっている印象を受ける！

4.2 適応的棄却メトロポリスサンプリング

Gilks とその共著者らは [GBT] において、棄却サンプリングに基づき、任意の特定の連続分布から確率変数を生成する賢い適応的な手法を開発した。このアルゴリズムでは目標分布の台が有界である必要があるものの、そうでない場合は、実用目的では目標分布を非常に大きい有限区間に制限すればよい。パッケージ `dlm` では関数 `arms` が提供されており、Gilks によって書かれた元の C のコードが移植されている。`arms` の引数は、出発点、二つの関数（ひとつは目標密度の対数を返し、もうひとつは目標密度の台に関する指標を返す）、必要な標本の数となる。対数密度と台の指標に関しては、引数`...` を通じて追加引数を渡すことができる。ヘルプのページにはいくつか例が含まれているが、それらの大部分は DLM には直接関係しないものである。重要な例は、次のように正規分布の混合を目標分布として扱う場合である。次のような目標分布を仮定しよう。

$$f(x) = \sum_{i=1}^{k} p_i \phi(x; \mu_i, \sigma_i)$$

ここで $\phi(\cdot; \mu, \sigma)$ は、平均が μ で分散が σ^2 の正規確率変数に関する密度である。以下のコードは、点 `x` におけるこの目標密度の対数を返す `R` の関数となる。

```
> lmixnorm <- function(x, weights, means, sds) {
+     log(crossprod(weights, exp(-0.5 * ((x - means) / sds)^2
+                                - log(sds))))
+ }
```

平均や標準偏差だけでなく、重み p_i もこの関数に対する追加引数となることに注意しよう。密度の台が実数直線全体であるので、適度に広い区間を「実用的な台」(practical support) として用いる。

```
> y <- arms(0, myldens = lmixnorm,
+           indFunc = function(x,...) (x > (-100)) * (x < 100),
+           n = 5000, weights = c(1, 3, 2),
+           means = c(-10, 0, 10), sds = c(7, 5, 2))
> summary(y)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  -34.4    -3.9     2.3     1.6     9.0    18.2

> library(MASS)
> truehist(y, prob = TRUE, ylim = c(0, 0.08), bty = 'o')
> curve(colSums(c(1, 3, 2) / 6 *
+               dnorm(matrix(x, 3, length(x), TRUE),
+                     mean = c(-10, 0, 10), sd = c(7, 5, 2))),
+       add = TRUE)
> legend(-25, 0.07, "True density", lty = 1, bty = 'n')
```

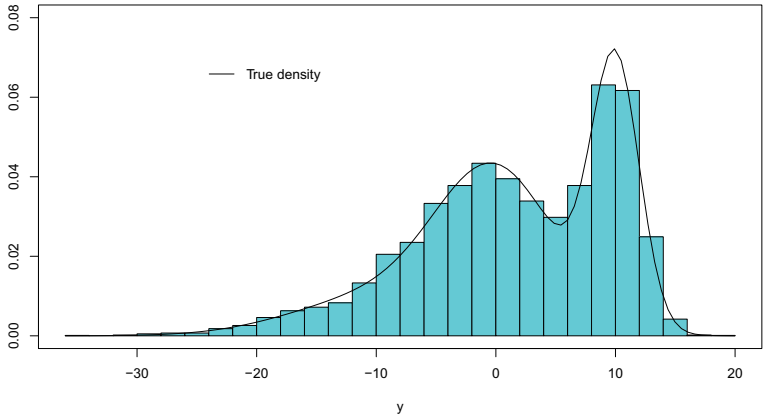

図 6　3 つの正規分布の混合

関数 arms において、多変量の目標密度から標本が生成されるように拡張ができると有益である。このような標本を得るためには、出発点か現在の点を通る直線をランダムに生成し、選ばれた直線に沿って一変量の ARMS アルゴリズムを適用すれば良い。ヘルプのページには、この種の応用に関する例がいくつか含まれている。

4.3　ギブスサンプリング：その一例

本節では、未知の分散を伴う DLM に関するギブスサンプラの単純な例を提示する。ここで提示する形のモデルは多くの応用にかなり共通しており、d **個の逆ガンマ**(d-inverse-gamma) モデルと呼ばれる事もある。

改めてイギリスのガス消費量に関するデータを考えよう。ここでは、ローカル線型トレンドと季節成分の和が雑音を伴って観測されるモデルを想定する。このモデルは、次のような観測されない状態に基づく事になる。

$$\theta_t = \left(\mu_t \ \beta_t \ s_t^{(1)} \ s_t^{(2)} \ s_t^{(3)}\right)'$$

ここで、μ_t は現在のレベル、β_t はトレンドの傾き、$s_t^{(1)}, s_t^{(2)}, s_t^{(3)}$ は季節成分（各々、現四半期、前四半期、前々四半期に相当）である。時点 t における観測値は以下のように与えられる。

$$y_t = \mu_t + s_t^{(1)} + v_t, \qquad v_t \sim \mathcal{N}(0, \sigma^2)$$

観測不可能な状態に関して、次のような動的特性を仮定する。

$$\begin{aligned}
\mu_t &= \mu_{t-1} + \beta_{t-1} \\
\beta_t &= \beta_{t-1} + w_t^{\beta}, && w_t^{\beta} \sim \mathcal{N}(0, \sigma_\beta^2) \\
s_t^{(1)} &= -s_{t-1}^{(1)} - s_{t-1}^{(2)} - s_{t-1}^{(3)} + w_t^{s}, && w_t^{s} \sim \mathcal{N}(0, \sigma_s^2) \\
s_t^{(2)} &= s_{t-1}^{(1)} \\
s_t^{(3)} &= s_{t-1}^{(2)}
\end{aligned}$$

