Cyril THOMMERET
LPSM - Sorbonne Université
SAFRAN Aircraft Engines

12 juin 2023

$\boxed{\eta \neq 0 \ \neq 1}$

On génère n_{exp} données t_k et autant d'échantillons $\tilde{t}_{k,\tilde{k}}$ de taille $\tilde{n}$.

On compare ensuite pour chaque k (allant de 1 à n_{exp}), le quantile d'ordre p (ou p_i) de $(\tilde{t}_{k,\tilde{k}})_{1\leq\tilde{k}\leq\tilde{n}}$ avec la valeur t_k correspondante.

La probabilité de rejet est ainsi définie : $r := \dfrac{1}{n_{exp}} \cdot \sum_{k=1}^{n_{exp}} \mathbb{1}_{t_k \geq \tilde{t}_{k,(\lceil \tilde{n}\cdot p \rceil)}}$

1 Go !

Nous appliquons le résultat pour les mélanges à deux composantes uniformes.

Le cadre adopté dans les analyses numériques est celui où le paramétrage de la première composante est entièrement déterminé (c'est-à-dire parfaitement connu.) Autrement dit, le paramètre $\theta_1^\star$ est supposé connu.

En ce qui concerne la seconde composante, nous le supposons inconnu mais en nous pliant à la contrainte qu'il soit de dimension 1 (*i.e.* : undimensionel.)

Bien sûr la proportion $\pi^\star$ avec laquelle intervient la seconde composante sera également supposée incconue.

1.1 Algorithme de test

L'objectif de cet algorithme est double.

Il nous permettra dans un premier temps de vérifier la convergence en loi de la statistique de test avec celle du suprémum du processus gaussien. Puis ?

1.1.1 Cadre de la divergence de Kullback-Leibler *modifiée*

Compte tenu des études précédemment réalisées, un simple rappel suffirait !

Rappelons que cette divergence est liée à la fonction génératrice :

Fonction génératrice

La fonction génératrice est : $\varphi_{KL_m}(x) := -\ln x + x - 1$, d'où $\varphi_{KL_m}(x) = 1 - \frac{1}{x}$.
Ainsi,

$$\begin{aligned}\varphi^{\#}_{KL_m}(x) : &= x \cdot \varphi'_{KL_m}(x) + \varphi'_{KL_m}(x) \\ &= x \cdot (1 - \frac{1}{x}) + \ln x - x + 1 \\ &= \ln x\end{aligned}$$

Calcul de $m_{\pi,\theta_2}(x)$

$m^{KL_m}_{\pi,\theta_2}(x) := \ln g_{\pi,\theta_2}(x) - \ln g(x) \qquad (\forall x \in \operatorname{supp} g_{\pi,\theta})$

Calcul de $\hat{\pi}_n(\theta_2)$

Soit $\hat{\pi}_n(\theta_2) := \arg\max_\pi \ \mathbb{P}_n m_{\pi,\theta}$ où $\mathbb{P}_n m_{\pi,\theta} = \frac{1}{n}\sum_{i=1}^n \ m_{\pi,\theta}(X_i)$.

$$\hat{\pi}_n(\theta_2) = \arg\max_\pi \ \frac{1}{n} \cdot \sum_{i=1}^{n} \{\ln g_{\pi,\theta_1^\star,\theta_2}(x) - \ln g(x)\}$$

$$\hat{\pi}(\mathbb{X}, \eta) = \{1 + \frac{\eta}{1-\eta}\} \cdot \frac{n_-(\mathbb{X}, \eta)}{n_{exp}} - \frac{\eta}{1-\eta}$$

Calcul de $a_n(\theta_2)$

$$\begin{aligned}a_n &:= \mathbb{P}_n \, \Psi^2_{\hat{\pi}} \,/\, H_n^2 \\ &= \mathbb{P}_n \Big[(\partial_\pi \, m_{\hat{\pi},\eta})^2 \Big] \,/\, \big\{ \partial^2_\pi \, \big[\, \mathbb{P}_n \, m_{\hat{\pi},\eta} \, \big] \, \big\}^2 \qquad \text{(selon les définitions)} \\ &= \mathbb{P}_n \Big[(\partial_\pi \, m_{\hat{\pi},\eta})^2 \Big] \,/\, \Big[\mathbb{P}_n \, \partial^2_\pi \, m_{\hat{\pi},\eta} \, \Big]^2 \qquad \text{(dérivation terme à terme d'une somme finie)} \\ &= \mathbb{P}_n \Big[(\partial_\pi \, m_{\hat{\pi},\eta})^2 \Big] \,/\, \Big[\mathbb{P}_n \, - (\partial_\pi \, m_{\hat{\pi},\eta})^2 \, \Big]^2 \qquad \text{(identité de l'information de Fisher)} \\ &= \mathbb{P}_n \Big[(\partial_\pi \, m_{\hat{\pi},\eta})^2 \Big] \,/\, \Big[- \, \mathbb{P}_n \, (\partial_\pi \, m_{\hat{\pi},\eta})^2 \, \Big]^2 \\ \implies a_n \ &= \ 1 \,/\, \Big[\mathbb{P}_n \, (\partial_\pi \, m_{\hat{\pi},\eta})^2 \, \Big]\end{aligned}$$

