Estimation par la méthode du noyau

- Données $x_1, \ldots, x_n$, noyau K fonction symétrique p.r. 0, bornée, $\int K(u)du = 1$

$$\widehat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right), \quad x \in \mathbb{R}$$

Estimation par la méthode du noyau

- Données $x_1, \ldots, x_n$, noyau K fonction symétrique p.r. 0, bornée, $\int K(u)du = 1$

$$\widehat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right), \quad x \in \mathbb{R}$$

- Noyau gaussien

$$K(u) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{u^2}{2}\right), \quad u \in \mathbb{R}$$

Estimation par la méthode du noyau

- Données $x_1, \ldots, x_n$, noyau K fonction symétrique p.r. 0, bornée, $\int K(u)du = 1$

$$\widehat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right), \quad x \in \mathbb{R}$$

- Noyau gaussien

$$K(u) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{u^2}{2}\right), \quad u \in \mathbb{R}$$

- Pour ce noyau,

$$\widehat{f}(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{\sqrt{2\pi}h} \exp\left(-\frac{(x - x_i)^2}{2h^2}\right)$$

Estimateur à noyau décomposé

- Tout estimateur à noyau est composé d'une somme de n fonctions, soit une pour chaque observation.

Estimateur à noyau décomposé

- Tout estimateur à noyau est composé d'une somme de n fonctions, soit une pour chaque observation.
- > x = c(-1.111, -0.257, 1.797, 2.163, -2.2264, -0.949)

Estimateur à noyau décomposé

- Tout estimateur à noyau est composé d'une somme de n fonctions, soit une pour chaque observation.
- > x = c(-1.111, -0.257, 1.797, 2.163, -2.2264, -0.949)
- > K = function(x,a) dnorm(x,a,1.09)/6
  # noyau gaussien, $n = 6, h = 1.09$ (règle de Silverman)
  > b = seq(-6,6,.01) # valeurs où l'on calcule K