DLM の観点でモデルを表現すると、上記の仮定は以下の関係を示唆している。

$$\begin{aligned}
V &= [\sigma^2] \\
W &= \mathrm{diag}(0, \sigma_\beta^2, \sigma_s^2, 0, 0)
\end{aligned}$$

このモデルの詳細に関しては、[WH] を参照されたい。以上から未知パラメータは、3 つの分散 $\sigma^2, \sigma_\beta^2, \sigma_s^2$ となる。これらの逆数、すなわち 3 つの精度に対して、各々の平均が $a, a_{\theta,2}, a_{\theta,3}$ で分散が $b, b_{\theta,2}, b_{\theta,3}$ の独立なガンマ事前分布を仮定する。

ギブスサンプラは、以下の全条件付き分布に基づいて実行できる（潜在変数として観測不可能な状態を加え、簡単な計算を行えば分かる）。

$$\begin{aligned}
\theta_{0:n} &\sim \mathcal{N}() \\
\sigma^2 &\sim \mathcal{IG}\left(\frac{a^2}{b} + \frac{n}{2}, \frac{a}{b} + \frac{1}{2}SS_y\right) \\
\sigma_\beta^2 &\sim \mathcal{IG}\left(\frac{a_{\theta,2}^2}{b_{\theta,2}} + \frac{n}{2}, \frac{a_{\theta,2}}{b_{\theta,2}} + \frac{1}{2}SS_{\theta,2}\right) \\
\sigma_s^2 &\sim \mathcal{IG}\left(\frac{a_{\theta,3}^2}{b_{\theta,3}} + \frac{n}{2}, \frac{a_{\theta,3}}{b_{\theta,3}} + \frac{1}{2}SS_{\theta,3}\right)
\end{aligned}$$

ここで、

$$\begin{aligned}
SS_y &= \sum_{t=1}^{n}(y_t - F_t\theta_t)^2 \\
SS_{\theta,i} &= \sum_{t=1}^{n}(\theta_{t,i} - (G_t\theta_{t-1})_i)^2, \qquad i = 2, 3
\end{aligned}$$

である。状態に関する全条件付き分布は、ある平均と分散を持つ正規分布になるがこれを明示的に導出する必要はない。というのも、`dlmBSample` が適切な分布から $\theta_{0:n}$ を生成してくれるためである。

関数 `dlmGibbsDIG` は他の関数よりも教育的な理由でこのパッケージに含まれているが、その中では上記で説明されているような全条件付分布に基づくギブスサンプラが実装されている。このサンプラを実行させる R のコード（一部）は、次のようになる。

```
> outGibbs <- dlmGibbsDIG(lGas, dlmModPoly(2) + dlmModSeas(4),
+                         a.y = 1, b.y = 1000, a.theta = 1,
+                         b.theta = 1000,
+                         n.sample = 1100, ind = c(2, 3),
+                         save.states = FALSE)
```

最初の 100 回分をバーンインとして破棄すると、シミュレートされた値と移動エルゴード平均のプロットが次のように得られる（図 7 を参照）。

```
> burn <- 100
> attach(outGibbs)
> dV <- dV[-(1:burn)]
> dW <- dW[-(1:burn), ]
> detach()
> par(mfrow=c(2,3), mar=c(3.1,2.1,2.1,1.1))
> plot(dV, type = 'l', xlab = "", ylab = "",
+      main = expression(sigma^2))
```

```
> plot(dW[ , 1], type = 'l', xlab = "", ylab = "",
+      main = expression(sigma[beta]^2))
> plot(dW[ , 2], type = 'l', xlab = "", ylab = "",
+      main = expression(sigma[s]^2))
> use <- length(dV) - burn
> from <- 0.05 * use
> at <- pretty(c(0, use), n = 3); at <- at[at >= from]
> plot(ergMean(dV, from), type = 'l', xaxt = 'n',
+      xlab = "", ylab = "")
> axis(1, at = at - from, labels = format(at))
> plot(ergMean(dW[ , 1], from), type = 'l', xaxt = 'n',
+      xlab = "", ylab = "")
> axis(1, at = at - from, labels = format(at))
> plot(ergMean(dW[ , 2], from), type = 'l', xaxt = 'n',
+      xlab = "", ylab = "")
> axis(1, at =  at - from, labels = format(at))
```