Or, pour tout $x \in \mathbb{R}$, $\partial_\pi \, m_{\hat{\pi},\eta}(x) = \dfrac{f_2(x;\eta) - f_1(x)}{g_{\hat{\pi},\eta}(x)}$

Finalement, $a_n = \Big[\, \mathbb{P}_n \Big\{ \, \dfrac{f_2(\bullet;\eta) - f_1}{g_{\hat{\pi},\eta}} \, \Big\}^2 \, \Big]^{-1}$

$\implies a_n^{-1} = \frac{1}{n}\sum_{i=1}^n \ \Big\{ \dfrac{f_2(X_i;\theta_2) - f_1(X_i;\theta_1^\star)}{g_{\hat{\pi},\theta_1^\star,\theta_2}(X_i)} \Big\}^2$

Calcul des $b(t,t')$

$b(t,t')$ est le coefficient $(1,1)$ de la matrice,

$$(\mathbb{P}_n\, H_n^t)^{-1} \cdot \{\mathbb{P}_n\, \psi_n^t \psi_n^{t'}\} \cdot \mathbb{P}_n\, (H_n^{t'})^{-1}$$

Lorsque $\theta_1^\star$ est supposé connu, nous avons :

$$\begin{aligned} \psi_n^t &\equiv \partial_\pi\, m_{\hat{\pi}_n(t),\theta_1^\star,t} \\ H_n^t &\equiv \partial_\pi^2\, m_{\hat{\pi}_n(t),\theta_1^\star,t} \end{aligned}$$

Or,

$$\begin{aligned} \partial_\pi m_{\pi,\theta} &\equiv \frac{\partial_\pi g_{\pi,\theta}}{g_{\pi,\theta}} \\ \partial_\pi^2 m_{\pi,\theta} &\equiv -(\partial_\pi m_{\pi,\theta})^2 \end{aligned}$$

Donc,

$$b(t,t') = \frac{\mathbb{P}_n\{\partial_\pi m_{\hat{\pi}(t),t} \cdot \partial_\pi m_{\hat{\pi}(t'),t'}\}}{\mathbb{P}_n(\partial_\pi m_{\hat{\pi}(t),t})^2 \cdot \mathbb{P}_n(\partial_\pi m_{\hat{\pi}(t'),t'})^2}$$

2 Découpe

2.1 Algorithme principal

Définissons $s(\mathbb{X},\eta) := \sqrt{\dfrac{|\mathbb{X}|}{a(\mathbb{X},\eta)}} \cdot \hat{\pi}(\mathbb{X},\eta)$.

Algorithm 1: Algorithme général : Test d'homogénéité

Données a priori : ϕ, $\{f_1\}$, $\{f_2\}$, $\theta_1^\star$

Input : n, n_{exp}, $\tilde{n}$, $\mathcal{D}(\Theta_2)$, $\tilde{\mathcal{D}}(\Theta_2)$, p

1. Générer le plan d'échantillonnage $\mathbb{X}_{n_{exp},n} = \begin{bmatrix} \mathbb{X}^{(1)} \\ \mathbb{X}^{(2)} \\ \dots \\ \mathbb{X}^{(n_{exp})} \end{bmatrix} = \begin{bmatrix} X_{1,1} & X_{1,2} & \dots & X_{1,n} \\ X_{2,1} & X_{2,2} & \dots & X_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ X_{n_{exp},1} & X_{n_{exp},2} & \dots & X_{n_{exp},n} \end{bmatrix}$

   où $X_{i,j} \sim g_{\pi^\star,\theta^\star}$.

2. Générer le vecteur $(t_k)_{1\leq k\leq n_{exp}}$ où $t_k := \mathrm{Max}_{\eta\in\mathcal{D}(\Theta_2)}\, s_k(\eta)$ avec $s_k(\eta) := s(\mathbb{X}^{(k)},\eta)$.