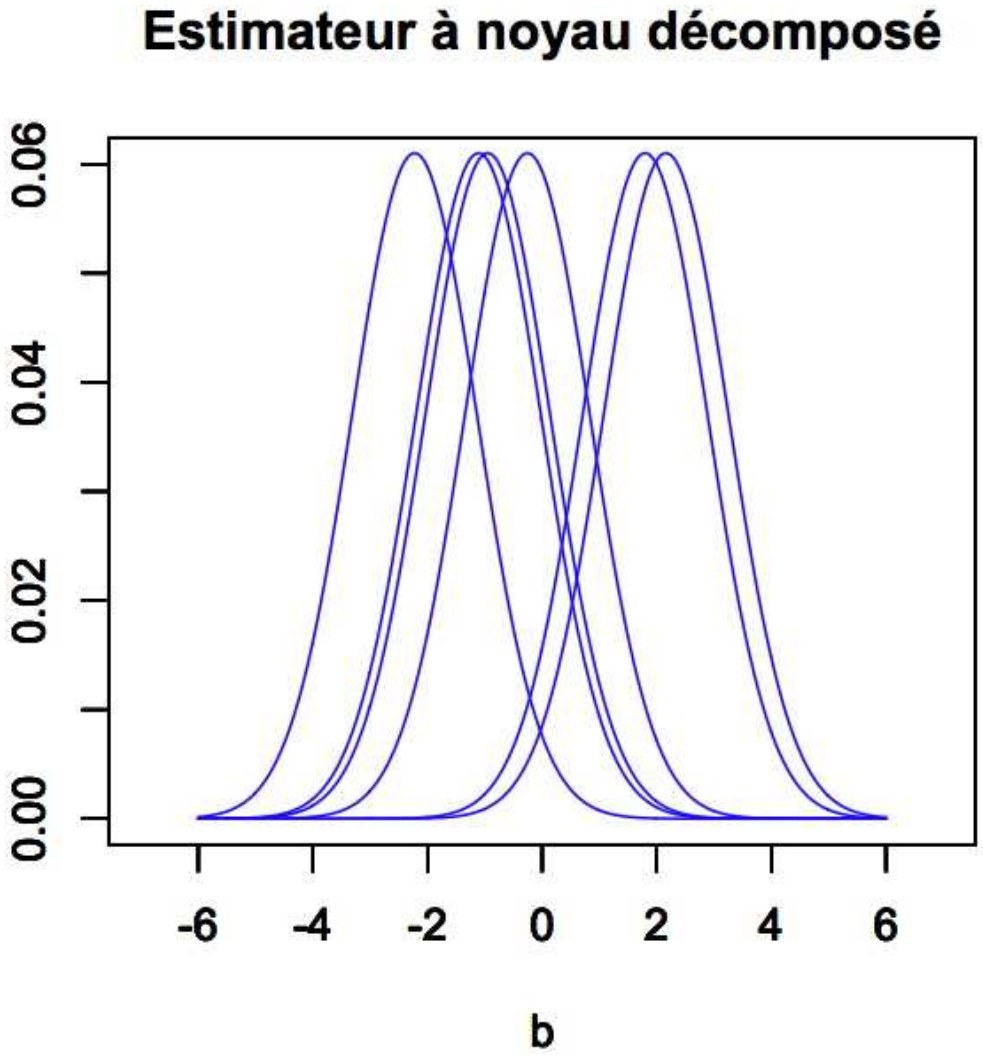
Estimateur à noyau décomposé

- Tout estimateur à noyau est composé d'une somme de n fonctions, soit une pour chaque observation.
- > x = c(-1.111, -0.257, 1.797, 2.163, -2.2264, -0.949)
- > K = function(x,a) dnorm(x,a,1.09)/6
  # noyau gaussien, $n = 6, h = 1.09$ (règle de Silverman)
  > b = seq(-6,6,.01) # valeurs où l'on calcule K
- > for (i in 1:6) {plot(b,K(b,x[i]),type="l",ylab=" ",
  xlim=c(-7,7),ylim=c(0,.06),col="blue");
  par(new=T)}
  #chaque observation détermine un graphique
  #xlim, ylim permettent une superposition parfaite des
  #graphiques

Estimateur à noyau décomposé

- Tout estimateur à noyau est composé d'une somme de n fonctions, soit une pour chaque observation.
- > x = c(-1.111, -0.257, 1.797, 2.163, -2.2264, -0.949)
- > K = function(x,a) dnorm(x,a,1.09)/6
  # noyau gaussien, $n = 6, h = 1.09$ (règle de Silverman)
  > b = seq(-6,6,.01) # valeurs où l'on calcule K
- > for (i in 1:6) {plot(b,K(b,x[i]),type="l",ylab=" ",
  xlim=c(-7,7),ylim=c(0,.06),col="blue");
  par(new=T)}
  #chaque observation détermine un graphique
  #xlim, ylim permettent une superposition parfaite des
  #graphiques
- > title("Estimateur du noyau decomposé")

Estimateur à noyau décomposé



Fonction `density`

- Forme générale de la fonction `density`
density(x, bw = "nrd0", adjust = 1, kernel = c("gaussian", "epanechnikov", "rectangular", "triangular", "biweight", "cosine", "optcosine"), weights = NULL, window = kernel, width, give.Rkern = FALSE, n = 512, from, to, cut = 3, na.rm = FALSE, ...)

