제12강 : 칼만필터링

금융 통계 및 시계열 분석

TRADE INFORMATIX

2014년 2월 18일

목차

통계적 추정 문제(Statistical Inference Problem)

- 주어진 data X에서 모형 파라미터 θ를 찾아낸다

1. 기존의 추정 방법론 (Batch estimation)
   - X만을 사용하여 파라미터 θ 추정
   - MLE (Maximum Likelihood Estimation)
     - Batch estimation의 방법 중 하나
     - likelihood를 최대화하는 θ를 찾는 최적화 문제
2. 베이지안 추정 (Bayesian estimation)
   - a prior 파라미터 분포 $P(\theta)$와 X을 사용하여 파라미터 분포 $P(\theta|X)$ 추정
   - 파라미터 분포에 대한 가정이 필요
   - Bayesain Rule에 의해 a prior 분포를 a posterior 분포로 변화

베이지안 추정 (Bayeisan Estimation)

- 베이지안 추정 방법론에서는
  추정할 파라미터가 결정론적으로 정해진 값(deterministic value)이 아니라
  확률분포로 기술되기 때문에 그 **파라미터의 분포를 찾아야 한다**고 가정한다.
- 결정론적 추정 방법론 :

$$y \sim N(\mu, 1)$$

  의 샘플데이터 $\{y_i\}$로부터 μ값을 추정 (point estimation)

- 베이지안 추정 방법론 :

$$y \sim N(\mu, 1), \quad \mu \sim N(m, C)$$

  의 샘플데이터 $\{y_i\}$로부터 μ의 분포 파라미터 m와 C를 추정

베이지안 추정의 예 1

- Normal-Normal 문제
  - 샘플 분포가 Normal 분포이고 그 Normal 분포의 평균값도 Normal 분포
  - 샘플 분포의 분산은 주어짐

$$\begin{aligned} P(x|\mu) &= N(\mu, \sigma^2) \\ P(\mu|\mu_0, \sigma_0^2) &= N(\mu_0, \sigma_0^2) \end{aligned}$$

- Normal-InverseGamma 문제
  - 샘플 분포가 Normal 분포
  - Normal 분포의 평균값은 Normal 분포
  - Normal 분포의 분산은 Inverse-Gamma 분포

$$\begin{aligned} P(x|\mu) &= N(\mu, \sigma^2) \\ P(\mu|\mu_0, \tau^2) &= N(\mu_0, \tau^2) \\ P(\sigma^2|a, b) &= \text{InverseGamma}(a, b) \end{aligned}$$

베이지안 추정의 예 2

- Gamma-Poisson 문제
  - 샘플 분포가 Poisson 분포이고 그 Poisson 분포의 파라미터는 Gamma 분포

$$\begin{aligned} P(x|\mu) &= \text{Poisson}(\lambda) \\ P(\lambda|\alpha,\beta) &= \text{Gamma}(\mu_0, \sigma_0^2) \end{aligned}$$

- Beta-Binomial 문제
  - 샘플 분포가 Binomial 분포
  - Binomial 분포의 성공확률 p은 Beta 분포

$$\begin{aligned} P(x|p) &= \text{Binom}(p) \\ P(p|\alpha,\beta) &= \text{Beta}(\alpha,\beta) \end{aligned}$$

베이지안 법칙

- 베이지안 추정 방법론에서는
  샘플 데이터가 추가될 때마다 그 데이터와 베이지안 법칙을 이용하여
  지금까지 구한 파라미터 분포를 갱신 (update) 한다.
- 베이지안 법칙 (Baysian Rule) :

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

  - $P(A)$: 사건 A가 발생할 확률
  - $P(B)$: 사건 B가 발생할 확률
  - $P(A \cap B)$: 사건 A가 사건 B가 동시에 발생할 확률
  - $P(A|B)$: 사건B가 이미 발생했을때 A가 또 발생할 확률

베이지안 법칙의 예

정적 통계 분석

- ❏ 어떤 희귀질병에 걸릴 확률 : 0.1%
- ❏ 감염이 되었을 경우 시약검사 결과가 양성일 확률 : 99%
- ❏ 감염이 되지 않았음에도 시약검사 결과가 양성일 확률 : 2%
- ❏ 어떤 환자의 검사결과가 양성으로 나왔을 경우 실제로 감염되었을 확률은?

- ❏ 질병 감염 확률 : $P(D) = 0.001$
- ❏ True Alarm 확률 : $P(T|D) = 0.99$
- ❏ False Alarm 확률 : $P(T|D^C) = 0.02$
- ❏ 검사결과가 양성일때 실제로 감염되었을 확률

$$P(D|T) = \frac{P(D \cap T)}{P(T)} = \frac{P(T|D)P(D)}{P(T)} = \frac{P(T|D)P(D)}{P(T|D)T(D) + P(T|D^C)T(D^C)}$$

$$= \frac{0.99 \times 0.001)}{0.99 \times 0.001 + 0.02 \times 0.999} = 0.047 = 4.7\%$$

베이지안 분포 추정

❑ 베이지안방법을 이용한 파라미터/상태변수 분포 추정

$$
\begin{aligned}
\pi(\theta|y) &= \frac{\pi(y \cap \theta)}{\pi(y)} \\
&= \frac{\pi(y|\theta)\pi(\theta)}{\pi(y)} \\
&= \frac{\pi(y|\theta)\pi(\theta)}{\int \pi(y|\theta)\pi(\theta)d\theta} \\
\text{updated distribution under } y &= \frac{\text{likelihood} \times \text{a prior distribution}}{\text{marginal distribution}}
\end{aligned}
$$

- updated distribution : 새로운 샘플 y을 이용하여 갱신된 파라미터 분포 추정치
- likelihood : 특정한 파라미터 하에서 그 샘플값이 나올 가능성
- a prior distribution : 지금까지 알고 있던 파라미터 분포 추정치
- marginal distribution : 샘플 y값이 나올 무조건부(unconditional) 분포

마코프 체인 (Markov Chain)

- 어떤 확률적 시계열 y_t $(t = 1, 2, \ldots, t)$의 조건부 확률분포 $\pi(y_t|y_1, \cdots, y_{t-1})$가 한 스텝전의 과거에만 의존하는 경우 **마코프 체인 (Markov Chain)**이라고 한다.

$$\pi(y_t|y_1, \cdots, y_{t-1}) = \pi(y_t|y_{t-1})$$

- 마코프 체인 (Markov Chain)의 결합확률분포 (joint distribution)는 다음과 같은 형태로 표현가능

$$\pi(y_1, \cdots, y_t) = \pi(y_1) \prod_{j=2}^{t} \pi(y_j|y_{j-1})$$

상태공간 모형 (State Space Model)

- ARMA 시계열 모형에서는 관측할 수 있는 출력값 x 자체의 결합확률분포 (joint density distribution)을 정의하여 시계열 정보의 변화를 기술

$$\pi(x_t|x_1,\ldots,x_{t-1}) = \frac{\pi(x_1,\ldots,x_t)}{\pi(x_1,\ldots,x_{t-1})}$$

- **상태공간 모형**에서는 보이지 않은 상태변수(state) θ의 결합확률분포를 정의하여 시계열 정보의 변화를 기술

$$\pi(\theta_t|\theta_1,\ldots,\theta_{t-1}) = \frac{\pi(\theta_1,\ldots,\theta_t)}{\pi(\theta_1,\ldots,\theta_{t-1})}$$

- 상태공간 모형에서 사람이 관측할 수 있는 것은 상태변수 θ 자체가 아닌 상태변수에 의존하는 출력 (measurements) y 라고 가정. 이 때 상태공간 모형에서 상태변수 θ와 출력 y의 관계는 다음과 같은 추가적인 조건부 확률분포에 의해 정의됨

$$\pi(y_t|\theta_t)$$

- 상태변수는 베이지안 추정방법에서 추정하려는 파라미터에 해당

조건부 독립성 (Conditional Independence)

❏ 상태공간 모형을 정의하기 위해서는 다음 조건이 필요

1. 상태변수가 마코프 체인
2. 출력y는 조건부 독립 : θ에 조건부로 서로 독립적이고 θ에만 의존

$$\pi(y_1, \ldots, y_t|\theta) = \prod_{i=1}^{n} \pi(y_i|\theta)$$

❏ 상태변수와 출력의 전체 분포는

- ▶ 최초의 상태변수 분포 $\pi(\theta_0)$와
- ▶ 조건부 분포 $\pi(\theta_i|\theta_{i-1})$, $\pi(y_i|\theta_i)$

