Sorry arima, I'm going Bayesian

Pierre Gauthier

École des Mines de Nancy

May 2019

Tuteur : Denis Villemonais

Introduction

- Projet à partir du billet ***Sorry ARIMA, but I'm Going Bayesian*** par Kim Larsen.
- Étudier une autre approche que celle classique pour les série temporelles.
- Utilisation des méthodes de Monté-carlo par chaîne de Markov.

Sommaire

Méthode de Monté-carlo par chaînes de Markov

Monté-Carlo Markov Chain

- Utilisé initialement lors du projet *Manhattan*, développé par Nicholas Metropolis et Stanislas Ulamn en 1949.
- Amélioration de la méthode avec l'échantillonnage préférentiel par Keith Hastings (1970)
- Application en optimisation (recuit simulé), en physique statistique, en machine learning, ...

Principe de l'Échantillonneur de Gibbs

- On veut obtenir la loi de $\Theta = (\theta_1, \ldots, \theta_n)$
- On ne peut pas exprimer $Loi(\Theta)$ mais on connaît $Loi(\theta_i | (\theta_1, \ldots, \theta_{i-1}, \theta_{i+1}, \ldots, \theta_n))$

Algorithme d'Échantillonnage de Gibbs

Soit $\Theta^{(t)} = \left(\theta_1^{(t)}, \ldots, \theta_n^{(t)}\right)$

1. Prendre des valeurs initiales Θ_0
2. Pour t de 1 à ...
   - Tirer $\theta_1^{(t+1)} \sim \mathbb{P}(\theta_1 | X, \theta_2 = \theta_2^{(t)}, \ldots, \theta_n = \theta_n^{(t)})$
   - ...
   - Tirer $\theta_n^{(t+1)} \sim \mathbb{P}(\theta_2 | X, \theta_1 = \theta_1^{(t+1)}, \ldots, \theta_{n-1} = \theta_{n-1}^{(t+1)})$

- Problème multidimensionnel $\rightarrow$ problème unidimensionnel
- La chaîne de Markov ainsi définie admet comme mesure invariante la distribution jointe de Θ
- On obtient des tirages $\Theta_{i_1}, \ldots, \Theta_{i_M}$ de la distribution jointe de Θ

Nous pouvons ainsi estimer les loi marginales des θ_i par

$$p_{\theta_i}(x) = \frac{1}{m} \sum_{t=1}^{m} p(x|\theta_1^{(t)}, \ldots, \theta_n^{(t)}))(Rao - Blackwellized)$$

Convergence ?

- Regarder la trace
- Regarder les autocorrélations des tirages.
- Comparer des parties de l'échantillon avec le *Geweke z-score*

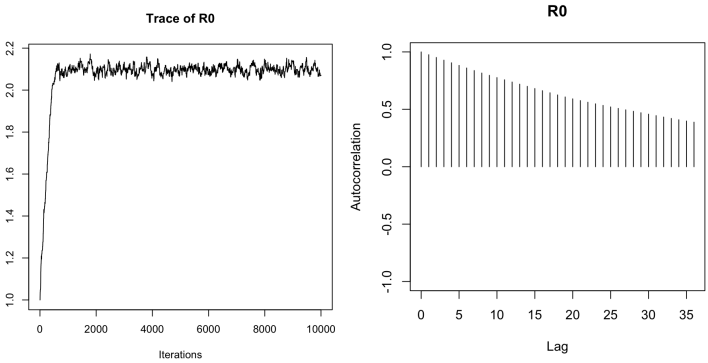


Figure – Exemple pour un paramètre de régression

Un exemple simple pour l'Échantilloneur de Gibbs

On simule la loi d'un vecteur gaussien $\begin{pmatrix} X \\ Y \end{pmatrix} \sim N\left[\begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \begin{pmatrix} \sigma_1^2 & \rho \\ \rho & \sigma_2^2 \end{pmatrix}\right]$

On utilise l'échantillonneur de Gibbs en itérant :

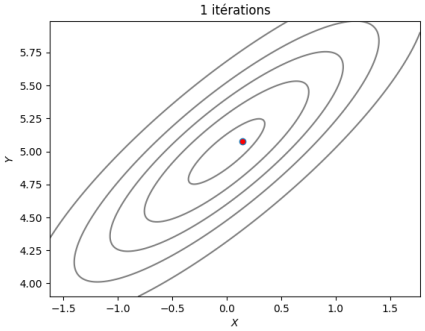
$$x_{n+1} \sim \mathcal{N}\left(\mu_1 + \rho\frac{\sigma_1}{\sigma_2}\left(y_n - \mu_2\right), \sigma_1^2(1-\rho)\right)$$