Fonction `density`

- Forme générale de la fonction `density`
density(x, bw = "nrd0", adjust = 1, kernel = c("gaussian", "epanechnikov", "rectangular", "triangular", "biweight", "cosine", "optcosine"), weights = NULL, window = kernel, width, give.Rkern = FALSE, n = 512, from, to, cut = 3, na.rm = FALSE, ...)
- x #vecteur de données univariées
bw #paramètre de lissage
bw="nrd0" #règle de Silverman
bw="nrd" #règle de Scott
bw="ucv" #règle de la validation croisée,
bw="SJ-dpi" #règle de Sheather-Jones

Fonction `density`

- Forme générale de la fonction `density`
density(x, bw = "nrd0", adjust = 1, kernel = c("gaussian", "epanechnikov", "rectangular", "triangular", "biweight", "cosine", "optcosine"), weights = NULL, window = kernel, width, give.Rkern = FALSE, n = 512, from, to, cut = 3, na.rm = FALSE, ...)
- x #vecteur de données univariées
bw #paramètre de lissage
bw="nrd0" #règle de Silverman
bw="nrd" #règle de Scott
bw="ucv" #règle de la validation croisée,
bw="SJ-dpi" #règle de Sheather-Jones
- kernel #type de noyau, par défaut ”gaussian”
n=512 #n. de points équidistants où f est estimée

Sortie de `density`

- Liste de longueur 7 dont les principales composantes sont:
  [[1]] x #vecteur des points où f est estimée
  [[2]] y #vecteur des valeurs de f estimées
  Par défaut n = 512 points ou valeurs. Si n > 512, le n. de points ou de valeurs est la puissance de 2 supérieure à 512.

Sortie de `density`

- Liste de longueur 7 dont les principales composantes sont:
  [[1]] x #vecteur des points où f est estimée
  [[2]] y #vecteur des valeurs de f estimées
  Par défaut n = 512 points ou valeurs. Si n > 512, le n. de points ou de valeurs est la puissance de 2 supérieure à 512.
- [[3]] bw #paramètre de lissage h utilisé
  [[4]] n #n. de points où $\widehat{f}$ est calculé (puissance de 2)

Sortie de `density`

- Liste de longueur 7 dont les principales composantes sont:
  [[1]] x #vecteur des points où f est estimée
  [[2]] y #vecteur des valeurs de f estimées
  Par défaut n = 512 points ou valeurs. Si n > 512, le n. de points ou de valeurs est la puissance de 2 supérieure à 512.
- [[3]] bw #paramètre de lissage h utilisé
  [[4]] n #n. de points où $\widehat{f}$ est calculé (puissance de 2)
- L'application de `density` ne produit aucun graphique. On obtient plutôt de l'information sur les valeurs estimées et sur h.

Sortie de `density`

- Liste de longueur 7 dont les principales composantes sont:
  [[1]] x #vecteur des points où f est estimée
  [[2]] y #vecteur des valeurs de f estimées
  Par défaut n = 512 points ou valeurs. Si n > 512, le n. de points ou de valeurs est la puissance de 2 supérieure à 512.
- [[3]] bw #paramètre de lissage h utilisé
  [[4]] n #n. de points où $\widehat{f}$ est calculé (puissance de 2)
- L'application de `density` ne produit aucun graphique. On obtient plutôt de l'information sur les valeurs estimées et sur h.
- La fonction `plot` appliquée à l'objet créé par `density` produit le graphique.

Une application de la méthode du noyau

- > par(mfrow=c(1,2))
  > taux = scan(“cdrate.dat”)
  Jeu des taux d’épargne. Bimodal: deux types d’institutions financières.

Une application de la méthode du noyau

- > par(mfrow=c(1,2))
  > taux = scan(“cdrate.dat”)
  Jeu des taux d’épargne. Bimodal: deux types d’institutions financières.
- Paramètre de lissage par défaut
  > bw.nrd0(taux) $h = 0.1152792$ (Silverman)