만으로 기술 가능

$$\pi(\theta_0, \ldots, \theta_t, y_1, \ldots, y_t) = \pi(\theta_0) \prod_{i=1}^{n} \pi(\theta_i|\theta_{i-1})\pi(y_i|\theta_i)$$

필터링 (Filtering), 스무딩 (Smoothing), 예측 (Forecasting)

- 필터링
  - 최신 상태변수의 최신 업데이트 분포 $\pi(\theta_t|y_1, \ldots, y_t)$ 계산
- 스무딩
  - 과거 상태변수의 최신 업데이트 분포 $\pi(\theta_t|y_1, \ldots, y_T)$ 계산
- 예측
  - 미래 상태변수의 최신 업데이트 분포 $\pi(\theta_{t+k}|y_1, \ldots, y_t)$ 계산

베이지안 회귀 필터링 (Bayesian Recursive Filtering)

1. 초기 파라미터 분포

$$\theta_0 \sim N(m_0, C_0)$$

2. 1-스텝 파라미터 예측

$$\pi(\theta_t|y_1, \ldots, y_{t-1}) = \int \pi(\theta_t|\theta_{t-1})\pi(\theta_{t-1}|y_1, \ldots, y_{t-1})d\theta_{t-1}$$

3. 1-스텝 출력 예측

$$\pi(y_t|y_1, \ldots, y_{t-1}) = \int \pi(y_t|\theta_t)\pi(\theta_t|y_1, \ldots, y_{t-1})d\theta_t$$

4. 파라미터 추정

$$\pi(\theta_t|t_1, \ldots, y_t) = \frac{\pi(y_t|\theta_t)\pi(\theta_t|y_1, \ldots, y_{t-1})}{\pi(y_t|y_1, \ldots, y_{t-1})}$$

베이지안 회귀 스무딩 (Bayesian Recursive Smoothing)

1. 1-스텝 과거 파라미터 추정

$$\pi(\theta_t|\theta_{t+1}, y_1, \ldots, y_{t-1}) = \frac{\pi(\theta_{t+1}|\theta_t)\pi(\theta_t|y_1, \ldots, y_t)}{\pi(\theta_{t+1}|y_1, \ldots, y_t)}$$

2. 스무딩

$$\pi(\theta_t|y_1, \ldots, y_T) = \pi(\theta_t|y_1, \ldots, y_t) \int \frac{\pi(\theta_{t+1}|\theta_t)}{\pi(\theta_{t+1}|y_1, \ldots, y_t)} \pi(\theta_{t+1}|y_1, \ldots, y_T) d\theta_{t+1}$$

베이지안 회귀 예측 (Bayesian Recursive Forecasting)

1. k-스텝 파라미터 예측

$$\pi(\theta_{t+k}|y_1,\ldots,y_{t-1}) = \int \pi(\theta_{t+k}|\theta_{t+k-1})\pi(\theta_{t+k-1}|y_1,\ldots,y_t)d\theta_{t+k-1}$$

2. k-스텝 출력 예측

$$\pi(y_{t+k}|y_1,\ldots,y_t) = \int \pi(y_{t+k}|\theta_{t+k})\pi(\theta_{t+k}|y_1,\ldots,y_t)d\theta_{t+k}$$

칼만필터 모형 가정

❏ 초기 파라미터 분포

$$\theta_0 \sim N(m_0, C_0)$$

❏ 다이나믹스

$$\theta_t \quad = \quad G_t\theta_{t-1} + w_t$$

❏ 출력관계

$$Y_t \quad = \quad F_t\theta_t + v_t$$

- θ_t : 시간 t에서의 파라미터의 기대값 벡터 $(p \times 1)$
- m_0 : 시간 0에서의 초기 파라미터 분포의 기대값 벡터 $(p \times 1)$
- C_0 : 시간 0에서의 초기 파라미터 분포의 공분산 행렬 $(p \times p)$
- G_t : 시간 t에서의 다이나믹스 행렬 $(p \times p)$
- $w_t \sim N(0, W_t)$: 시간 t에서의 이노베이션 $(p \times 1)$
- Y_t : 시간 t에서의 출력값 벡터 $(m \times 1)$
- F_t : 시간 t에서의 출력 행렬 $(m \times p)$
- $v_t \sim N(0, V_t)$: 시간 t에서의 출력 잡음 $(m \times 1)$

칼만 필터링, 칼만 스무딩, 칼만 예측

- 칼만 필터링
  - 최신 상태변수의 최신 추정치 $m_t = E(\theta_t|y_1, \ldots, y_t)$ 계산
- 칼만 스무딩
  - 과거 상태변수의 최신 추정치 $s_t = E(\theta_t|y_1, \ldots, y_T)$ 계산
- 칼만 예측
  - 미래 상태변수의 최신 추정치 $f_{t+k} = E(\theta_{t+k}|y_1, \ldots, y_t)$ 계산

칼만 필터링 (Kalman Filtering)

1. 초기 파라미터 분포

$$\pi(\theta_{t-1}|y_1,\ldots,y_{t-1}) \sim N(m_{t-1}, C_{t-1})$$

2. 1-스텝 파라미터 예측

$$\begin{aligned} a_t &= E(\theta_t|y_1,\ldots,y_{t-1}) = G_t m_{t-1} \\ R_t &= \mathrm{Var}(\theta_t|y_1,\ldots,y_{t-1}) = G_t C_{t-1} G_t^T + W_t \end{aligned}$$

3. 1-스텝 출력 예측

$$\begin{aligned} f_t &= E(y_t|y_1,\ldots,y_{t-1}) = F_t a_t \\ Q_t &= \mathrm{Var}(y_t|y_1,\ldots,y_{t-1}) = F_t R_t F_t^T + V_t \end{aligned}$$

4. 파라미터 추정

$$\begin{aligned} m_t &= E(\theta_t|y_1,\ldots,y_t) = a_t + R_t F_t^T Q_t^{-1}(y_t - f_t) \\ C_t &= \mathrm{Var}(\theta_t|y_1,\ldots,y_t) = R_t - R_t F_t^T Q_t^{-1} F_t R_t \end{aligned}$$

칼만 스무딩 (Kalman Smoothing)

1. 초기 파라미터 분포

$$\pi(\theta_t|y_1,\ldots,y_T) \sim N(s_t, S_t)$$

2. 스무딩

$$\begin{aligned} s_t &= m_t + C_t G_{t+1}^T R_{t+1}^{-1}(s_{t+1} - a_{t+1}) \\ S_t &= C_t - C_t G_{t+1}^T R_{t+1}^{-1}(R_{t+1} - S_{t+1}) R_{t+1}^{-1} G_{t+1} C_t \end{aligned}$$

칼만 예측 (Kalman Forecasting)

1. k-스텝 파라미터 예측

$$\begin{aligned} a_t(k) &= G_{t+k} a_{t-1,k-1} \\ R_t(k) &= G_{t+k} R_t G_{t+k}^T + W_{t+k} \end{aligned}$$

2. k-스텝 출력 예측

$$\begin{aligned} f_t(k) &= F_{t+k} a_t(k) \\ Q_t(k) &= F_{t+k} R_t(k) F_{t+k}^T + V_t \end{aligned}$$

R에서 칼만 필터링, 스무딩, 예측

- dlm 패키지 사용
- 모델 생성 : `dlm(m0, C0, FF, V, GG, W)`
  - `m0` : θ의 초기 분포의 기대값 m_0
  - `C0` : θ의 초기 분포의 분산 C_0
  - `FF, GG` : 출력 행렬 및 다이나믹스 행렬
  - `V, W` : 출력잡음 및 이노베이션 분산
  - 실제로는 `dlmModXXX` 함수를 사용
- 필터링 : `dlmFilter(y, mod)`
  - `y` : 데이터 벡터
  - `mod` : `dlm` 등으로 생성한 모델 오브젝트
- 스무딩 : `dlmSmooth(y, mod)`
  - `y` : 데이터 벡터
  - `mod` : `dlm` 등으로 생성한 모델 오브젝트
- 예측 : `dlmForecast(mod, nAhead)`
  - `mod` : `dlmFilter` 등으로 생성한 필터링 결과
  - `nAhead` : 예측 스텝수

Convenience Models

- 실제로는 `dlm` 명령을 이용하여 직접 행렬을 입력하기 보다는 다음과 같은 모델생성 명령어를 사용
- 모형은 + 기호를 이용하여 조합 (state augmentation) 가능
  - `dlmModPoly` : 다항식 모형
  - `dlmModARMA` : ARMA 모형
  - `dlmModReg` : 회귀분석 모형
  - `dlmModSeas` : 계절성 모형
  - `dlmModTrig` : 주파수 모형