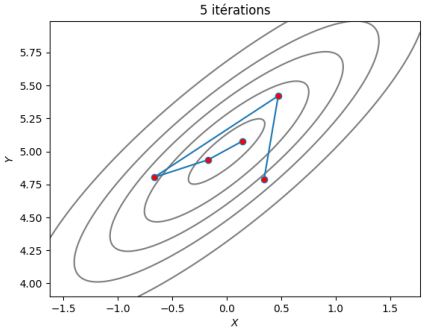
$$y_{n+1} \sim \mathcal{N}\left(\mu_2 + \rho\frac{\sigma_2}{\sigma_1}\left(x_{n+1} - \mu_1\right), \sigma_2^2(1-\rho)\right)$$

```python
def sample_x_given_y(y, mean, var):
    mu = mean[0] + var[0,1] * math.sqrt(var[0, 0]) / math.sqrt(var[1, 1]) * (y - mean[1])
    var = var[0, 0] * (1 - var[1,0])
    return np.random.normal(mu, var)

def sample_y_given_x(x, mean, var):
    mu = mean[1] + var[0, 1] * math.sqrt(var[1, 1]) / math.sqrt(var[0, 0]) * (x - mean[0])
    var = var[1, 1] * (1 - var[1,0])
    return np.random.normal(mu, var)

def gibbs_sampler(mean, var, N_iter):
    samples = np.zeros((N_iter, 2))
    y = mean[1]

    for i in range(N_iter):
        x = sample_x_given_y(y, mean, var)
        y = sample_y_given_x(x, mean, var)
        samples[i, :] = [x, y]

    return samples
```

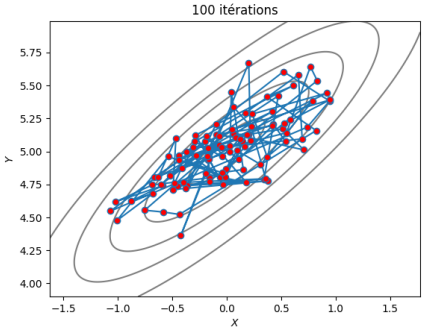
Listing 1 – Implémentation Python

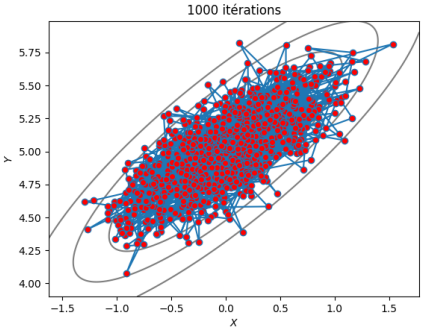
Simulations $\begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 5 \end{pmatrix}$, $\sigma_1^2 = 1$, $\sigma_2^2 = 0.5$, $\rho = 0.6$



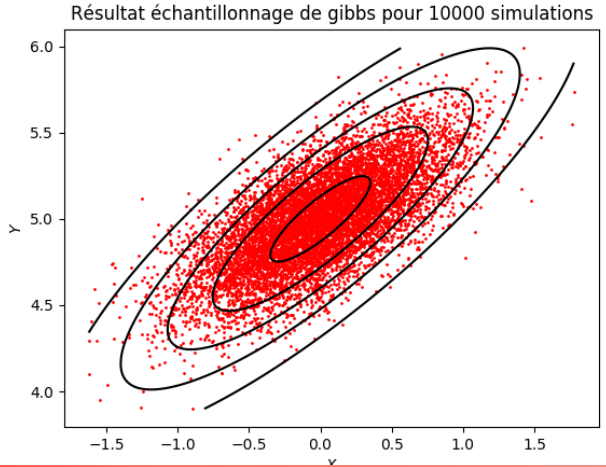


Simulations $\begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 5 \end{pmatrix}$, $\sigma_1^2 = 1$, $\sigma_2^2 = 0.5$, $\rho = 0.6$





Simulations $\begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 5 \end{pmatrix}$, $\sigma_1^2 = 1$, $\sigma_2^2 = 0.5$, $\rho = 0.6$



Application numérique : Une alternative à l'algorithme E-M pour le mélange de gaussiennes

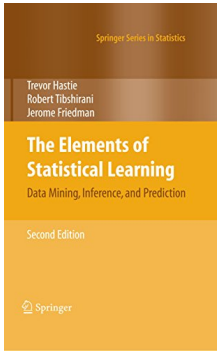
Échantillon $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$ suivant une loi $f(\boldsymbol{x}_i, \theta)$ avec des variables latentes $\boldsymbol{z} = (\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_n)$

→ **Algorithme E-M :** Maximisation de $L(\boldsymbol{\theta}; \mathbf{X}) = p(\mathbf{x}|\boldsymbol{\theta}) = \int p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta}) d\mathbf{Z}$

$$L(\mathbf{x}; \boldsymbol{\theta}) = Q\left(\boldsymbol{\theta}; \boldsymbol{\theta}^{(c)}\right) - H\left(\boldsymbol{\theta}; \boldsymbol{\theta}^{(c)}\right)$$

$$\begin{cases} Q\left(\boldsymbol{\theta}; \boldsymbol{\theta}^{(c)}\right) = E\left[L((\mathbf{x}, \mathbf{z}); \boldsymbol{\theta}) | \boldsymbol{\theta}^{(c)}\right] \\ H\left(\boldsymbol{\theta}; \boldsymbol{\theta}^{(c)}\right) = E\left[\sum_{i=1}^{n} \log f\left(z_i | \boldsymbol{x}_i, \boldsymbol{\theta}\right) | \boldsymbol{\theta}^{(c)}\right] \end{cases}$$

$\boldsymbol{\theta}^{(c+1)} = \arg\max_{\boldsymbol{\theta}} \left(Q\left(\boldsymbol{\theta}, \boldsymbol{\theta}^{(c)}\right)\right)$ fait tendre la suite $L\left(\mathbf{x}; \boldsymbol{\theta}^{(c+1)}\right)$ vers un max local