Une application de la méthode du noyau

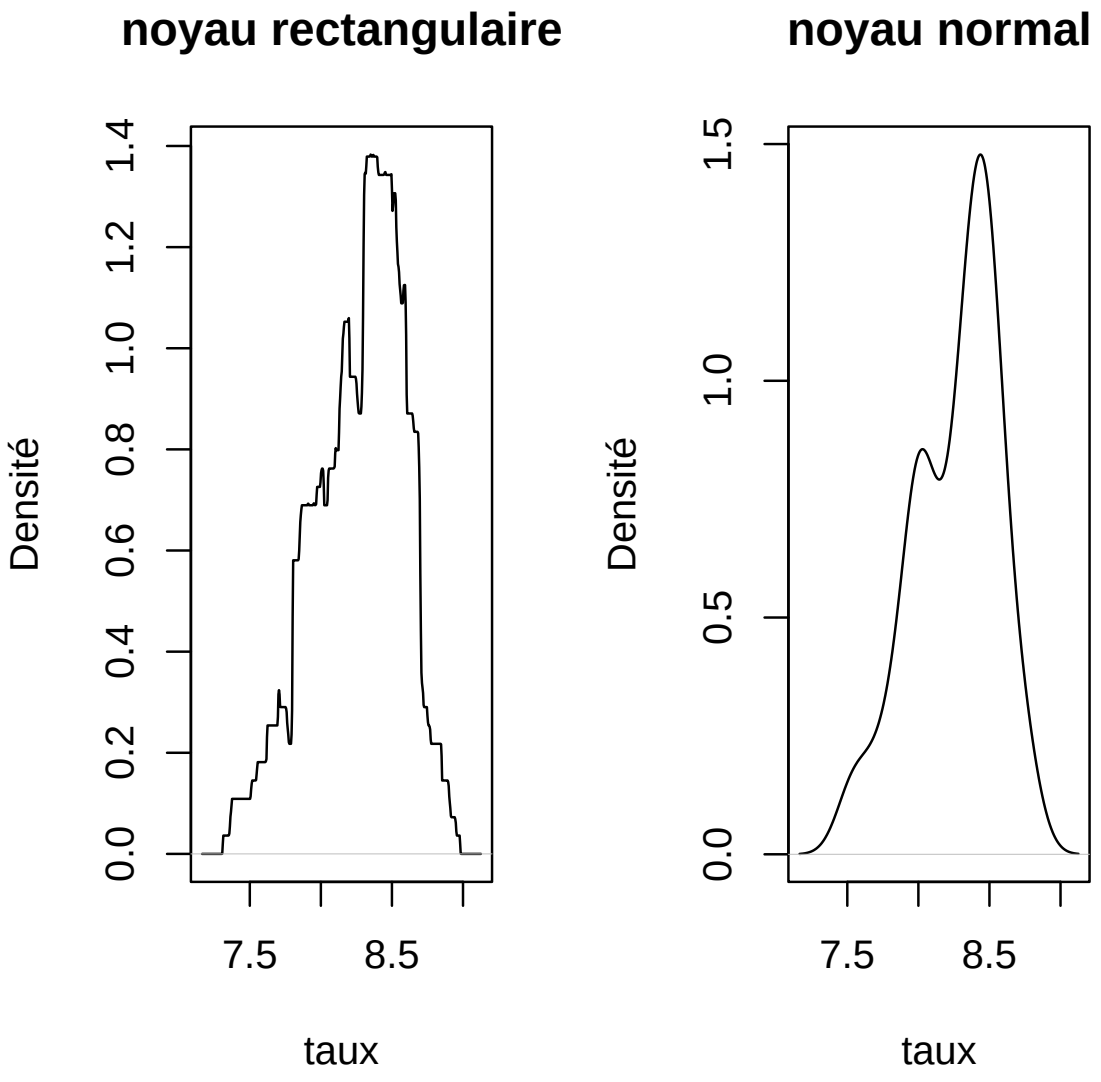
- > par(mfrow=c(1,2))
  > taux = scan(“cdrate.dat”)
  Jeu des taux d’épargne. Bimodal: deux types d’institutions financières.
- Paramètre de lissage par défaut
  > bw.nrd0(taux) $h = 0.1152792$ (Silverman)
- > bw.nrd(taux) $h = 0.1357733$ (Scott)
  > bw.SJ(taux) $h = 0.0687455$ (Sheather-Jones)