Polynomial Model

❑ 다항식 (polynomial) 을 이용한 추세 (trend) 모형

$$E(y_{t+k}|y_1, \ldots, y_t) = a_{t,0} + a_{t,1}k + \cdots + a_{t,n-1}k^{n-1}$$

- $n = 1$: local level model
- $n = 2$: linear growth model

❑ `dlmModPoly(order, dV, dW, m0, C0)`

- `order` : 다항식 모델의 차수
- `dV` : 관측잡음의 분산행렬
- `dW` : 이노베이션 분산행렬
- `m0` : 상태변수 초기 평균벡터
- `C0` : 상태변수 초기 분산행렬

Local Level Model

- Local Level Model : 다항식 모델에서 $n = 1$
- Random Walk plus Noise, Steady Model, ARIMA(0,1,1)

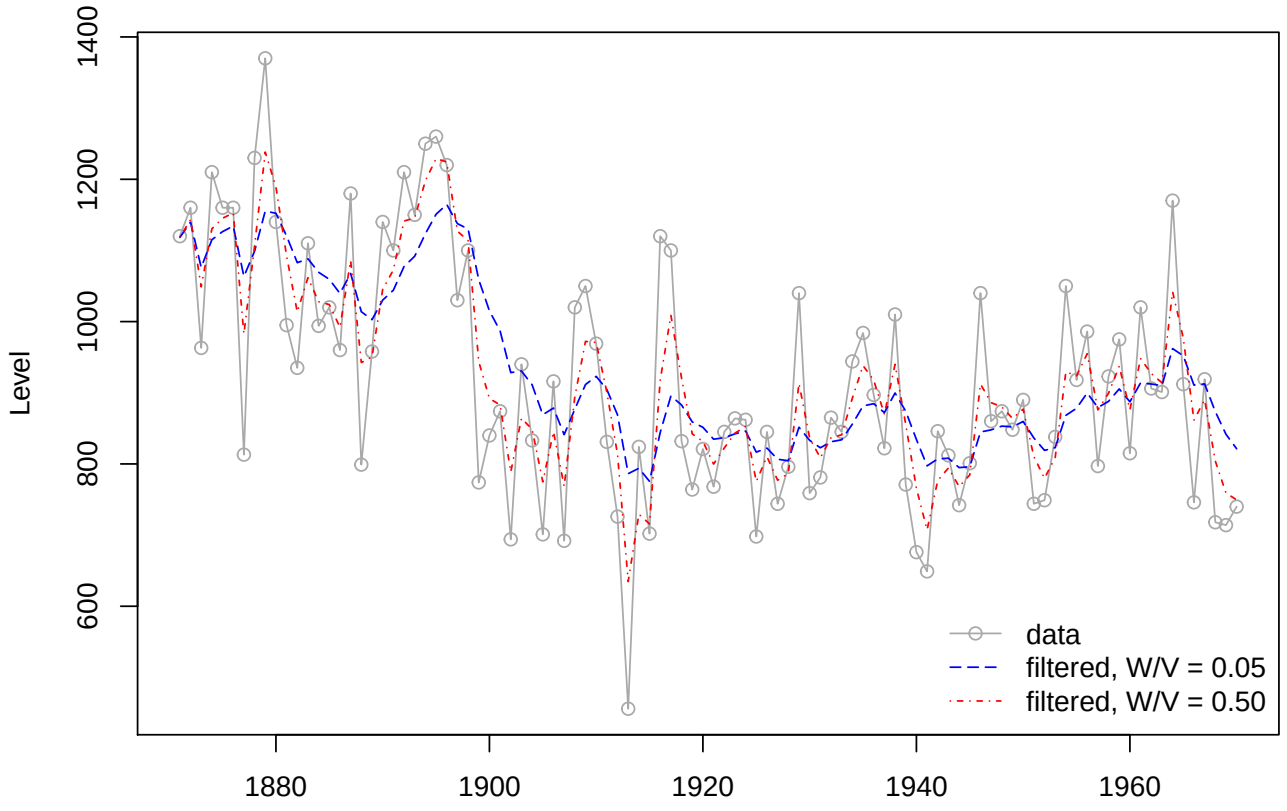
$$
\begin{aligned}
Y_t &= \mu_t + v_t \ , \quad v_t \sim N(0, V) \\
\mu_t &= \mu_{t-1} + w_t \ , \quad w_t \sim N(0, W)
\end{aligned}
$$

$$
\begin{aligned}
E(y_{t+k}|y_1, \ldots, y_t) &= m_t \\
y_{t+k}|y_1, \ldots, y_t &\sim N(m_t, Q_t(k)) \\
Q_t(k) &= C_t + W_t + V
\end{aligned}
$$

Polynomial Model : Filtering

```
> require(dlm, quietly=TRUE)
> plot(Nile, type='o', col= "darkgrey", xlab="", ylab="Level")
> mod1 <- dlmModPoly(order=1, dV=15100, dW=755)
> mod2 <- dlmModPoly(order=1, dV=15100, dW=7550)
> NileFilt1 <- dlmFilter(Nile, mod1)
> NileFilt2 <- dlmFilter(Nile, mod2)
> lines(dropFirst(NileFilt1$m), lty="longdash", col="blue")
> lines(dropFirst(NileFilt2$m), lty="dotdash", col="red")
> leg <- c("data",
+          paste("filtered, W/V =",
+                format(c(W(mod1)/V(mod1), W(mod2)/V(mod2)))))
> legend("bottomright", legend=leg, bty = "n",
+        col=c("darkgrey", "blue", "red"),
+        lty=c("solid", "longdash", "dotdash"),
+        pch=c(1, NA, NA))
```

