Fonction de noyau et estimation d'une densité

Julien Parfait Bidias Assala

2024-08-25

Background

Intérêt

Les fonctions de noyaux jouent un rôle crucial en statistiques non paramétriques, en particulier pour l'estimation de densité via l'estimateur à noyau (ou méthode de Parzen-Rosenblatt). Cette méthode permet d'estimer la distribution d'une variable continue sans faire d'hypothèses strictes sur la forme de cette distribution, contrairement aux méthodes paramétriques où l'on suppose généralement une distribution spécifique (comme la loi normale).

Les différentes fonctions de noyaux

1. Le noyau de Rosenblatt ou noyau rectangulaire

$$K(x) = \frac{1}{2}\mathbf{1}_{[-1,1]}(x)$$

2. Le noyau de Biweight

$$K(x) = \frac{15}{16}[(1-x^2)]^2\mathbf{1}_{\{|x|\leq 1\}}$$

Les différentes fonctions de noyaux

3. Le noyau Gaussien

$$K(x) = \frac{1}{(2\pi)^{\frac{1}{2}}} \exp\{-\frac{x^2}{2}\}$$

4. Le noyau triangulaire

$$K(x) = (1 - |x|)\mathbf{1}_{\{|x|\leq 1\}}$$

5. Le noyau de Triweight

$$\frac{35}{32}(1 - x^2)^3\mathbf{1}_{\{|x|\leq 1\}}$$

Les différentes fonctions de noyaux

6. Le noyau cosinus

$$\frac{\pi}{4} cos(\frac{x\pi}{2}) \mathbf{1}_{[-1,1]}(x)$$

7. Le noyau d'Epanechnitov

$$\frac{3}{4}(1-x^2)\mathbf{1}_{\{|x|\leq 1\}}$$

Calcul des différents noyaux avec le logiciel R

Pour représenter ces fonctions nous allons d'abord commencer par les créer.

```
rosenblat = function(x){
  K= (1/2)*ifelse((1-x^2 >=0), 1, 0)
  return(K)
}

biweight = function(x){
  K = (16/15)*((1-x^2)^2)*ifelse((1-x^2>=0), 1, 0)
  return(K)
}

gaussien = function(x){
  K = (1/sqrt(2*pi))*exp(-((x^2)/2))
  return(K)}
```

Calcul des différents noyaux

La fonction **ifelse** permet de retourner une valeur suivant une condition. Elle très pratique si vous voulez programmer des fonctions indicatrices. Par exemple, le premier code signifie qu'on crée une fonction qui prend x comme paramètre, ensuite on multiplie $1 - |x|$ par une fonction qui retourne 1 si $|x| \leq 1$ c'est-à-dire : $1 - x^2 \geq 0$ et 0 sinon.

```
triangulaire = function(x){
  K = (1-abs(x))*ifelse((1-x^2>=0), 1, 0)
  return(K)
}

triweight = function(x){
  K = (35/12)*((1-x^2)^3)*ifelse((1-x^2>=0), 1, 0)
  return(K)
}
```

Calcul des différents noyaux

```
ncosinus =function(x){
  K = (pi/4)*(cos((x*pi)/2))*ifelse((1-x^2>=0), 1, 0)
  return(K)}

epanechnikov =function(x){
  K = (3/4)*(1-x^2)*ifelse((1-x^2>=0), 1, 0)
  return(K)
}
```

Représentation graphique

Pour la représentation graphique nous allons considérer un ensemble de 1000 valeurs comprises entre -2 et 2 que nous mettons dans un objet x. Ensuite, nous prenons $h = 0,9$ pour toutes les fonctions de noyau.

```
# Ensemble de valeurs x
x = seq(-2, 2, length.out = 1000)
# On applique nos fonctions au point x
absc_gaussien      = gaussien(x / 0.9)
absc_epanechnikov = epanechnikov(x / 0.9)
absc_triangulaire = triangulaire(x / 0.9)
absc_biweight      = biweight(x / 0.9)
absc_cosinus       = ncosinus(x / 0.9)
absc_triweight     = triweight(x / 0.9)
absc_rosenblat     = rosenblat(x / 0.9)
```

Représentation graphique

Nous combinons le tout dans un object de type data frame ou chaque fonction de noyau est associée à son nom en colonne. L'idée étant de pouvoir utiliser ggplot2 pour le graphe.

```
# On importe la libraire necessaire
library(ggplot2)
# On cree le dataframe
base_combine = data.frame(x = x,
                  gaussien     = absc_gaussien,
                  epanechnikov = absc_epanechnikov,
                  triangulaire = absc_triangulaire,
                  biweight     = absc_biweight,
                  cosinus      = absc_cosinus,
                  triweight    = absc_triweight,
                  rosenblat   = absc_rosenblat)
```