Une application de la méthode du noyau

- > par(mfrow=c(1,2))
  > taux = scan(“cdrate.dat”)
  Jeu des taux d’épargne. Bimodal: deux types d’institutions financières.
- Paramètre de lissage par défaut
  > bw.nrd0(taux) $h = 0.1152792$ (Silverman)
- > bw.nrd(taux) $h = 0.1357733$ (Scott)
  > bw.SJ(taux) $h = 0.0687455$ (Sheather-Jones)
- > plot(density(taux,kernel=”rectangular”),xlab=”taux”, ylab=”Densité”,main=”noyau rectangulaire”)
  > plot(density(taux,kernel=”gaussian”),xlab=”taux”, ylab=”Densité”,main=”noyau normal”)
  #on utilise ici le paramètre de lissage par défaut (règle de Silverman)

Deux estimateurs à noyau

- Jeu `cdrate.dat` (69 taux d'intérêt), h Silverman



Package `sm`

- Fonction `sm.density` : > library(sm)
sm.density(x, h, model = "none", weights = NA, group=NA, ...)#estimation avec le noyau gaussien

Package `sm`

- Fonction `sm.density` : > library(sm)
  sm.density(x, h, model = "none", weights = NA, group=NA, ...)#estimation avec le noyau gaussien
- Produit le graphique de l'estimateur à noyau gaussien

Package `sm`

- Fonction `sm.density` : > library(sm) sm.density(x, h, model = "none", weights = NA, group=NA, ...)#estimation avec le noyau gaussien
- Produit le graphique de l'estimateur à noyau gaussien
- x #données uni- (vecteur), bi- ou tridimensionnelles (matrice, tableau); h #paramètre de lissage (par défaut, règle de Scott)

Package `sm`

- Fonction `sm.density` : > library(sm)
  sm.density(x, h, model = "none", weights = NA, group=NA, ...)#estimation avec le noyau gaussien
- Produit le graphique de l'estimateur à noyau gaussien
- x #données uni- (vecteur), bi- ou tridimensionnelles (matrice, tableau); h #paramètre de lissage (par défaut, règle de Scott)
- Autres arguments intéressants (voir `sm.options`)
  add=TRUE #ajoute à un graphique la densité estimée
  display="se" #produit une bande de variabilité
  method #choix de h, ="normal" (Scott), "cv" (validation croisée), "sj" (Sheather-Jones)
  hmult #multiple du h, $= 1$ par défaut

Variation du paramètre de lissage

- > library(sm)