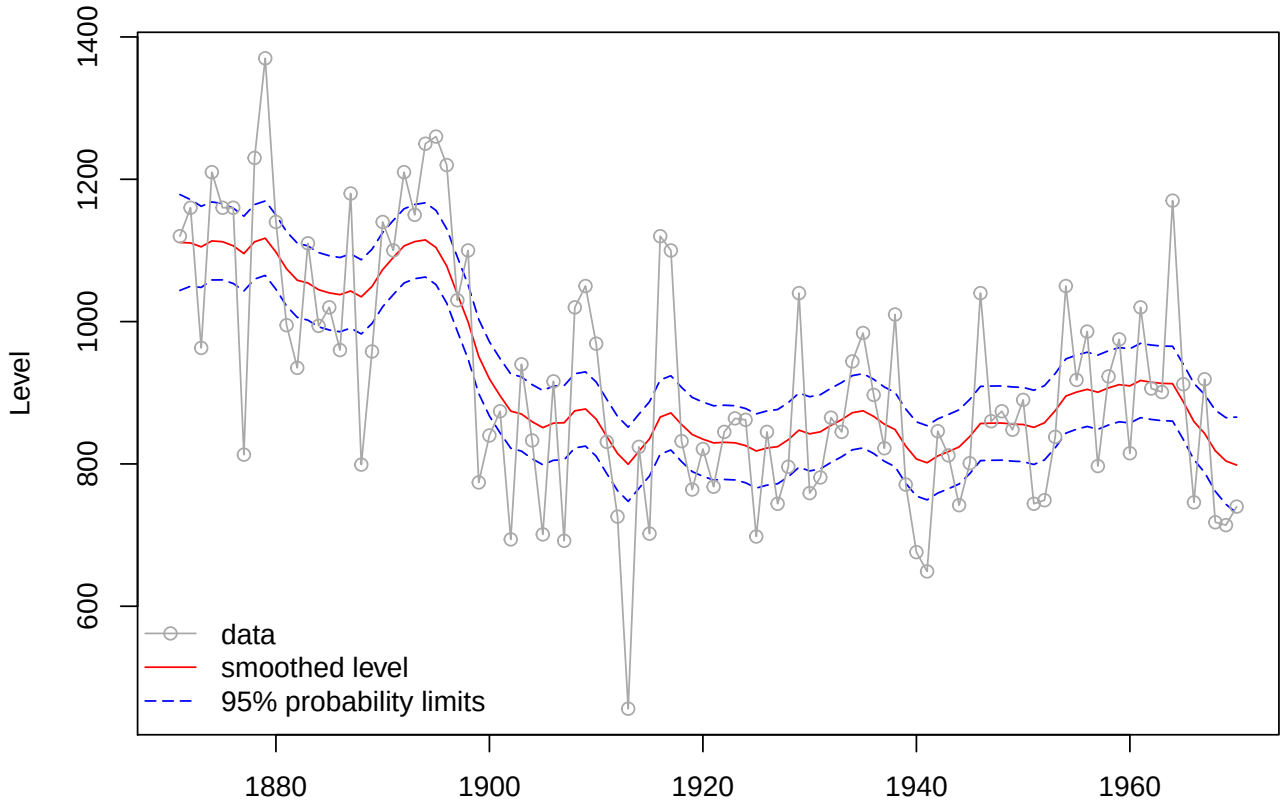
Polynomial Model : Filtering



Polynomial Model : Smoothing

```
> mod <- dlmModPoly(order=1, dV=15100, dW=1468)
> NileFilt <- dlmFilter(Nile, mod)
> NileSmooth <- dlmSmooth(NileFilt)
> hwid <- qnorm(0.025, lower=FALSE) * 2 +
+   sqrt(unlist(dlmSvd2var(NileSmooth$U.S, NileSmooth$D.S)))
> smooth <- cbind(NileSmooth$s,
+                 as.vector(NileSmooth$s) + hwid %o% c(-1, 1))
> plot(dropFirst(smooth), plot.type="s", type="l",
+      lty=c(1, 5, 5), ylab="Level", xlab="",
+      col=c("red", "blue", "blue"),
+      ylim=range(Nile))
> lines(Nile, type="o", col="darkgrey")
> legend("bottomleft", col=c("darkgrey", "red", "blue"),
+        lty=c(1, 1, 5), pch=c(1, NA, NA), bty="n",
+        legend=c("data", "smoothed level", "95% probability limits"))
```

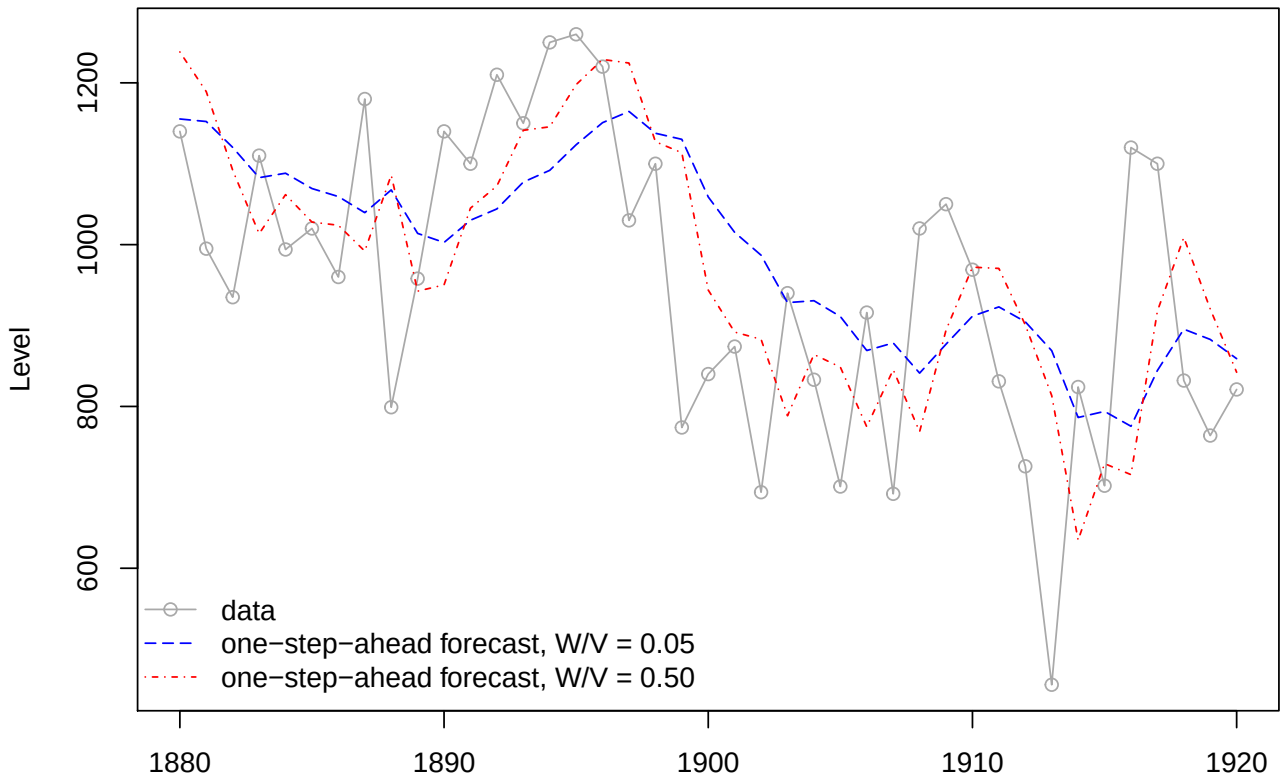
Polynomial Model : Smoothing



Polynomial Model : Forecasting

```
> a <- window(cbind(Nile, NileFilt1$f, NileFilt2$f),
+             start=1880, end=1920)
> plot(a[, 1], type='o', col="darkgrey",
+      xlab="", ylab="Level")
> lines(a[, 2], lty="longdash", col="blue")
> lines(a[, 3], lty="dotdash", col="red")
> leg <- c("data",
+          paste("one-step-ahead forecast, W/V =",
+                format(c(W(mod1) / V(mod1), + W(mod2) / V(mod2)))))
> legend("bottomleft", legend=leg,
+        col = c("darkgrey", "blue", "red"),
+        lty = c("solid", "longdash", "dotdash"),
+        pch = c(1, NA, NA), bty = "n")
```

Polynomial Model : Forecasting



Polynomial Model : Forecasting

```
> a <- window(cbind(Nile, NileFilt1$f, NileFilt2$f),
+             start=1880, end=1920)
> plot(a[, 1], type='o', col="darkgrey",
+      xlab="", ylab="Level")
> lines(a[, 2], lty="longdash", col="blue")
> lines(a[, 3], lty="dotdash", col="red")
> leg <- c("data",
+          paste("one-step-ahead forecast, W/V =",
+                format(c(W(mod1) / V(mod1), + W(mod2) / V(mod2)))))
> legend("bottomleft", legend=leg,
+        col = c("darkgrey", "blue", "red"),
+        lty = c("solid", "longdash", "dotdash"),
+        pch = c(1, NA, NA), bty = "n")
```

Linear Growth Model

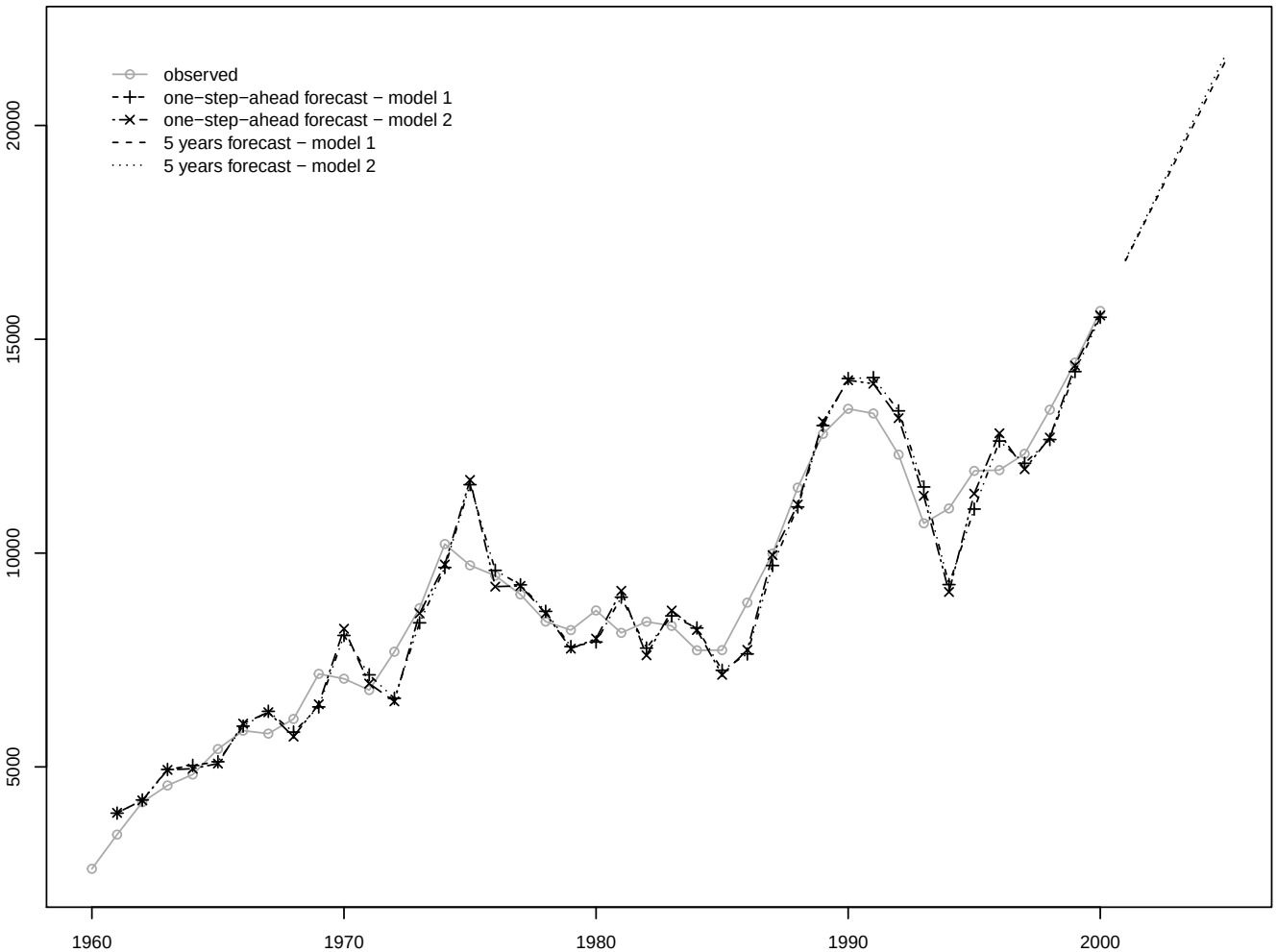
- Linear Growth Model : 다항식 모델에서 $n = 2$
- ARIMA(0,2,2)