D'où la code ci-dessous pour la représentation graphique :

Représentation graphique

```
graphe = ggplot(base_combine, aes(x = x)) +
  geom_line(aes(y = gaussien, color = "Gaussien")) +
  geom_line(aes(y = epanechnikov, color = "Epanechnikov")) +
  geom_line(aes(y = triangulaire, color = "Triangulaire")) +
  geom_line(aes(y = biweight, color = "Biweight")) +
  geom_line(aes(y = cosinus, color = "Cosinus")) +
  geom_line(aes(y = triweight, color = "Triweight")) +
  geom_line(aes(y = rosenblat, color = "Rosenblatt")) +
  labs(title = "Fonctions de noyau",x = "x",y = "Valeur") +
  scale_color_manual(
    values = c("Gaussien" = "blue",
               "Epanechnikov" = "red", "Triangulaire" = "green
               "Biweight" = "purple",
               "Cosinus" = "orange", "Triweight" = "brown",
               "Rosenblatt" = "pink")) +
  theme_gray()
```

Représentation graphique

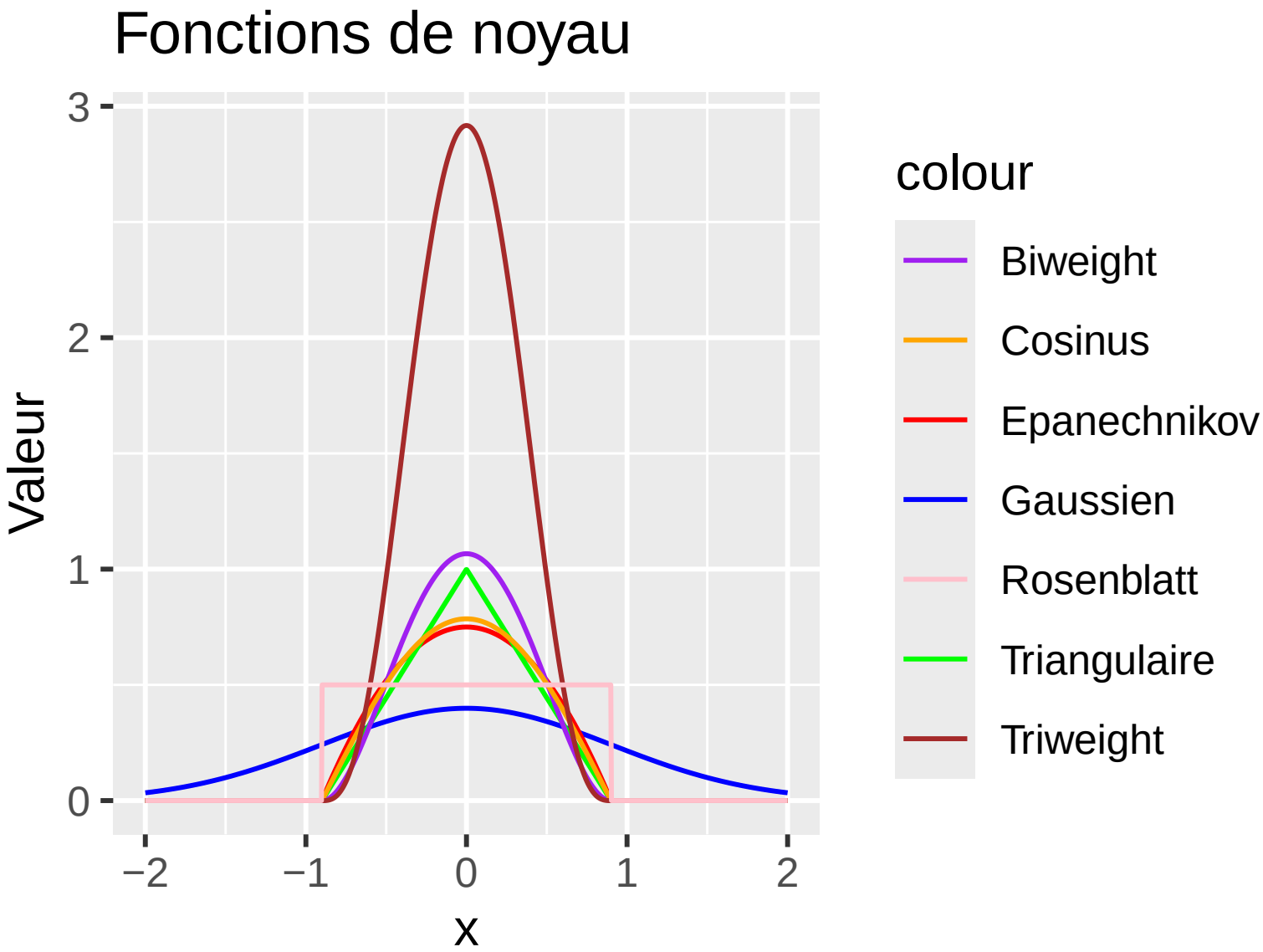


Figure 1: Les différents noyaux

Application à l'estimation d'une densité

Pour chaque fonction de noyau, on peut effectuer une estimation d'une densité. Par exemple, la distribution des salaires d'une entreprise. Pour effectuer une estimation de la densité pour chaque fonction, nous allons utiliser l'estimateur à noyau. Il s'écrit :