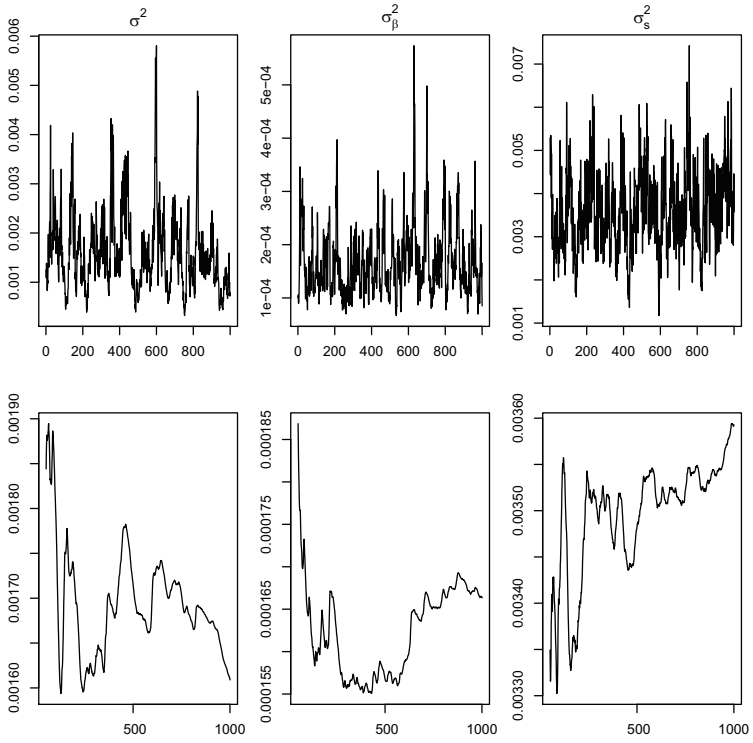

図 7　軌跡のプロット (上) と移動エルゴード平均 (下)

関数 mcmcMean を用いると、ギブスサンプラの出力から、未知の分散 3 つ分に関する事後推定値とモンテカルロ標準誤差を得ることができる。

```
> mcmcMean(cbind(dV[-(1:burn)], dW[-(1:burn), ]))

                W.2         W.3
  1.59e-03   1.67e-04   3.61e-03
 (1.05e-04) (6.11e-06) (9.33e-05)
```

5 結びの言葉

パッケージ dlm の主な機能について説明し、例証を行ってきた。このパッケージはベイズ流の MCMC に基づく応用を想定して開発されたが、最尤推定やカルマン・フィルタリング/カルマン平滑化にも適用することができる。主な設計目標は、フィルタリングや平滑化および尤度評価のアルゴリズムに関する柔軟性と数値安定性であった。この 2 つの目標には若干関連がある。というのも、一般的な DLM に関して素朴にカルマン・フィルタを実装しただけでは、数値安定性に関して悪影響を受けることが知られているためである。したがって、どんな種類の DLM でも自由に規定できる環境では、前述した不安定性の問題を極力回避しようとする試みが重要であった。このパッケージで用いられたアルゴリズムは、分散行列を SVD の形で逐次的に評価する方法に基づいている。これは平方根フィルタ (もしくは平滑化器) の一形態と考えることができるが、コレスキー分解の伝播に基づく標準的な平方根フィルタよりはるかに頑健である。実際 SVD に基づくアルゴリズムでは、行列 W が正則である必要すらない (ただし、V は非特異である必要がある)。

このパッケージではベイズ推定に関して、任意の一変量もしくは多変量の DLM に関するギブスサンプラの実装を容易にするためのツールが提供されている。関数 dlmBSample(状態を生成) や arms(ARMS の多変量拡張版) を単独もしくは組み合わせて使う事でギブスサンプラが得られ、このギブスサンプラにより様々な種類のモデルや事前分布に対して、ベイズ流の事後分布に関する推定が実行できる様になる。

参考文献

[CK] Carter, C.K. and Kohn, R. (1994). On Gibbs sampling for state space models. *Biometrika*, 81.

[DK] Durbin, J. and Koopman, S.J. (2001). *Time Series analysis by state space methods*. Oxford University Press.

[DKj] J. ダービン・S.J. クープマン著. 和合肇・松田安昌訳. (2004). **状態空間モデリングによる時系列分析入門**. シーエーピー出版.

[FS] Früwirth-Schnatter, S. (1994). Data augmentation and dynamic linear models. *journal of Time Series Analysis*, 15.

[GBT] Gilks, W.R., Best, N.G. and Tan, K.K.C. (1995). Adaptive rejection Metropolis sampling within Gibbs sampling (Corr: 97V46 p541-542 with Neal, R.M.), *Applied Statistics*, 44.

[H] Harvey, A.C. (1989). *Forecasting, Structural Time Series Models, and the Kalman Filter*. Cambridge University Press.

[S] Shephard, N. (1994). Partial non-Gaussian state space models. *Biometrika*, 81.

[WH] West, M. and Harrison, J. (1997). *Bayesian forecasting and dynamic models*. (Second edition. First edition: 1989), Springer, N.Y.

[ZL] Zhang, Y. and Li, R. (1996). Fixed-interval smoothing algorithm based on singular value decomposition. *Proceedings of the 1996 IEEE international conference on control applications*.