$$
\begin{aligned}
Y_t &= \mu_t + v_t \ , \quad v_t \sim N(0, V) \\
\mu_t &= \mu_{t-1} + \beta_{t-1} + w_{t,1} \ , \quad w_{t,1} \sim N(0, W_1) \\
\beta_t &= \beta_{t-1} + w_{t,2} \ , \quad w_{t,2} \sim N(0, W_2)
\end{aligned}
$$

Linear Growth Model

```
> invSpain <- ts(read.table("~/FSA/AB/Datasets/invest2.dat",
+                           colClasses = "numeric")[,2], start = 1960)
> mod1 <- dlmModPoly(dV = 10, dW = c(102236, 321803))
> mod1Filt <- dlmFilter(invSpain, mod1)
> fut1 <- dlmForecast(mod1Filt, n=5)
> mod2 <- dlmModPoly(dV = 10, dW = c(0, 515939))
> mod2Filt <- dlmFilter(invSpain, mod2)
> fut2 <- dlmForecast(mod2Filt, n=5)
> par(mar=c(2,3,1,0) + 0.1, cex=0.7)
> plot(invSpain, xlim=c(1960, 2005), ylim=c(2500, 22000),
+      xlab="", ylab="Investments", type='o', col = "darkgrey")
> lines(dropFirst(mod1Filt$f), type='o', lty="431313", pch=3)
> lines(dropFirst(mod2Filt$f), type='o', lty="22848222", pch=4)
> lines(fut1$f, lty="dashed")
> lines(fut2$f, lty="dotted")
> legend("topleft", bty = "n",
+        legend=c("observed", "one-step-ahead forecast - model 1",
+                 "one-step-ahead forecast - model 2",
+                 "5 years forecast - model 1",
+                 "5 years forecast - model 2"),
+        lty=c("solid", "431313", "22848222", "dashed", "dotted"),
+        pch=c(1,3,4,-1,-1),
+        col=c("darkgrey", rep("black", 4)), inset = 0.05)
```

Polynomial Model : Forecasting



ARMA Model

❏ ARMA 모형

$$y_k = \sum_{j=1}^{p} \phi_j y_{t-j} + \sum_{j=1}^{q} \psi_j e_{t-j}$$

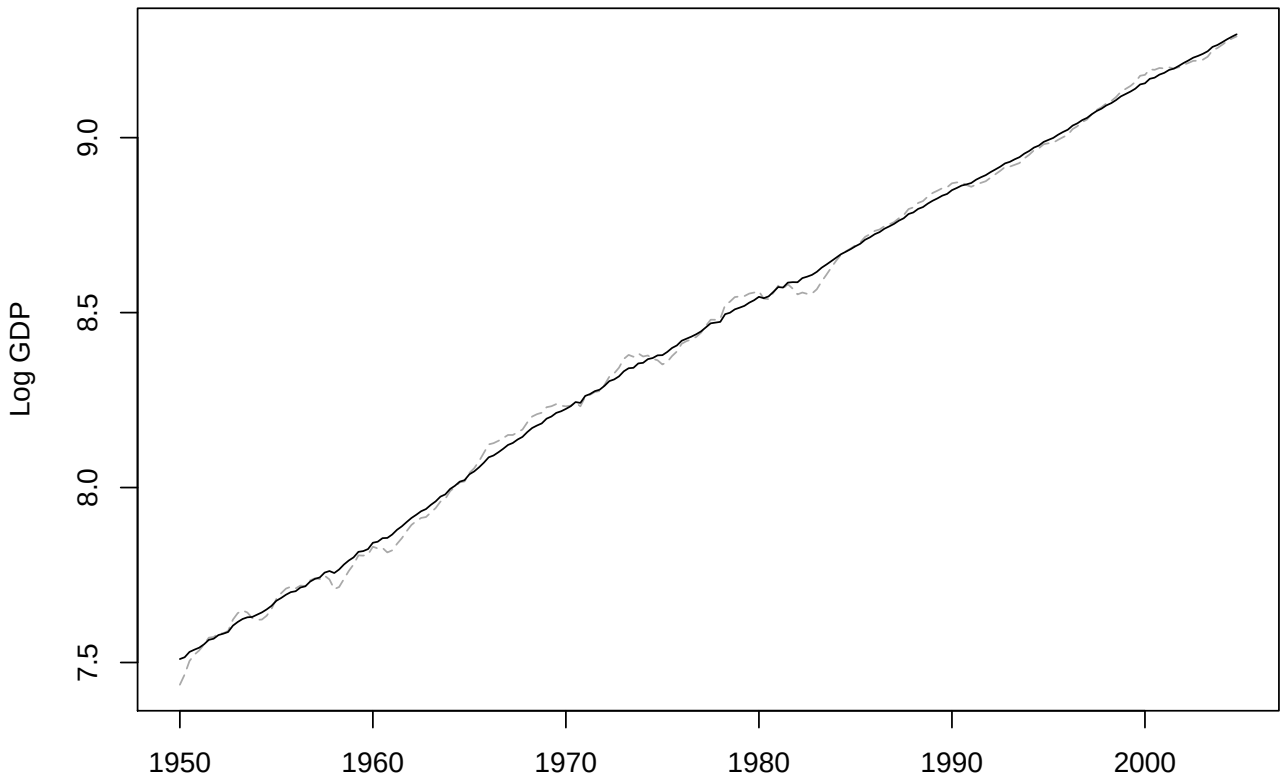
❏ `dlmModARMA(ar, ma, sigma2, dV, m0, C0)`

- `ar` : AR 계수
- `ma` : MA 계수
- `sigma2` : 이노베이션 분산
- `dV` : 관측잡음의 분산행렬
- `m0` : 상태변수 초기 평균벡터
- `C0` : 상태변수 초기 분산행렬