3. Générer la matrice $[t_{k,\tilde{k}}]_{\substack{1\leq k\leq n_{exp}\\ 1\leq\tilde{k}\leq\tilde{n}}}$ où $t_{k,\tilde{k}} := \mathrm{Max}_{\eta\in\tilde{\mathcal{D}}(\Theta_2)}\, Y_\eta(\mathbb{X}^{(k)})$

$$\text{avec } (Y_\eta(\mathbb{X}))_{\eta\in\tilde{\mathcal{D}}(\Theta_2)} \sim \mathcal{N}(0,\Sigma(\mathbb{X})) \text{ et } \Sigma(\mathbb{X}) := \Big[\frac{b(\mathbb{X},\eta,\eta')}{\sqrt{a(\mathbb{X},\eta)\cdot a(\mathbb{X},\eta')}}\Big]_{\eta,\eta'\in\tilde{\mathcal{D}}(\Theta_2)}$$

5. Calculer le vecteur des probabilités de rejet $r := \dfrac{1}{n_{exp}} \cdot \sum_{k=1}^{n_{exp}} \mathbb{1}\{t_k \geq t_{k,(\lceil\tilde{n}\cdot(1-p)\rceil)}\}$

Fonction: Spécification du modèle de mélange '**spec**'

Arguments : `nomDistribution.1, nomDistribution.2, parametrage.1`
optionnels : `nombreParametres.distribution_2, positionParam.2_inconnu, valeurParam.2_connue`

- `nomDistribution.1, nomDistribution.2` : sont les noms des distributions des composantes du mélanges. Elles sont de types *string*.
  Les valeurs admises sont : `"exp", "norm", "lnorm", "weibull","unif"`.

  Exemple : `nomDistribution.1 = "unif", nomDistribution.2 = "unif"`

- `parametrage.1` : est le paramétrage (supposé connu) de la premmière composante.

  Exemples :
  `nomDistribution.1 = "norm", parametrage.1 = c(0,1)` pour une loi normale centrée réduite.
  `nomDistribution.1 = "unif", parametrage.1 = c(-1,1)` pour une loi uniforme sur $[-1, 1]$.
  `nomDistribution.1 = "exp", parametrage.1 = 0.5` pour une loi exponentielle de moyenne 2.

- `nombreParametres.distribution_2` : est le nombre de paramètre de la seconde composante (sa valeur par défaut vaut 1.)

  Exemples : `nomDistribution.2 = "norm", nombreParametres.distribution_2 = 2`

- `positionParam.2_inconnu` : est la position du paramètre inconnu invernant dans la définition des loi sous R (sa valeur par défaut vaut 1.)

  Exemples :
  `nomDistribution.1 = "norm", positionParam.2_inconnu = 1` si la moyenne est inconnue.
  `nomDistribution.1 = "norm", positionParam.2_inconnu = 1` si c'est l'écart-type qui n'est pas connu.

- `valeurParam.2_connue` : valeur du paramètre connue de la seconde composante (sa valeur par défaut est `NaN`.)

N.B. : Les arguments optionnels sont nécessaires si et seulement si la seconde composante admet deux paramètres.