Variation du paramètre de lissage

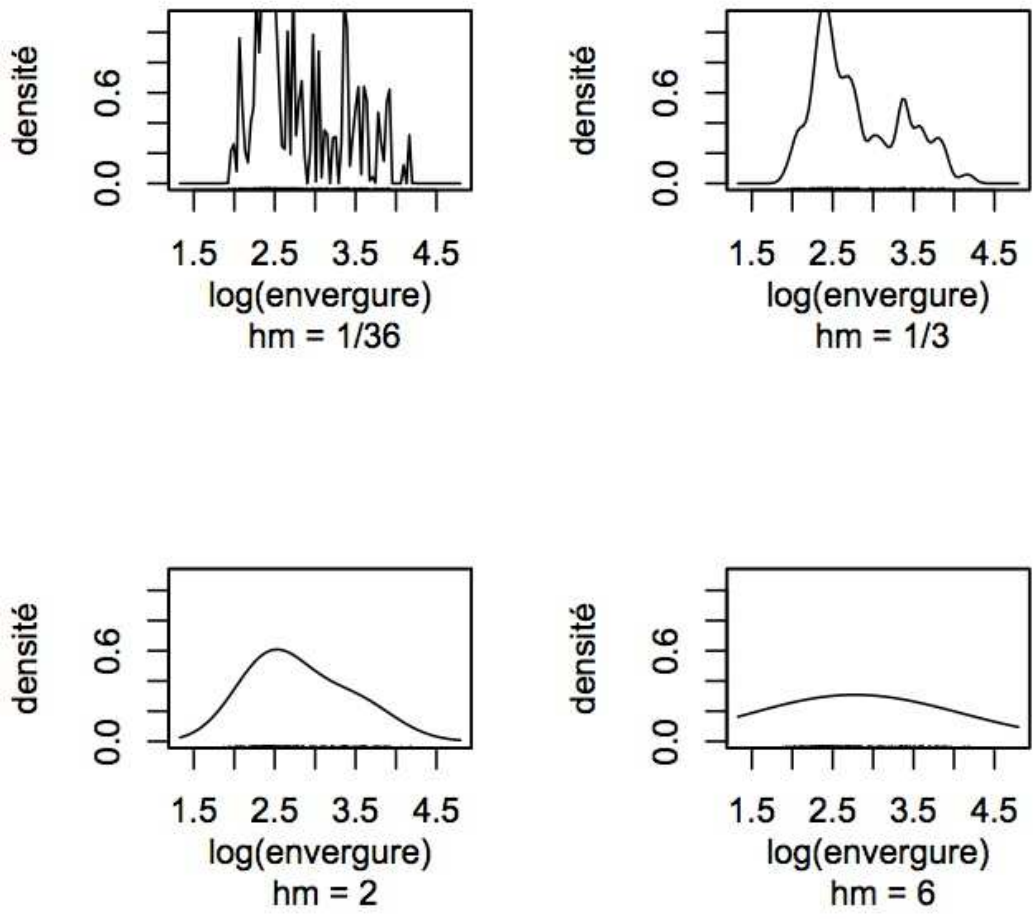
- > library(sm)
- Jeu `aircraft`: 709 modèles d'avion du 20^e siècle, tableau à 8 variables, incluant Span et Period
  > provide.data(aircraft)# commande propre à `sm`
  > y = log(Span[Period==3])#var. Span 1956–1984
  > par(mfrow=c(2,2))

Variation du paramètre de lissage

- > library(sm)
- Jeu `aircraft`: 709 modèles d'avion du 20^e siècle, tableau à 8 variables, incluant Span et Period
  > provide.data(aircraft)# commande propre à `sm`
  > y = log(Span[Period==3])#var. Span 1956–1984
  > par(mfrow=c(2,2))
- sm.density(y,hmult=1/36,xlab="log(envergure)\n hm = 1/36", ylab="densité")# hmult facteur multipliant h optimal
  sm.density(y,hmult=1/3,xlab="log(envergure)\n hm = 1/3", ylab="densité")
  sm.density(y,hmult=2,xlab="log(envergure)\n hm = 2", ylab="densité")
  sm.density(y,hmult=6,xlab="log(envergure)\n hm = 6",ylab="densité")

Noyau gaussien (variation de h)

- Densité de l'envergure de 709 modèles d'avion



Comparaison de 3 densités

- > y1 = log(Span)[Period==1]#début du 20^e siècle
  > y2 = log(Span)[Period==2]#milieu du 20^e siècle
  > y3 = log(Span)[Period==3]#fin du 20^e siècle