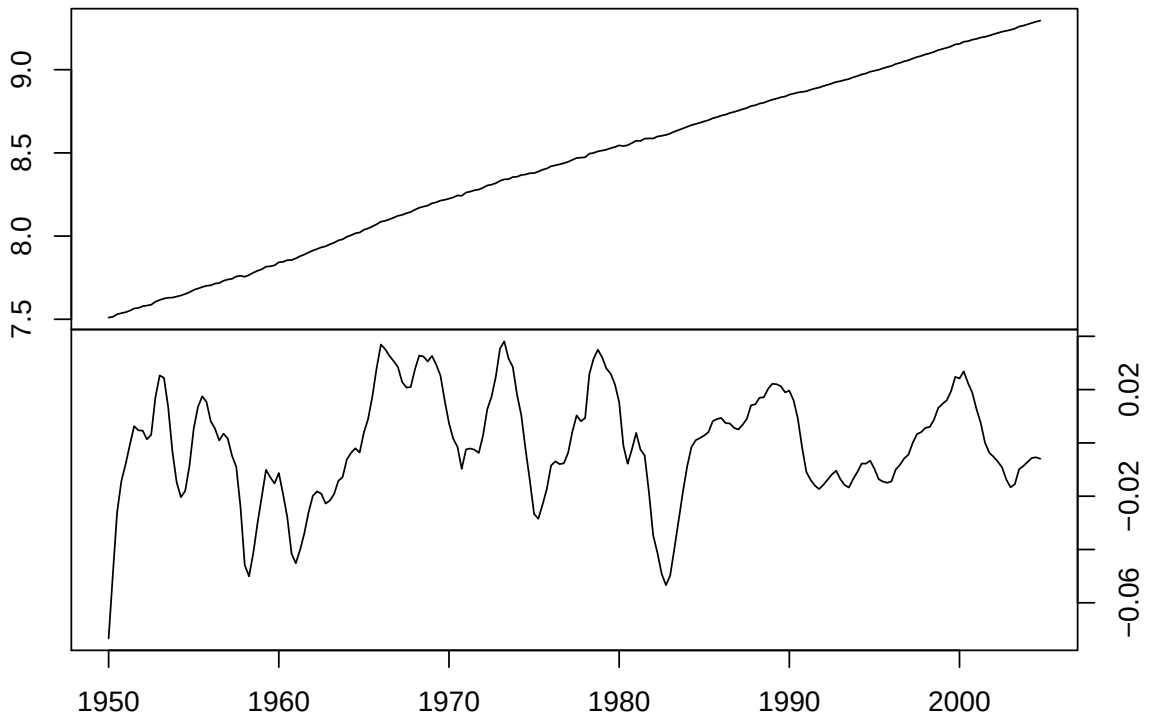
Polynomial Model + ARMA Model

```
> gdp <- ts(gdp, frequency = 4, start = 1950)
> Lgdp <- log(gdp)
> level0 <- Lgdp[1]
> slope0 <- mean(diff(Lgdp))
> u <- c(-10.3060877, -18.9597189, -10.1841145, 1.4806264, -0.5468111)
> trend <- dlmModPoly(dV=1e-7, dW=exp(u[1:2]),
+                     m0=c(level0, slope0), C0=2*diag(2))
> gap <- dlmModARMA(ar=u[4:5], sigma2=exp(u[3]))
> dlmGap <- trend + gap
> gdpSmooth <- dlmSmooth(Lgdp, dlmGap)
> plot(cbind(Lgdp, dropFirst(gdpSmooth$s[, 1])),
+      xlab = "", ylab = "Log GDP", lty = c("longdash", "solid"),
+      col = c("darkgrey", "black"), plot.type = "single")
```

Polynomial Model : Filtering



Polynomial Model : Filtering



Regression Model

❏ 외부 설명변수를 사용한 회귀분석 모형

$$y_t = \beta_1 + \beta_2 x_t + e_t$$

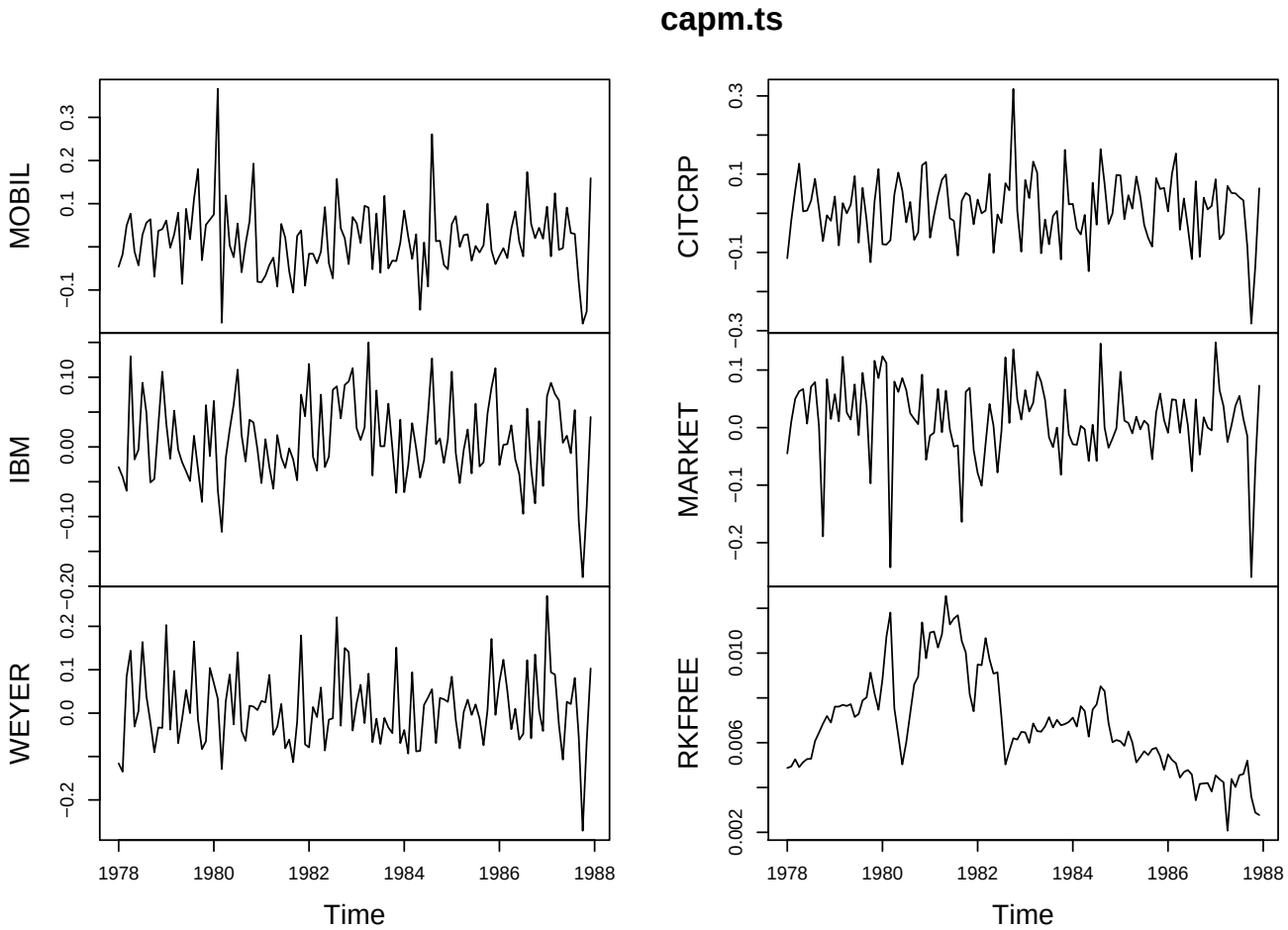
❏ `dlmModReg(X, addInt, dV, dW, m0, C0)`

- `X` : 설명변수 행렬
- `addInt` : intercept 유무
- `dW` : 이노베이션 분산
- `dV` : 관측잡음의 분산행렬
- `m0` : 상태변수 초기 평균벡터
- `C0` : 상태변수 초기 분산행렬

