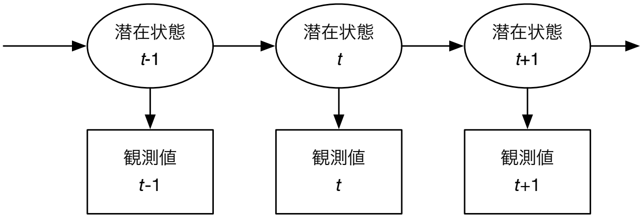
Stan と dlm による状態空間モデル[1]

伊東宏樹

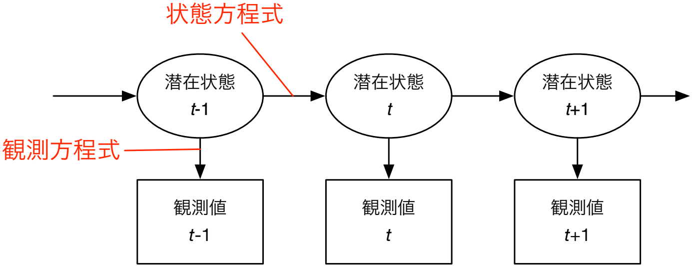
2016-10-29 SappoRo.R #7

[1] ファイル一式は https://github.com/ito4303/SappoRoR7 においておきます

状態空間モデル



状態空間モデル



動的線形モデル

線形・正規分布の状態空間モデル

- 観測方程式

$$\mathbf{y}_t \sim N(\mathbf{F}'\boldsymbol{\theta}_t, \mathbf{V})$$

- 状態方程式

$$\boldsymbol{\theta}_t \sim N(\mathbf{G}\boldsymbol{\theta}_{t-1}, \mathbf{W})$$

- 初期値

$$\boldsymbol{\theta}_0 \sim N(\mathbf{m}_0, \mathbf{C}_0)$$

ローカルレベルモデル

1 階差分の動的線形モデル

- 観測方程式

$$y_t = \theta_t + v_t, \quad v_t \sim N(0, \sigma_v^2)$$
$$\downarrow$$
$$F = 1, V = \sigma_v^2$$

- 状態方程式

$$\theta_t = \theta_{t-1} + w_t, \quad w_t \sim N(0, \sigma_w^2)$$
$$\downarrow$$
$$G = 1, W = \sigma_w^2$$

Stan

- 統計的プログラミング言語
  - MCMC (HMC, NUTS) によるベイズ推定
- 観測方程式と状態方程式をそのまま Stan コードにすれば、状態の推定ができる
  - 『岩波データサイエンス Vol.1』や『Stan と R でベイズ統計モデリング』の松浦さんのコードを参照
- 今回はあえて、Stan の `gaussian_dlm_obs()` 分布と、R の dlm パッケージを使ってみる

gaussian_dlm_obs

y ~ **gaussian_dlm_obs**(F, G, V, W, $m0$, $C0$);

- カルマンフィルタのパラメータ (分散) を推定する
- 引数は dlm パッケージに対応

カルマンフィルタ

- 行列計算により、フィルタリング・平滑化・予測をおこなう[2]
  - **フィルタリング** $\{y_1, y_2, \ldots, y_t\}$ から θ_t を推定
  - **平滑化** $\{y_1, y_2, \ldots, y_t\}$ から $\{\theta_1, \theta_2, \ldots, \theta_t\}$ を推定
  - **予測** $\{y_1, y_2, \ldots, y_t\}$ から $\{\theta_{t+1}, \theta_{t+2}, \ldots\}$ および $\{y_{t+1}, y_{t+2}, \ldots\}$ を推定
- パラメータとして、観測方程式・状態方程式の共分散行列（と初期値）が必要

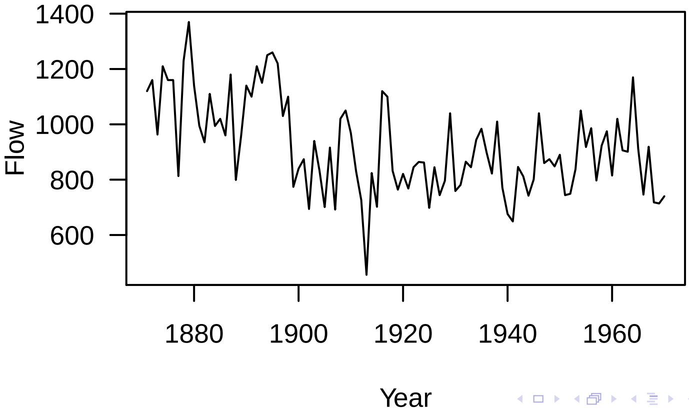
[2]具体的な計算式は、野村俊一『カルマンフィルタ—R を使った時系列予測と状態空間モデル』共立出版, 2016 などを参照

dlm パッケージ

- 動的線形モデル (Dynamic Linear Model) を扱うパッケージ
- カルマンフィルター
- パラメータの最尤推定
- パラメータのベイズ推定

データ

```
## ナイル川の流量データ
data(Nile)