Script 1 – Spécification du mélange

```r
specificationDuMelange = function(nomDistribution.1,
                                  nomDistribution.2,
                                  parametrage.1,
                                  nombreParametres.distribution_2 = 1,
                                  positionParam.2_inconnu = 1,
                                  valeurParam.2_connue = NaN){

    distributionList = c("exp", "norm", "lnorm", "weibull","unif")
    test = (nomDistribution.1 %in% distributionList) & (nomDistribution.1 %in% distributionList)
    if( test == FALSE ){ print("Erreur de saisie dans le nom d'une des composantes")}

    # PREMIERE COMPOSANTE #
    if(length(parametrage.1) == 1){
        rf1 = function(n) get(paste0("r", nomDistribution.1))(n, parametrage.1)
        df1 = function(x) get(paste0("d", nomDistribution.1))(x, parametrage.1)
    }
    if(length(parametrage.1) == 2){
        rf1 = function(n) get(paste0("r", nomDistribution.1))(n, parametrage.1[1], parametrage.1[2])
        df1 = function(x) get(paste0("d", nomDistribution.1))(x, parametrage.1[1], parametrage.1[2])
    }
    assign("rf1", rf1, envir = .GlobalEnv)
    assign("df1", df1, envir = .GlobalEnv)

    # DEUXIEME COMPOSANTE #
    if(nombreParametres.distribution_2 == 1){
        rf2 = function(n,eta) get(paste0("r", nomDistribution.2))(n, eta)
        df2 = function(x,eta) get(paste0("d", nomDistribution.2))(x, eta)
    }
    else{ # nombreParametres.distribution_2 == 2
    if(nombreParametres.distribution_2 != 2) return("Erreur: Le nombre de paramètres de la composante ne peut que prendre les valeurs 1 ou 2.")
    if(positionParam.2_inconnu == 1){
        rf2 = function(n,eta) get(paste0("r", nomDistribution.2))(n, eta, valeurParam.2_connue)
        df2 = function(x,eta) get(paste0("d", nomDistribution.2))(x, eta, valeurParam.2_connue)
    }
    else{ # positionParam.2_inconnu == 2
    if(positionParam.2_inconnu != 2) return("Erreur: La position du paramètre inconnu de la seconde composante.")
        rf2 = function(n,eta) get(paste0("r", nomDistribution.2))(n, valeurParam.2_connue, eta)
        df2 = function(x,eta) get(paste0("d", nomDistribution.2))(x, valeurParam.2_connue, eta)
    }
    }
    assign("rf2", rf2, envir = .GlobalEnv)
    assign("df2", df2, envir = .GlobalEnv)

    # FONCTIONS DU MELANGE #
    rmix = function(n,pi,eta){
        output = sample(x = c(0,1), size = n, replace = TRUE, prob = c(1-pi,pi))

        nb1 = sum(output)

        output[output == 1] = rf2(nb1,eta)
        output[output == 0] = rf1(n-nb1)

        return(output)
    }
    dmix = function(x,pi,eta) (1-pi)*df1(x) + pi*df2(x,eta)

    assign("rmix", rmix, envir = .GlobalEnv)
    assign("dmix", dmix, envir = .GlobalEnv)
}
```

Fonction: Génération du plan d'échantillonnage `MATRICE_ECHANTILLONAGE`

Données a priori : Spécification du mélange via la fonction `SPECIFICATIION_DU_MELANGE`.

Arguments : `nexp, n, pi.star, eta.star`

- `nexp` : nombre d'échantillon à générer.
- `n` : taille des échantillons.

Définit lors de l'execution : Dans l'environnement global (*i.e.* : `.GlobalEnv`) la matrice `matrice.dechantillonnage` de taille $(n_{exp} \times n)$ où pour tout i, j : $\mathbf{x}_{i,j} \sim g_{\pi^\star,\theta^\star}$.

```r
MATRICE_ECHANTILLONNAGE = function(nexp, n, pi.star, eta.star){
    if(!exists("rmix")) "La spécification du mélange n'a pas opérée."

    matrice.dechantillonnage = matrix(data = rmix(n = n*nexp, pi = pi.star, eta = eta.star), nrow = nexp, byrow = TRUE)
    assign("matrice.dechantillonnage", matrice.dechantillonnage, envir = .GlobalEnv)
}
```

Script 2 – Génération de la matrice d'échantillonnage

2.2 Cas spécifique au mélange de lois uniformes sous la divergence KL_m

Rappelons les formules précédemment calculés dans ce cadre :

$$\hat{\pi}(\mathbb{X}, \eta) = \{1 + \frac{\eta}{1-\eta}\} \cdot \frac{n_-(\mathbb{X}, \eta)}{n_{exp}} - \frac{\eta}{1-\eta}$$

$$a(\mathbb{X}, \eta) = \frac{n}{\frac{n_-(\mathbb{X}, \eta)}{\left(\frac{\eta}{1-\eta} + \hat{\pi}(\mathbb{X}, \eta)\right)^2} + \frac{n_+(\mathbb{X}, \eta)}{\left(1 - \hat{\pi}(\mathbb{X}, \eta)\right)^2}}$$

$$b(\mathbb{X}; \eta, \eta') = \frac{\mathbb{P}_n\{\partial_\pi m_{\hat{\pi}(\mathbb{X},\eta),\eta} \cdot \partial_\pi m_{\hat{\pi}(\mathbb{X},\eta'),\eta'}\}}{\mathbb{P}_n\{(\partial_\pi m_{\hat{\pi}(\mathbb{X},\eta),\eta})^2\} \cdot \mathbb{P}_n\{(\partial_\pi m_{\hat{\pi}(\mathbb{X},\eta'),\eta'})^2\}}$$

$$(\mathbb{X}, \eta) \longmapsto (\hat{\pi}(\mathbb{X}, \eta), a(\mathbb{X}, \eta), s(\mathbb{X}, \eta))$$

Fonction: Sous-fonction `CALCUL_STATISTIQUE` \\ Mélanges Uniformes & KL_m

Données a priori : $\mathbb{X}(\omega) = \mathbf{x}$

Arguments : `echantillon, eta`

- `echantillon` : le vecteur des données $\mathbf{x}$.
- `eta` : le paramètre η sous lequel les calculs sont effectués.