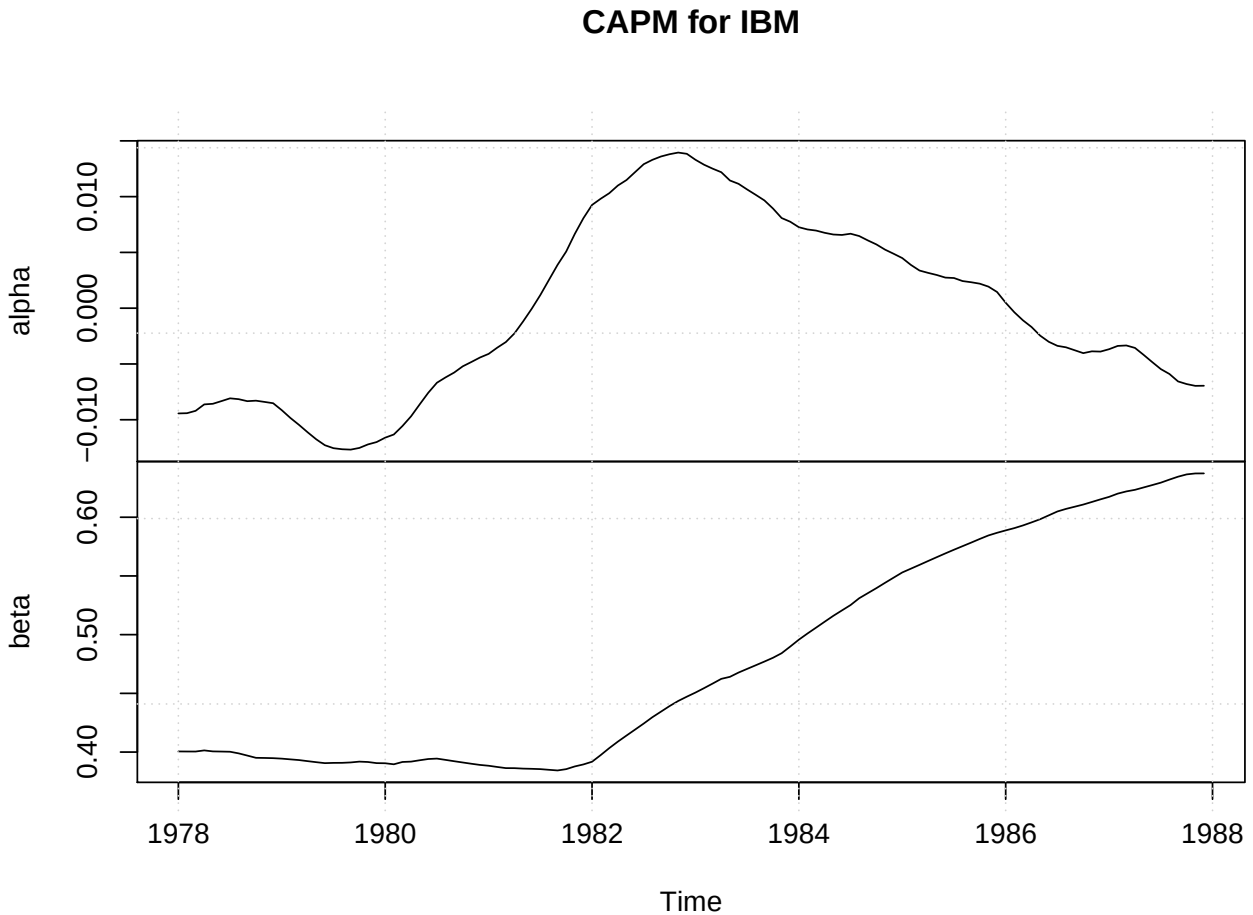
Regression Model

```
> IBM <- capm.ts[, "IBM"] - capm.ts[, "RKFREE"]
> x <- capm.ts[, "MARKET"] - capm.ts[, "RKFREE"]
> mod <- dlmModReg(x,
+                  dV=2.328397e-03,
+                  dW=c(1.100181e-05, 6.496259e-04))
> outS <- dlmSmooth(IBM, mod)
> s <- dropFirst(outS$s)
> colnames(s) <- c("alpha", "beta")
> plot(s, main="CAPM for IBM")
> grid()
```

Regression Model



Regression Model



Seasonality Model

- 계절성 모형
- 주기 s인 모형의 경우 다음과 같은 $s-1$차 행렬을 이용하여 state rotation

$$F = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}$$

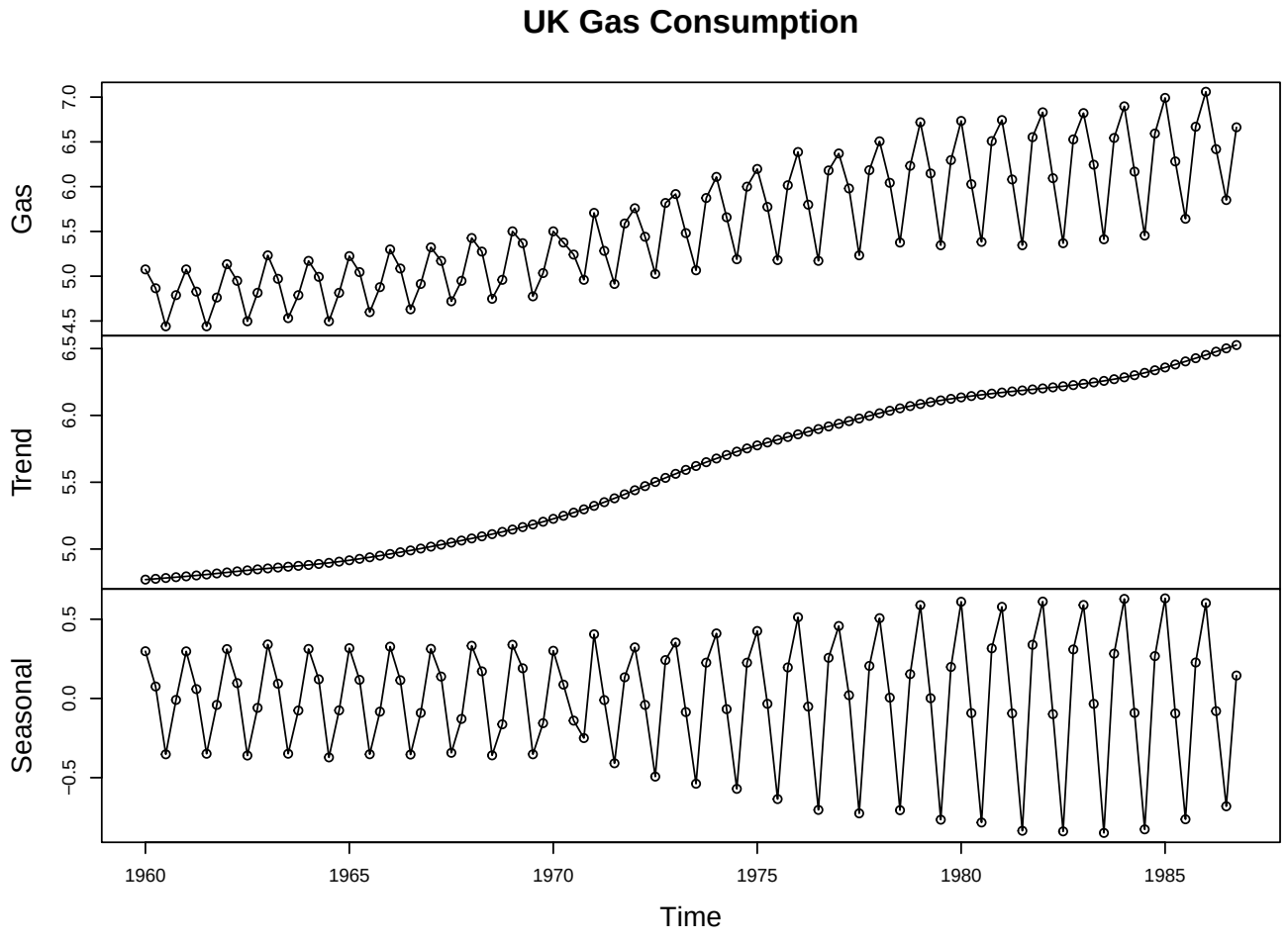
$$G = \begin{bmatrix} -1 & -1 & \cdots & -1 & -1 \\ 1 & 0 & & 0 & 0 \\ 0 & 1 & & 0 & 0 \\ & & \vdots & 0 & 0 \\ 0 & 0 & & 1 & 0 \end{bmatrix}$$

- `dlmModSeas(frequency, dV, dW, m0, C0)`
  - `frequency` : 주기
  - `dW` : 이노베이션 분산
  - `dV` : 관측잡음의 분산행렬
  - `m0` : 상태변수 초기 평균벡터
  - `C0` : 상태변수 초기 분산행렬

Seasonality Model

```
> data(UKgas)
> lGas <- log(UKgas)
> dlmGas <- dlmModPoly() + dlmModSeas(4)
> diag(W(dlmGas))[2:3] <- c(7.901268e-6, 3.308592e-3)
> V(dlmGas) <- 1.822496e-03
> 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")
```