Application : mélange de gaussiennes

- Les données $\mathbf{x}$ viennent de gaussiennes $\{C_1, \ldots, C_M\}$
- $\boldsymbol{z}_i \in \{1, \ldots, m\}$ si un individu i vient de $C_1, \ldots, C_M$

$$\boldsymbol{x}_i | (\boldsymbol{z}_i = j) \sim \mathcal{N}\left(\mu_j, \sigma_j^2\right) \quad \mathbb{P}(\boldsymbol{z}_i = j) = \pi_j$$

$\theta = \left(\pi, \mu, \sigma^2\right)$

- **E-M** : maximisation de $\hat{\theta}_{\mathrm{MLE}}(x) = \arg\max_{\theta} f(x|\theta)$

$$\ell_\theta(x) = \sum_{n=1}^{N} \log \left(\sum_{j=1}^{M} \pi_j \, \mathcal{N}_{\mu_j, \sigma_j^2}(x_n) \right)$$

- **Bayesien** estimateur $\hat{\theta}_{MMSE}(x) = E[\theta|x]$

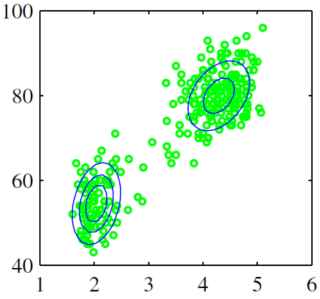


Figure – Mélange de deux gaussiennes [F. Sur, Introduction à l'apprentissage automatique]

Application : Échantillonneur de Gibbs pour le mélange de 2 gaussiennes

Pour 2 gaussiennes : $\begin{cases} \boldsymbol{x}_i | (\boldsymbol{z}_i = 1) \sim \mathcal{N}(\mu_1, \sigma_1^2) & \mathbb{P}(\boldsymbol{z}_i = 1) = \pi_1 \\ \boldsymbol{x}_i | (\boldsymbol{z}_i = 2) \sim \mathcal{N}(\mu_2, \sigma_2^2) & \mathbb{P}(\boldsymbol{z}_i = 2) = \pi_2 = 1 - \pi_1 \end{cases}$

$\rightarrow$ On veut obtenir $\Theta = (\boldsymbol{z}, \mu) = (\boldsymbol{z_1}, \ldots, \boldsymbol{z_1}, \mu)$ (π, σ fixés à $\hat{\pi}, \hat{\sigma}$)

Application : Échantillonneur de Gibbs pour le mélange de 2 gaussiennes

- $\Theta = (\boldsymbol{z}, \mu) = (\boldsymbol{z_1}, \ldots, \boldsymbol{z_1}, \mu)$
- $\boldsymbol{z}\big|\mu$: On a $p(\mathcal{C}_1|x) = \frac{p(\mathcal{C}_1)p(x|\mathcal{C}_1)}{p(\mathcal{C}_1)p(x|\mathcal{C}_1)+p(\mathcal{C}_2)p(x|\mathcal{C}_2)}$

  Utiliser $\mathbb{P}(\boldsymbol{z}_i = j|\mu, \boldsymbol{x}) = \frac{\hat{\pi}_j p_{\mathcal{N}}(\boldsymbol{x}_i, \mu_j, \hat{\sigma}_j)}{\hat{\pi}_1 p_{\mathcal{N}}(\boldsymbol{x}_i, \mu_1, \hat{\sigma}_1) + \hat{\pi}_2 p_{\mathcal{N}}(\boldsymbol{x}_i, \mu_2, \hat{\sigma}_2)} \quad j \in \{1, 2\}$
- $\mu\big|\boldsymbol{z}$:

(à priori)	(à postériori)	(= maj des coefs)
$\boldsymbol{x_i}|\mu, \boldsymbol{z_i} = j \sim \mathcal{N}(\mu_j, 1/\hat{\tau}_j)$	$p(\mu_j|\boldsymbol{z}_i, \mathbf{x}) \propto p(\boldsymbol{x}|\mu, \boldsymbol{z_i} = j)p(\mu_j)$	$\tau'_{j_0} = \tau_0 + N\hat{\tau}_j$
$\mu_j \sim \mathcal{N}(\mu_0, 1/\tau_0)$	$\sim \mathcal{N}(\mu'_{j_0}, 1/\tau'_{j_0})$	$\mu'_{j_0} = \frac{N\ \hat{\tau}_j\ \bar{x} + \tau_0\mu_0}{N\hat{\tau}_j + \tau_0}$

$\left(\tau_j = \mathbf{1}/\sigma_j^{\mathbf{2}},\ \bar{x} = \frac{\mathbf{1}}{N}\sum_{i=\mathbf{0}}^{N} x_i\right)$

Échantilloneur de Gibbs pour une mixture gaussienne

1. Prendre les valeurs initiales $\Theta_0 = (\boldsymbol{z}^{(0)}, \mu_1^{(0)}, \mu_2^{(0)})$, μ_0, τ_0
2. Pour t de à ...
   - Pour i de 1 à N tirer