Sortie : `hatpi, an, s`

- `hatpi` : il s'agit de $\hat{\pi}(\mathbf{x}, \eta)$;
- `an` : correspond à $a(\mathbf{x}, \eta)$;
- `s` : renvoie à $s(\mathbf{x}, \eta)$.

Script 3 – Calcul des statistques

```
CALCUL_STATISTIQUE = function(echantillon, eta){

    n.moins = sum(echantillon < eta)
    hatpi = (1+eta/(1-eta))*n.moins/length(echantillon) - eta/(1-eta)

    n.plus = n - n.moins

    if(n.plus == 0 | n.moins == 0){
        an = 1/(1-eta)**2
    }
    else{
        denom.1 = n.moins/(eta/(1-eta) + hatpi)**2
        denom.2 = n.plus/(1-hatpi)**2
        an = n/(denom.1 + denom.2)
    }

    s = sqrt(n/an)*hatpi

    return(c(hatpi,an,s))
}

```

Fonction: `MATRICE_STATISTIQUE`

Arguments : `matrice.dechantillonnage, eta`

`matrice.dechantillonnage` : $\mathbb{X}_{n_{exp},n}$

`eta` : η

Définit lors de l'execution : Dans l'environnement global (*i.e.* : `.GlobalEnv`) la matrice `matrice.statistique`.

Script 4 – Matrice de statistiques

```
            MATRICE_STATISTIQUE = function(matrice.dechantillonnage, eta){
                matrice.statistique = t(apply(X = matrice.dechantillonnage, eta = eta, FUN = CALCUL_STATISTIQUE,
↪  MARGIN = 1))

                matrice.statistique = data.frame(matrice.statistique)
                names(matrice.statistique) = c("hatpi","an","t")

                assign("matrice.statistique", matrice.statistique, envir = .GlobalEnv)
              }
```

$$(\mathbb{X}_{n_{exp},n},\eta) = (\begin{bmatrix} \mathbb{X}^{(1)} \\ \mathbb{X}^{(2)} \\ \dots \\ \mathbb{X}^{(n_{exp})} \end{bmatrix},\eta) \longmapsto \begin{bmatrix} \hat{\pi}(\mathbb{X}^{(1)},\eta) & a(\mathbb{X}^{(1)},\eta) & s(\mathbb{X}^{(1)},\eta) \\ \hat{\pi}(\mathbb{X}^{(2)},\eta) & a(\mathbb{X}^{(2)},\eta) & s(\mathbb{X}^{(2)},\eta) \\ \vdots & \vdots & \vdots \\ \hat{\pi}(\mathbb{X}^{(n_{exp})},\eta) & a(\mathbb{X}^{(n_{exp})},\eta) & s(\mathbb{X}^{(n_{exp})},\eta) \end{bmatrix}$$

Fonction: VECTEUR_T

Script 5 – VECTEUR_T

```
            VECTEUR_T = function(matrice.dechantillonnage, discretisation.Theta_2){
                vecteur.s = sapply(X = discretisation.Theta_2, FUN = (function(eta)
↪  MATRICE_STATISTIQUE(matrice.dechantillonnage, eta)$s) )
                assign("vecteur.s", vecteur.s, envir = .GlobalEnv)
                vecteur.t = apply(X = vecteur.s, FUN = max, MARGIN = 1)
                assign("vecteur.t", vecteur.t, envir = .GlobalEnv)
            }
```

2.2.1 Matrice $[t_{k,\tilde{k}}]$

Chaque matrice $\Sigma^{(k)}$ est symétrique définie positive. Il y a donc $(nb2^2 - nb2)/2$ termes en dessous de la diagonale à calculer, plus $nb2$ termes diagonaux.

$$b(\mathbb{X}, \eta) = \frac{1}{\mathbb{P}_n\left[\{\partial_\pi \, m_{\hat{\pi}(\eta),\pi}\}^2\right]}$$