```



Stan コード: data ブロック

```
data {
  int<lower=0>  N;    // データ点の数
  matrix[1, N]  Y;    // データ
  real          M0;   // 状態の初期値
  cov_matrix[1] C0;   // 共分散の初期値
}

```

Stan コード: transformed data ブロック

```
transformed data {
  matrix[1, 1]  F;
  matrix[1, 1]  G;
  vector[1]     m0; // サイズ 1のベクトル

  F[1, 1] = 1;
  G[1, 1] = 1;
  m0[1] = M0;
}
```

Stan コード: parameters ブロックと transformed parameters ブロック

```
parameters {
  real<lower=0> s2[2];
}

transformed parameters {
  vector[1]     v;
  cov_matrix[1] w;

  v[1] = s2[1];
  w[1, 1] = s2[2];
}

```

Stan コード: model ブロック

```
model {
  Y ~ gaussian_dlm_obs(F, G, v, w, m0, C0);
}
```

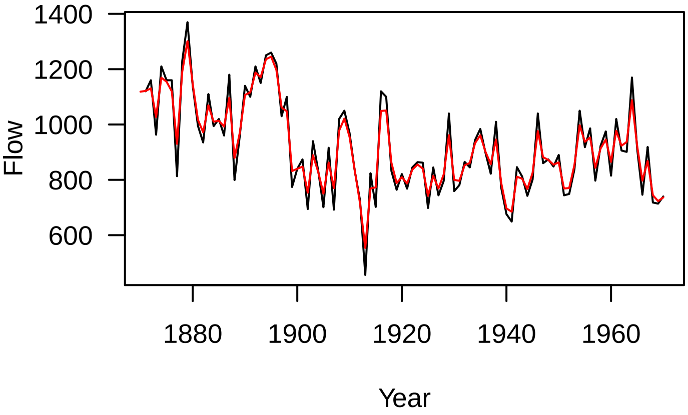
R コード

```
## Stan にわたすデータのリスト
stan_data <- list(N = length(Nile),
                  Y = matrix(Nile, 1),
                  M0 = 100,
                  C0 = matrix(100, 1, 1))

## あてはめ
fit <- stan("dlm1.stan", data = stan_data,
            pars = c("s2"), seed = 1,
            iter = 2000, warmup = 1000)
```

平滑化

```
## パラメータの事後平均をとりだす
s2 <- get_posterior_mean(fit, pars = "s2")[, "mean-all chains"]

## Stan で推定したパラメータで、dlm のモデル定義
mod <- dlmModPoly(order = 1,
                  dV = s2[1], dW = s2[2])

## 平滑化
smo <- dlmSmooth(Nile, mod)
```

平滑化

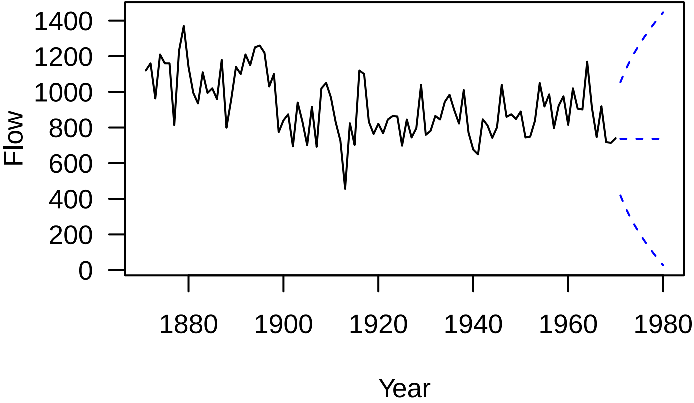


赤線が平滑化した値

予測の R コード

```
## カルマンフィルター
filt <- dlmFilter(Nile, mod)

## 状態
m <- filt$m[-1]

## 予測
## 予測する年数
nyear <- 10

## 予測期間の初期値の設定
m0(mod) <- tail(m, 1)
C0(mod) <- diag(1) * s2[2]

fore <- dlmForecast(mod, nAhead = nyear)

```

予測



青実線が予測値，青点線は 80%予測区間

トレンドモデル

傾きを組み込んだモデル、あるいは 2 階差分の動的線形モデル

- 観測方程式

$$
\begin{aligned}
&y_t = \theta_{1,t} + v_t \\
&\downarrow \\
&y_t = \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} \theta_{1,t} \\ \theta_{2,t} \end{pmatrix} + v_t \\
&\downarrow \\
&\mathbf{F} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, V = \sigma_v^2
\end{aligned}
$$