     $\boldsymbol{z}_i^{(t+1)} \in \{1, 2\}$ avec $\mathbb{P}(\boldsymbol{z}_i^{(t+1)} = 1 | \mu^{(t)}) = \dfrac{\hat{\pi}_1 p_{\mathcal{N}}(\boldsymbol{x}_i, \mu_1^{(t)}, \hat{\sigma}_1)}{\hat{\pi}_1 p_{\mathcal{N}}(\boldsymbol{x}_i, \mu_1^{(t)}, \hat{\sigma}_1) + \hat{\pi}_2 p_{\mathcal{N}}(\boldsymbol{x}_i, \mu_2^{(t)}, \hat{\sigma}_2)}$
   - Pour $j \in \{1, 2\}$

     avec $\tau'_{j_0} = \tau_0 + N\hat{\tau}_j \quad \mu'_{j_0} = \dfrac{N\ \hat{\tau}_j\ \bar{x} + \tau_0 \mu_0}{N\hat{\tau}_j + \tau_0}$

     Tirer $\mu_j^{(t+1)}\ \boldsymbol{z} \sim \mathcal{N}\left(\mu'_{j_0}, 1/\tau'_{j_0}\right)$

```
Z_given_mu <- function(X,Z,mu,pi_1){
    # Z variables latentes

    remove_i <- (c(1:length(X)) !=i) # enleve variable i
    estimate_sigma_1 <- sd(X[(remove_i & Z == 1)]) # classe 1
    estimate_sigma_2 <- sd(X[(remove_i & Z == 2)]) # classe 2

    for (i in 1:length(Z)){

        proba1 <- pi_1 * dnorm(X[i],mu[1],estimate_sigma_1) /
            (pi_1 * dnorm(X[i],mu[1],estimate_sigma_1) +
            (1-pi_1) * dnorm(X[i],mu[2],estimate_sigma_2))

        Z[i] = sample(1:2, size=1,prob=c(proba1, 1-proba1),
            replace=TRUE)
    }
    return(Z)
}
```

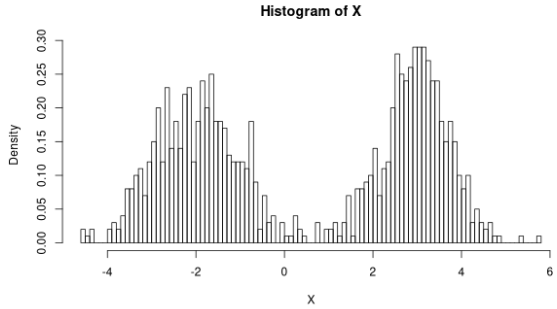
```
mu_given_Z = function(X, Z, mu_prior){
    # Z variables latentes
    # mu_prior contient parametres loi a priori de mu

    mu = rep(0,2) ; sigma = rep(0,2)

    for(j in 1:2){

        sample_j_size = sum(Z==j)
        sample_j_mean = mean(X[Z==j])
        sigma[j] = sd(X[Z==j]) ; precision_j = 1 / sigma[j]^2

        precision_post = sample_j_size * precision_j +
            mu_prior$precision

            mean_post = (sample_j_mean * sample_j_size *
            precision_j + mu_prior$mean *
            mu_prior$precision ) / precision_post

        mu[j] = rnorm(1,mean_post,sqrt(1/precision_post)) # on
                tire mu selon la loi normale a posteriori
    }
    return(list(mu = mu, sigma = sigma))
    }
```

```
echantillonneur_gibbs <- function(X,N_simu){
    # X sont les donnees
    # initalisation
    .
    .

    for (k in 1:N_simu){

        Z <- Z_given_mu(X,Z,mu,pi_1) ; param_post <- mu_given_Z(X, Z, mu_prior)
        mu = param_post$mu
    .
    .
    .
    }
    return(list(Z, mu_1, mu_2))
}
```

Listing 2 – Implémentation en R

Simulations

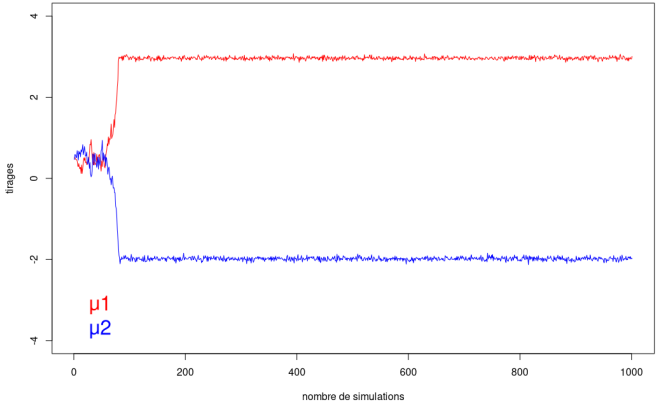
Données : on gènere équiprobablement 1000 points à partir de 2 gaussiennes $\mathcal{N}(-2,1), \mathcal{N}(-3,0.5)$



On prend à priori $\mu \sim \mathcal{N}(0,4)$

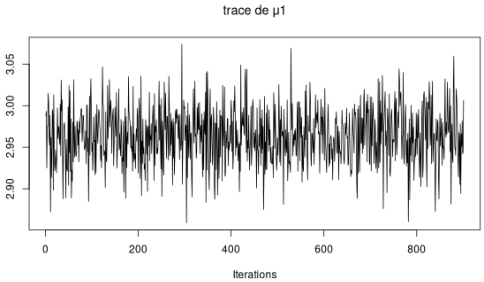
Simulations

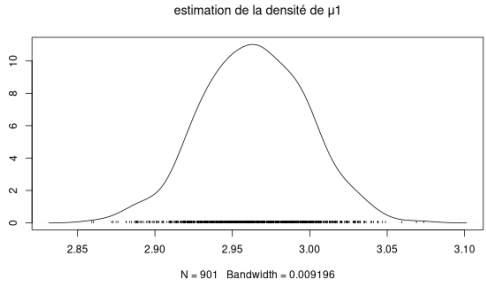
Pour 1000 itérations



→ *burn-in* de 100

Simulations : diagnostic de μ avec le package coda





(sans le *burn-in*)