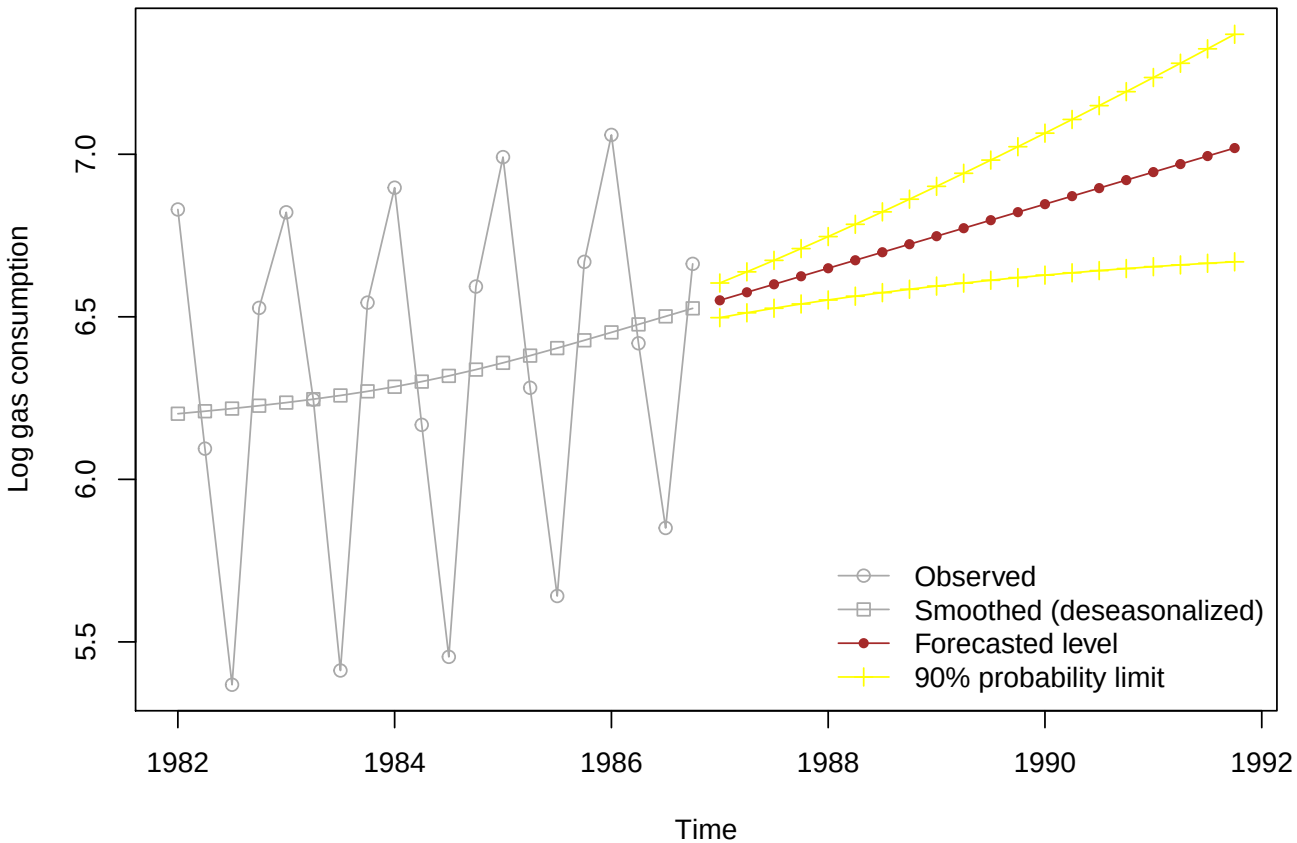
Seasonality Model



Seasonality Model

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

Seasonality Model



모형 계수의 추정

- MLE(Maximum Likelihood Estimation) 사용

$$p(y_1, \ldots, y_n; \psi) = \prod_{t=1}^{n} p(y_t | y_1, \ldots, y_{t-1}; \psi)$$

- 칼만필터의 경우 log-likelihood는 Quadratic form

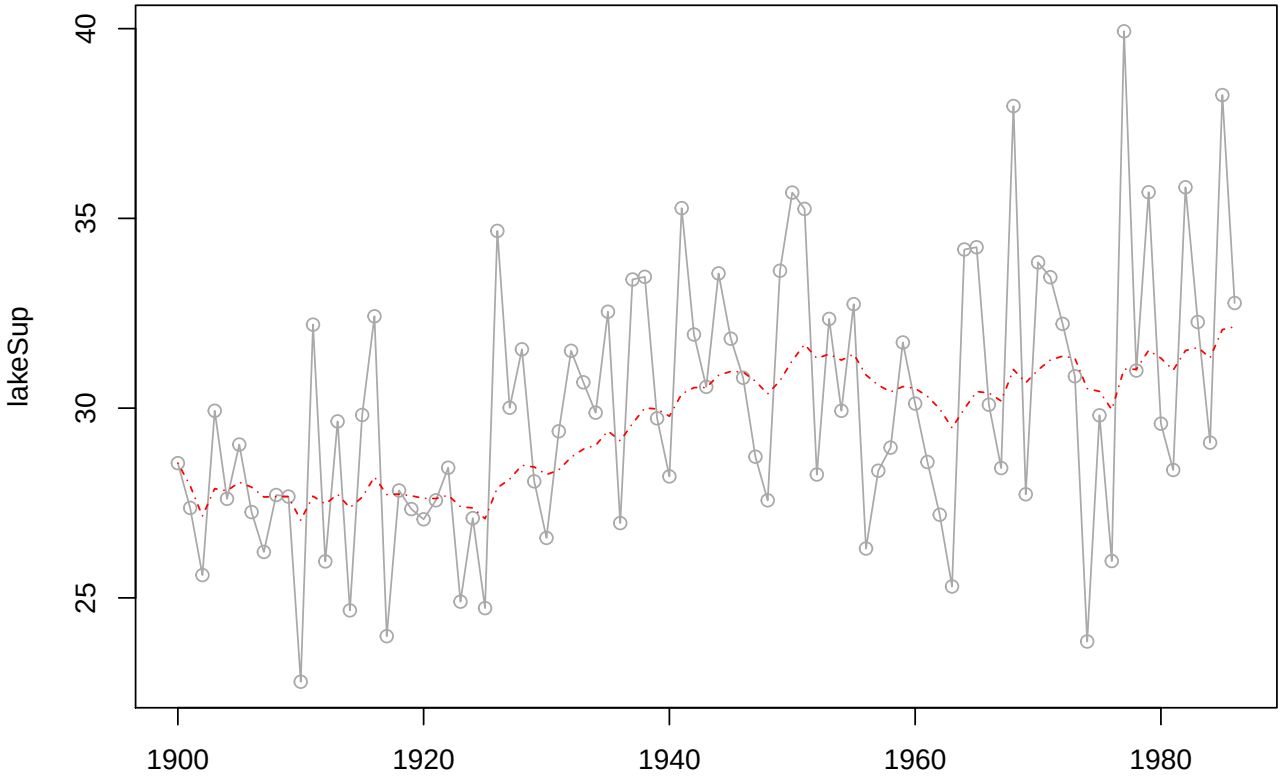
$$l(\psi) = -\frac{1}{2} \sum_{t=1}^{n} \log |Q_t| - \frac{1}{2} \sum_{t=1}^{n} (y_t - f_t)^T Q_t^{-1} (y_t - f_t)$$

- `dlmMLE(y, parm, build)`
  - `y` : 시계열 데이터
  - `parm` : 파라미터 초기값
  - `build` : 파라미터 값을 받아서 dlm 모형을 출력하는 함수

모형 계수의 추정 예 1

```
> build <- function(parm) {
+   dlmModPoly(order=1,
+              dV=exp(parm[1]),
+              dW=exp(parm[2]))
+   }
> fit <- dlmMLE(lakeSup, rep(0, 2), build)
> fit$convergence
> mod <- build(fit$par)
> unlist(mod[c("V", "W")])
> lakeSupFilt <- dlmFilter(lakeSup, mod)
> plot(lakeSup, type='o', col= "darkgrey", xlab="")
> lines(dropFirst(lakeSupFilt$m), lty="dotdash", col="red")
```

모형 계수의 추정 예 1



모형 계수의 추정 예 2

```
> level0 <- Lgdp[1]
> slope0 <- mean(diff(Lgdp))
> buildGap <- function(u) {
+   trend <- dlmModPoly(dV = 1e-7, dW = exp(u[1 : 2]),
+       m0 = c(level0, slope0),
+                                  C0 = 2 * diag(2))
+  gap <- dlmModARMA(ar = u[4 : 5], sigma2 = exp(u[3]))
+  return(trend + gap)
+ }
> init <- c(-3, -1, -3, .4, .4)
> outMLE <- dlmMLE(Lgdp, init, buildGap)
> dlmGap <- buildGap(outMLE$par)
> outMLE$par
[1] -10.3060877 -18.9597189 -10.1841145   1.4806264
[5]  -0.5468111
```