Comparaison de 3 densités

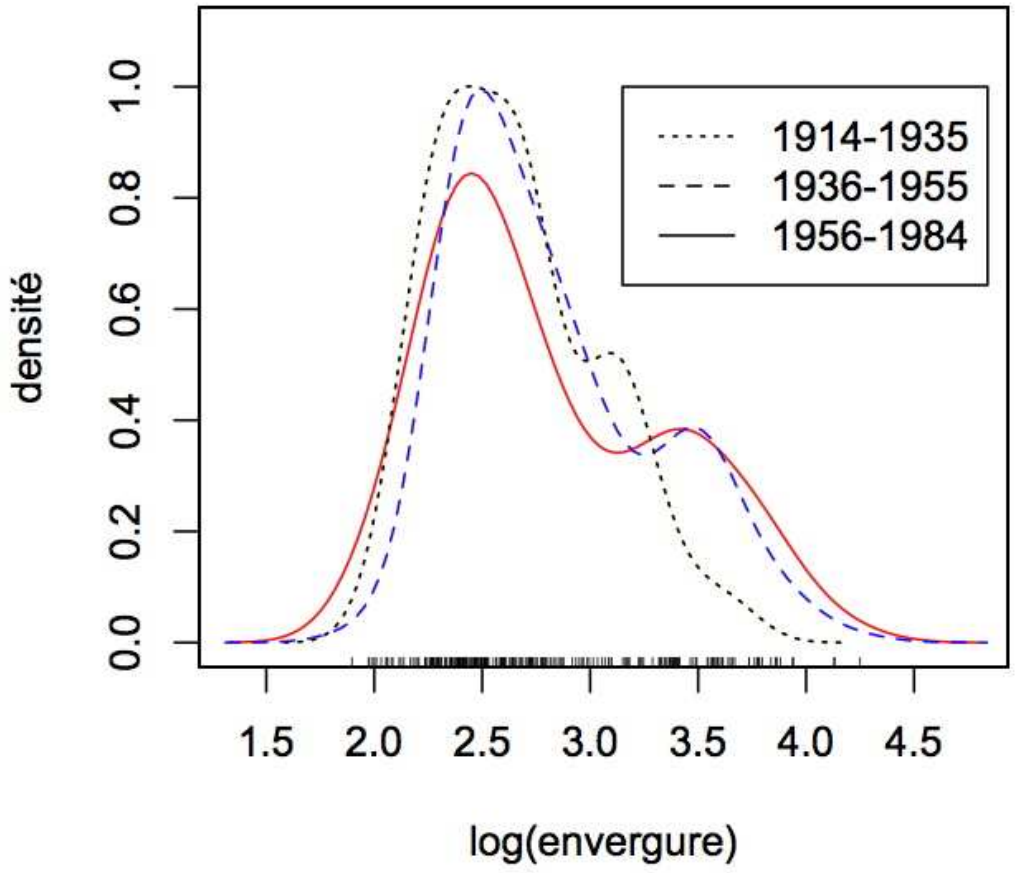
- > y1 = log(Span)[Period==1]#début du 20^e siècle
  > y2 = log(Span)[Period==2]#milieu du 20^e siècle
  > y3 = log(Span)[Period==3]#fin du 20^e siècle
- Superposition des graphiques
  > sm.density(y3,xlab="log(envergure)",ylab="densité", col="red")
  > sm.density(y2,add=T,lty=2,col="blue")
  > sm.density(y1,add=T,lty=3,col="black")

Comparaison de 3 densités

- > y1 = log(Span)[Period==1]#début du 20^e siècle
  > y2 = log(Span)[Period==2]#milieu du 20^e siècle
  > y3 = log(Span)[Period==3]#fin du 20^e siècle
- Superposition des graphiques
  > sm.density(y3,xlab="log(envergure)",ylab="densité", col="red")
  > sm.density(y2,add=T,lty=2,col="blue")
  > sm.density(y1,add=T,lty=3,col="black")
- Légende
  > legend(3.15,1,c("1914-1935","1936-1955","1956-1984"),lty=3:1)
  #3 périodes du 20^e siècle couvertes par le jeu `aircraft`

Comparaison de 3 densités (suite)