Simulations : diagnostic de μ avec le package coda

variable	moyenne	95% CI lower	95% CI upper	vraie valeur
μ_1	2.963882	2.899498	3.032716	3
μ_2	-1.981757	-2.061083	-1.902244	-2

Approche bayésienne pour les séries temporelles

Approche bayésinne pour la modélisation des séries temporelles

- Les modèle espace-états

		bruit blanc gaussien
equation d'observation	$y_t = Z_t^T \alpha_t + \epsilon_t$	$\epsilon_t \sim N(0, H_t)$
equation de transition	$\alpha_{t+1} = T_t \alpha_t + R_t \eta_t$	$\eta_t \sim N(0, Q_t)$

- y_t observations
- α_t variables d'états / latentes / cachées
- Z_t matrice de mesure
- T_t matrice de transition

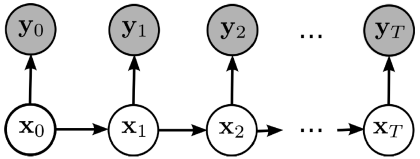


Figure – Hidden Markov Chain[researchgate.net]

Approche bayésienne pour la modélisation des séries temporelles

→ Bayesian structural time series (BSTS)

		bruit blanc gaussien
observation	$y_t = \mu_t + \beta^T x_t + \tau_t + \varepsilon_t$	$\varepsilon_t \sim N\left(0, \sigma_\varepsilon^2\right)$
regression	$\beta^T x_t$	
tendance + marche aléatoire	$\mu_t = \mu_{t-1} + \delta_{t-1} + u_t$	$u_t \sim N\left(0, \sigma_u^2\right)$
marche aléatoire	$\delta_t = \delta_{t-1} + v_t$	$v_t \sim N\left(0, \sigma_v^2\right)$
saisonnalité	$\tau_t = -\sum_{s=1}^{s-1} \tau_{t-s} + w_t$	$w_t \sim N\left(0, \sigma_w^2\right)$

→ Bayesian structural time series (BSTS)

observation	$y_t = Z_t^T \alpha_t + \epsilon_t$	$\epsilon_t \sim N(0, H_t)$
	Z_t^T $\begin{pmatrix} 1 & 0 & \beta^T \mathbf{x}_t \end{pmatrix}$	α_t^T $\begin{pmatrix} \mu_t & \delta_t & 1 \end{pmatrix}^T$
equation de transition	$\alpha_{t+1} = T_t \alpha_t + R_t \eta_t$	$\eta_t \sim N(0, Q_t)$
α_t $\begin{pmatrix} \mu_t \\ \delta_t \\ 1 \end{pmatrix}$	T_t $\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$N_t \eta_t$ $\begin{pmatrix} u_t \\ v_t \\ w_t \end{pmatrix}$

→ estimation des paramètres

Loi à postériori états cachés α_t : Le filtre de Kalman

Itérations sur l'estimation $p(\alpha_t|y_{1:t}) \sim \mathcal{N}(\hat{\alpha}_t, P_t)$

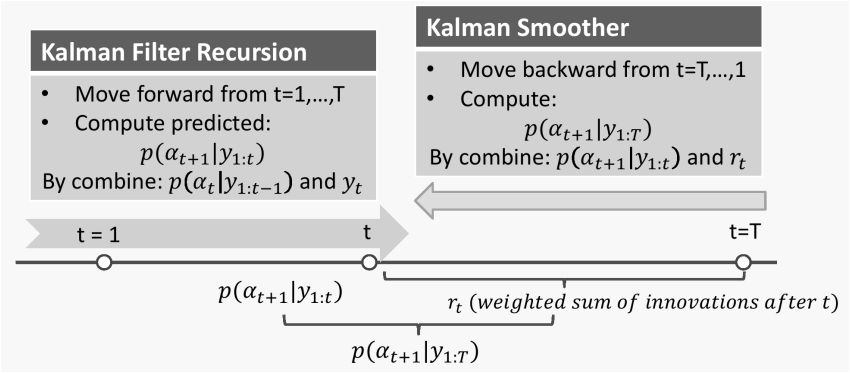


Figure – [github : anhdanggit/nowcasting-google-queries/]

Loi à postériori de β : *spike-and-slab prior*

- On prend la partie regression $y_t^* = y_t - \mu_t$
- On utilise pour β une distribution à priori *spike-and-slab* :
  - $p(\gamma) = \prod_{k=1}^{N} \pi^{\gamma_k}(1-\pi)^{1-\gamma_k}$, $\gamma_k \in \{0,1\}$ $N = Card(\mathbf{x})$
  - Á priori : $p\left(\beta, \gamma, \sigma_\varepsilon^2\right) = p\left(\beta_\gamma | \gamma, \sigma_\varepsilon^2\right) p\left(\sigma_\varepsilon^2 | \gamma\right) p(\gamma)$
  - $\beta_\gamma \left| \sigma_\epsilon^2, \gamma \sim \mathcal{N}\left(b_\gamma, \sigma_\epsilon^2 \left(\Omega_\gamma^{-1}\right)^{-1}\right) \right. \quad \sigma_\epsilon^2 \left| \gamma \sim IG\left(\frac{\nu}{2}, \frac{ss}{2}\right)\right.$

  papramètres à priori : ν nombre de paramètres, $\frac{ss}{\nu} = \left(1 - R^2\right) s_y^2$, $\Omega^{-1} \propto X^T X$