トレンドモデル

- 状態方程式

$$\theta_{1,t} = \theta_{1,t-1} + \theta_{2,t-1} + w_{1,t}$$
$$\theta_{2,t} = \theta_{2,t-1} + w_{2,t}$$
$$\downarrow$$
$$\begin{pmatrix} \theta_{1,t} \\ \theta_{2,t} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \theta_{1,t-1} \\ \theta_{2,t-1} \end{pmatrix} + \begin{pmatrix} w_{1,t} \\ w_{2,t} \end{pmatrix}$$
$$\downarrow$$
$$\mathbf{G} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \mathbf{W} = \begin{pmatrix} \sigma_{w1}^2 & 0 \\ 0 & \sigma_{w2}^2 \end{pmatrix}$$

data ブロック

```
data {
  int<lower=0>  N;  // データ点の数
  matrix[1, N]  Y;  // データ
  vector[2]     M0;
  cov_matrix[2] C0;
}
```

transformed data ブロック

```
transformed data {
  matrix[2, 1]  F;
  matrix[2, 2]  G;

  // F
  F[1, 1] = 1;
  F[2, 1] = 0;

  // G
  G[1, 1] = 1;
  G[1, 2] = 1;
  G[2, 1] = 0;
  G[2, 2] = 1;
}

```

parameters ブロックと transformed parameters ブロック

```
parameters {
  real<lower=0> s2[3];
}

transformed parameters {
  vector[1]      v;
  cov_matrix[2] w;

  v[1] = s2[1];
  w[1, 1] = s2[2];
  w[1, 2] = 0;
  w[2, 1] = 0;
  w[2, 2] = s2[3];
}
```

model ブロック

```
model {
  Y ~ gaussian_dlm_obs(F, G, v, w, M0, C0);
}
```

R コード

```
## Stan にわたすデータのリスト
stan_data <- list(N = length(Nile),
                  Y = matrix(Nile, 1),
                  M0 = c(100, 1),
                  C0 = matrix(c(1e7, 0, 0, 1e7), 2, 2))

## あてはめ
fit <- stan("dlm2.stan", data = stan_data,
            pars = c("s2"), seed = 1,
            iter = 2000, warmup = 1000)
```

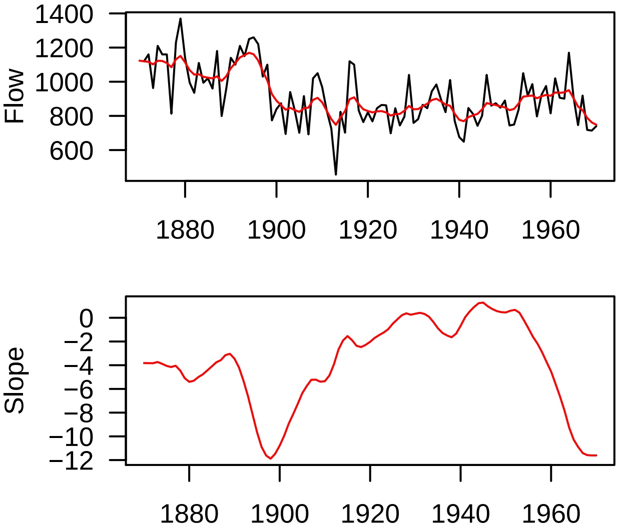
R コード

```r
## パラメータの事後平均をとりだす
s2 <- get_posterior_mean(fit, pars = "s2")[, "mean-all chains"]

## dlm でのモデル定義
mod <- dlmModPoly(order = 2,
                  dV = s2[1], dW = s2[2:3])

## 平滑化
smo <- dlmSmooth(Nile, mod)

```

予測の R コード

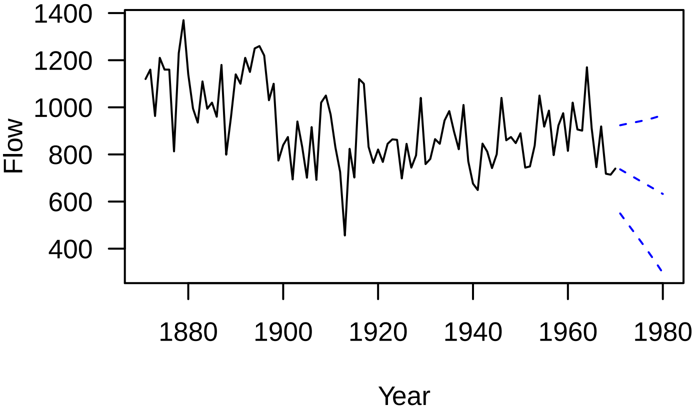
```
## カルマンフィルター
filt <- dlmFilter(Nile, mod)

## 状態
m <- filt$m[-1, ]

## 予測
## 予測する年数
nyear <- 10

## 予測期間の初期値の設定
m0(mod) <- m[length(Nile), ]
C0(mod) <- diag(2) * s2[2:3]

fore <- dlmForecast(mod, nAhead = nyear)

```

平滑化



赤線が平滑化した値

予測



青実線が予測値，青点線は 80%予測区間

まとめ

- Stan の `gaussian_dlm_obs` 分布で動的線形モデルのパラメータを推定することができる。
- 推定したパラメータの値を dlm パッケージの関数で使用してカルマンフィルタを適用できる。

まとめ

- とはいえ、普通に状態をベイズ推定するなら、全部 Stan でモデルを書くほうがてっとりばやいかも。