- Superposition de 3 graphiques avec `sm.density`



Précision de l'estimateur à noyau

- Estimateur biaisé

$$E[\widehat{f}(x)] \approx f(x) + \frac{h^2}{2}\sigma_K^2 f''(x)$$

Précision de l'estimateur à noyau

- Estimateur biaisé

$$E[\widehat{f}(x)] \approx f(x) + \frac{h^2}{2}\sigma_K^2 f''(x)$$

- la variance dépend de $f(x)$:

$$\mathrm{Var}[\widehat{f}(x)] \approx \frac{f(x)R(K)}{nh}$$

Précision de l'estimateur à noyau

- Estimateur biaisé

$$E[\widehat{f}(x)] \approx f(x) + \frac{h^2}{2}\sigma_K^2 f''(x)$$

- la variance dépend de $f(x)$:

$$\mathrm{Var}[\widehat{f}(x)] \approx \frac{f(x)R(K)}{nh}$$

- $\Longrightarrow$ Bande de confiance difficile à construire

Précision de l'estimateur à noyau

- Estimateur biaisé

$$E[\widehat{f}(x)] \approx f(x) + \frac{h^2}{2}\sigma_K^2 f''(x)$$

- la variance dépend de $f(x)$:

$$\mathrm{Var}[\widehat{f}(x)] \approx \frac{f(x)R(K)}{nh}$$

- $\Longrightarrow$ Bande de confiance difficile à construire
- On peut montrer que, indépendamment de f et x,

$$\mathrm{Var}\left[\sqrt{\widehat{f}(x)}\right] = E\left[\widehat{f}(x)\right] - \left(E\left[\sqrt{\widehat{f}(x)}\right]\right)^2 \approx \frac{R(K)}{4nh}.$$

Précision de l'estimateur à noyau

- Bande de variabilité de 2 écarts types p.r. à $E\left[\sqrt{\widehat{f}(x)}\right]$

$$\sqrt{\widehat{f}(x)} \pm 2\sqrt{R(K)/4nh} = \sqrt{\widehat{f}(x)} \pm \sqrt{R(K)/nh}$$

Précision de l'estimateur à noyau

- Bande de variabilité de 2 écarts types p.r. à $E\left[\sqrt{\widehat{f}(x)}\right]$

$$\sqrt{\widehat{f}(x)} \pm 2\sqrt{R(K)/4nh} = \sqrt{\widehat{f}(x)} \pm \sqrt{R(K)/nh}$$

- En élevant au carré, on peut en déduire un IC de $E[\widehat{f}(x)]$ pour tout x (bande de confiance)

Précision de l'estimateur à noyau

- Bande de variabilité de 2 écarts types p.r. à $E\left[\sqrt{\widehat{f}(x)}\right]$

$$\sqrt{\widehat{f}(x)} \pm 2\sqrt{R(K)/4nh} = \sqrt{\widehat{f}(x)} \pm \sqrt{R(K)/nh}$$

- En élevant au carré, on peut en déduire un IC de $E[\widehat{f}(x)]$ pour tout x (bande de confiance)
- $\neq$ bande de confiance pour f car $\widehat{f}$ est biaisé

Précision de l'estimateur à noyau

- Bande de variabilité de 2 écarts types p.r. à $E\left[\sqrt{\widehat{f}(x)}\right]$

$$\sqrt{\widehat{f}(x)} \pm 2\sqrt{R(K)/4nh} = \sqrt{\widehat{f}(x)} \pm \sqrt{R(K)/nh}$$

- En élevant au carré, on peut en déduire un IC de $E[\widehat{f}(x)]$ pour tout x (bande de confiance)
- $\neq$ bande de confiance pour f car $\widehat{f}$ est biaisé
- Construction avec `sm.density` et l'argument `display`

```
> provide.data(aircraft)
> y = log(Span[Period==3])
> sm.density(y,xlab="log(envergure)",ylab="densité",
display="se")
```

Bande de variabilité pour $E[\widehat{f}(\cdot)]$