- On utilise les propriété des lois conjugé pour obtenir les loi à postériori

  $\beta_\gamma | \sigma_\epsilon, \gamma, \mathbf{y}^*$ $\quad$ $\gamma_\epsilon^2 | \gamma, \mathbf{y}^*$ $\quad$ $\gamma | \mathbf{y}^*$
- Intérêt de la *spike-and-slab*

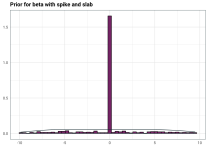


Figure – [batisengul.co.uk]

$\beta_\gamma = \beta[\gamma_k \neq 0]$

Échantilloneur de Gibbs pour BSTS : SSVS algorithm

$\Theta = (\gamma, \beta, \sigma_\varepsilon^2, \sigma_v^2, \sigma_u^2)$

- Choisir les paramètres à priori v, R^2, s_y^2, π
- Tirer $\gamma, \beta, \sigma_\varepsilon^2, \sigma_v^2, \sigma_u^2$ σ_u^2,σ_u^2,σ_w^2 sont tiré selon la loi $.|\gamma \sim IG\left(\frac{\nu}{2}, \frac{ss}{2}\right)$

Sur $1, \ldots, M$:

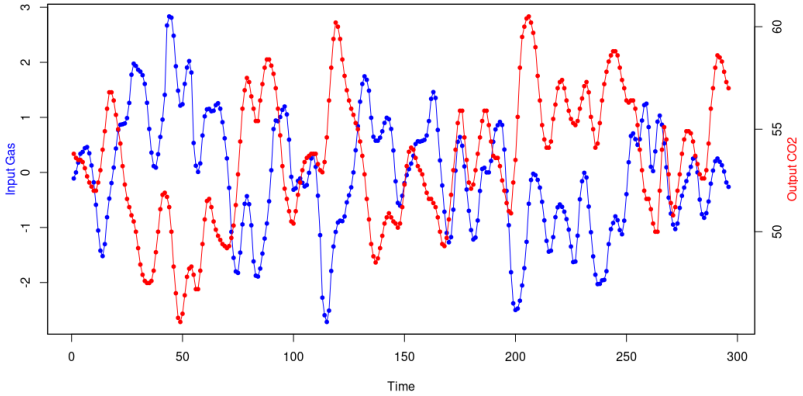
1. Après application du filtre de Kalman, on tire les états latents α depuis $p\left(\alpha|y, \gamma, \beta, \sigma_\varepsilon^2, \sigma_v^2, \sigma_u^2\right)$
2. On tire σ_u^2 et σ_v^2 selon $p\left(\frac{1}{\sigma_u^2}, \frac{1}{\sigma_v^2}|y, \alpha, \beta, \sigma_\varepsilon^2\right)$
3. On tire β et σ_ϵ^2 selon $p\left(\beta, \sigma_\varepsilon^2|y, \alpha, \sigma_u^2, \sigma_v^2\right)$

On prend comme modèle la moyenne des tirages $\left(\Theta^m, \ldots, \Theta^M\right)$

Utilisation du modèle BSTS sur un jeu de données

On prend le jeu de données ***CO2*** comprenant 295 observations

- *Input Gas* : arrivée d'essence
- *Output CO2* : CO2 en sortie

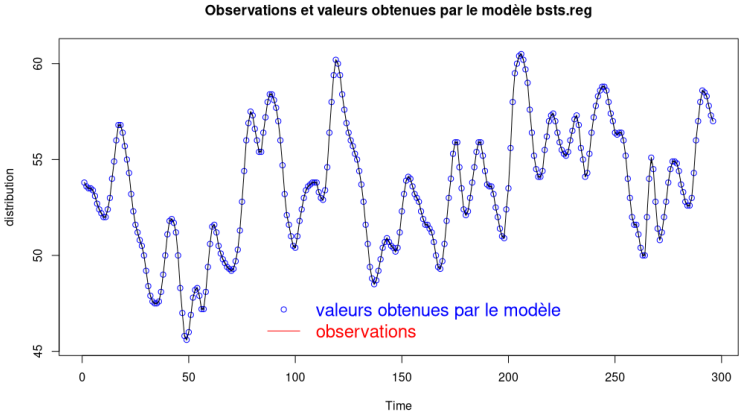


On utilise le modèle précédent pour exprimer *Input Gas* en fonction de *Output CO2*.

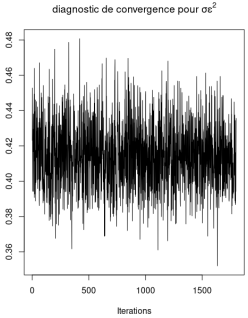
```
library(bsts)
ss <- AddLocalLinearTrend(list(), CO2$output.co2)
bsts.reg <- bsts(output.co2 ~ ., state.specification =
    ss, data =
CO2, niter = 2000, ping=0, seed=2016)
```

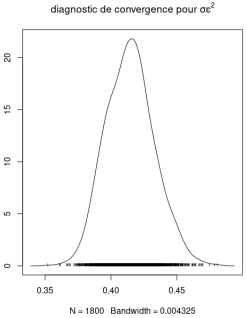
$$y_t = \mu_t + \beta^T \mathbf{x}_t + \epsilon_t \qquad \epsilon_t \sim N\left(0, \sigma_\epsilon^2\right)$$
$$\mu_t = \mu_{t-1} + \delta_{t-1} + u_t \qquad u_t \sim \mathcal{N}\left(0, \sigma_u^2\right)$$

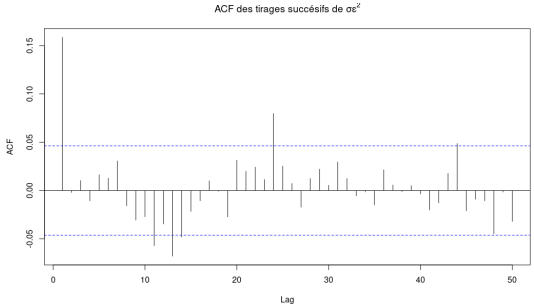
Listing 3 – package *bsts*



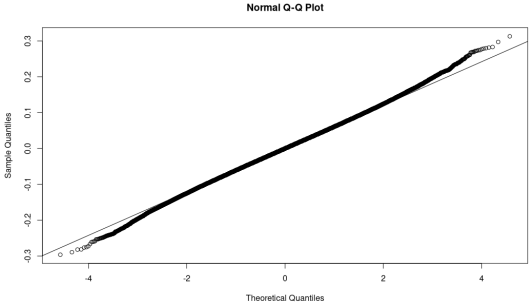
Convergence

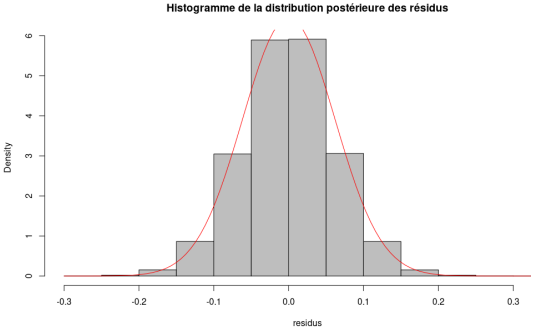






Résidus






```
> MAPE=mean(abs(colMeans(bsts.reg$one.step.prediction.errors))/CO2$output.co2)
> MAPE
[1] 0.005890585
> MAPE
```

Comparaison avec un modéle à fontion de transfert

Modèle à fonction de transfert : $Y_t = \mu + \frac{\Omega(B)}{\Delta(B)} X_{t-b} + \frac{\Theta(B)}{\Phi(B)} \varepsilon_t$
Avec :

- (Y_t) chronique à modeliser
- (X_t) chronique explicative
- $\frac{\Omega(B)}{\Delta(B)}$ fonction de transfert
- (u_t) chronique des erreurs

$\rightarrow$ On blanchit la chronique X_t : $\chi_t = \frac{\Phi_1(B)}{\Theta_1(B)} X_t$

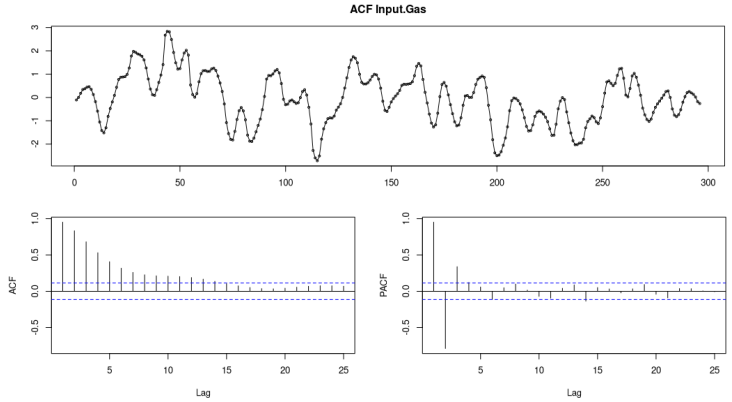
$\rightarrow \Upsilon_t = \frac{\Phi_1(B)}{\Theta_1(B)} Y_t = \frac{\Omega(B)}{\Delta(B)} \chi_{t-b} + \widetilde{\varepsilon}_t$

On utilise le corrélogramme croisé $\rho(h) = \frac{\mathrm{Cov}(\chi_t, \Upsilon_{t+h})}{\sqrt{\mathrm{Var}(\chi_t) \cdot \mathrm{Var}(\Upsilon_t)}} = \begin{cases} \nu_h \frac{\sigma_\chi}{\sigma_r} & \text{si } h \geqslant 0 \\ 0 & \text{si } h < 0 \end{cases}$

où $\Upsilon_t = \sum_{h \geqslant 0} \nu_h \chi_{t-h} + \widetilde{\varepsilon}_t$

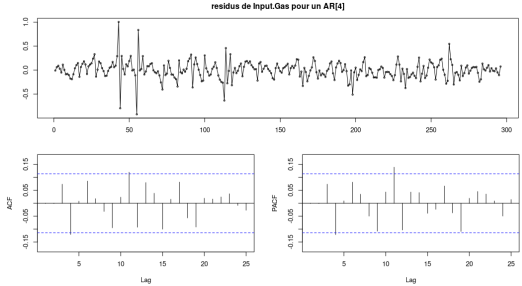
Application sur le jeu de données CO2

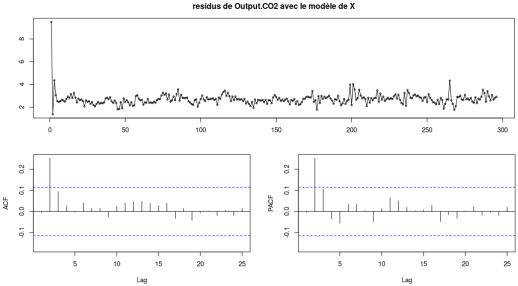
→ blanchissement de *Input Gas*.



Application sur le jeu de données CO2

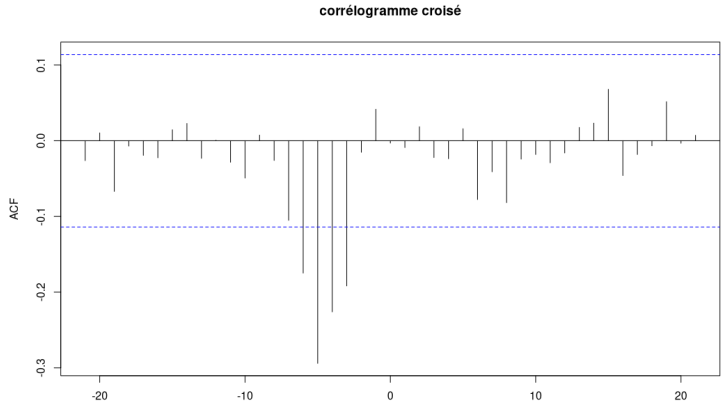
→ blanchissement de *Input Gas*.





Application sur le jeu de données CO2

→ Determiner la fonction de transfert



Application sur le jeu de données CO2

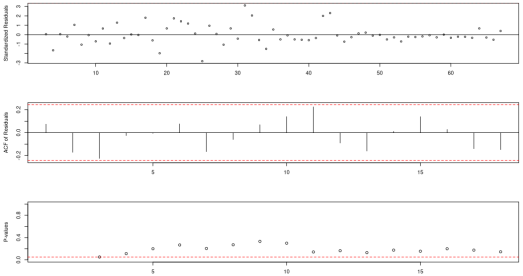
→ Le meilleur modèle est un $AR[2]$ au numérateur et un polynome de degré 4 au dénominateur

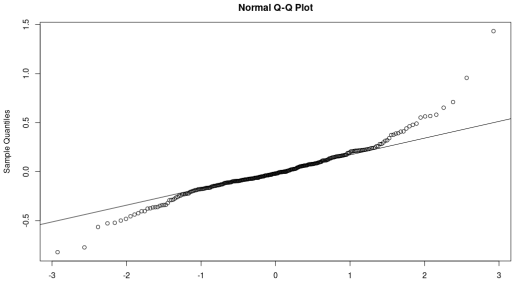
```
> output_XY=arimax(Y,order=c(2,0,0),transfer=list(c(1,5)),fixed=c(NA,NA,NA,NA,0,0,0,NA,NA,NA),xtransf=X)
> summary(output_XY)
Coefficients:
          ar1      ar2  intercept  T1-AR1  T1-MA0  T1-MA1  T1-MA2   T1-MA3   T1-MA4
      1.5272  -0.6288    53.3618  0.5490       0       0       0  -0.5310  -0.3801
s.e.  0.0467   0.0495     0.1375  0.0392       0       0       0   0.0738   0.1017
       T1-MA5
      -0.5180
s.e.   0.1086

sigma^2 estimated as 0.0571:  log likelihood = 2.08,  aic = 9.83

Training set error measures:
                      ME      RMSE       MAE          MPE      MAPE      MASE       ACF1
Training set 0.0001700879 0.2389594 0.1681788 -0.001732428 0.3130213 0.2806151 0.02877323
```

```
> coeftest(output_XY)
z test of coefficients:
           Estimate Std. Error  z value  Pr(>|z|)
ar1         1.527181   0.046723  32.6859 < 2.2e-16 ***
ar2        -0.628841   0.049471 -12.7114 < 2.2e-16 ***
intercept 53.361773   0.137503 388.0769 < 2.2e-16 ***
T1-AR1     0.549027   0.039191  14.0089 < 2.2e-16 ***
T1-MA3    -0.530964   0.073814  -7.1933 6.325e-13 ***
T1-MA4    -0.380125   0.101704  -3.7376 0.0001858 ***
T1-MA5    -0.518006   0.108562  -4.7715 1.829e-06 ***
```

Application sur le jeu de données CO2

→ Diagnostic des résidus





Conclusion

Conclusion

- Arguments de l'auteur pour une approche bayesienne
- Bayesiens contre fréquentistes
- Les avantage de l'approche bayésienne

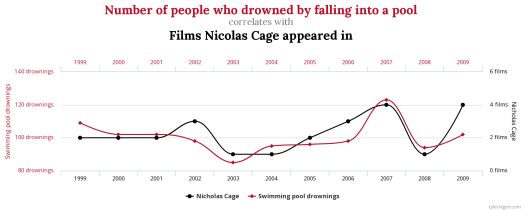


Figure – [http ://www.tylervigen.com/]

Merci de votre attention