$$\hat{f}_n(x) = \frac{1}{nh} \sum_{i=1}^{n} K(\frac{x - X_i}{h})$$

Où h : Largeur de bande (ou paramètre de lissage) qui contrôle la finesse du lissage de la densité. n la taille de l'échantillon. $K()$: Fonction de noyau, X_i : Les observations de l'échantillon.

Construction de la fonction

```
estimation.densite = function(data, x, h, noyau) {
  n = nrow(data)
  k = length(x)
  fx =  rep(0, k)
  for (i in 1:k){
    somme=0
      for(j in 1:n){
    calcule           =  (x[i]- data[j]) / h
    valeur.noyau      = noyau(calcule)
    somme = somme + valeur.noyau
    fx[i] = (1 / (n * h)) * somme
  }
  }

  return(fx)
}
```

Application avec données réelles

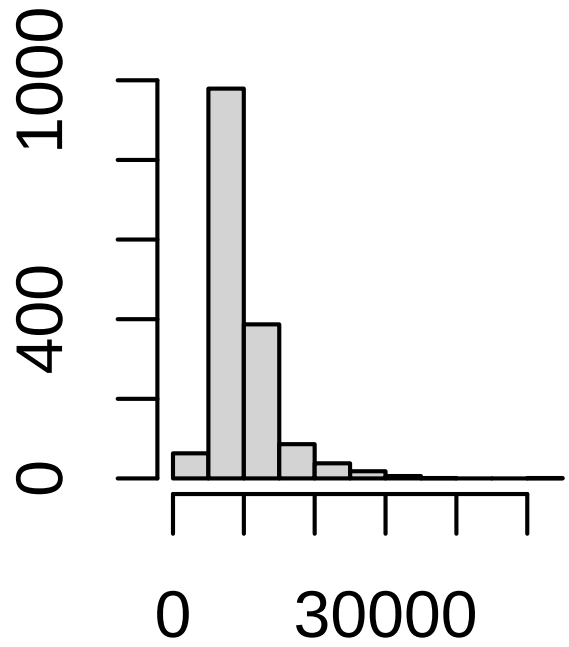
Dans cette section on montre comment la densité d'une variable est généralement estimée par les logiciels statistiques. Supposons qu'on dispose d'une distribution des salaires. On va effectuer la représentation graphique de la densité de notre distribution des salaires après l'avoir centrée et réduite. Le tableau ci-dessous présente les 6 premières lignes de la base de données.

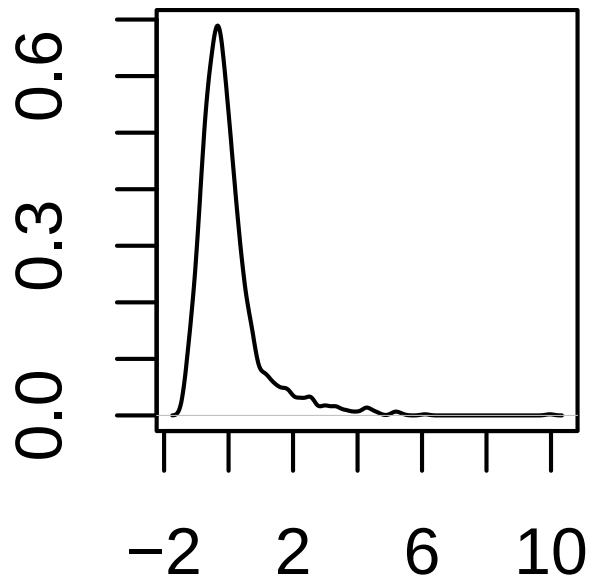
Table 1: Salaires

x
8977.08
8714.75
6920.92
10833.33
10747.83
8000.00

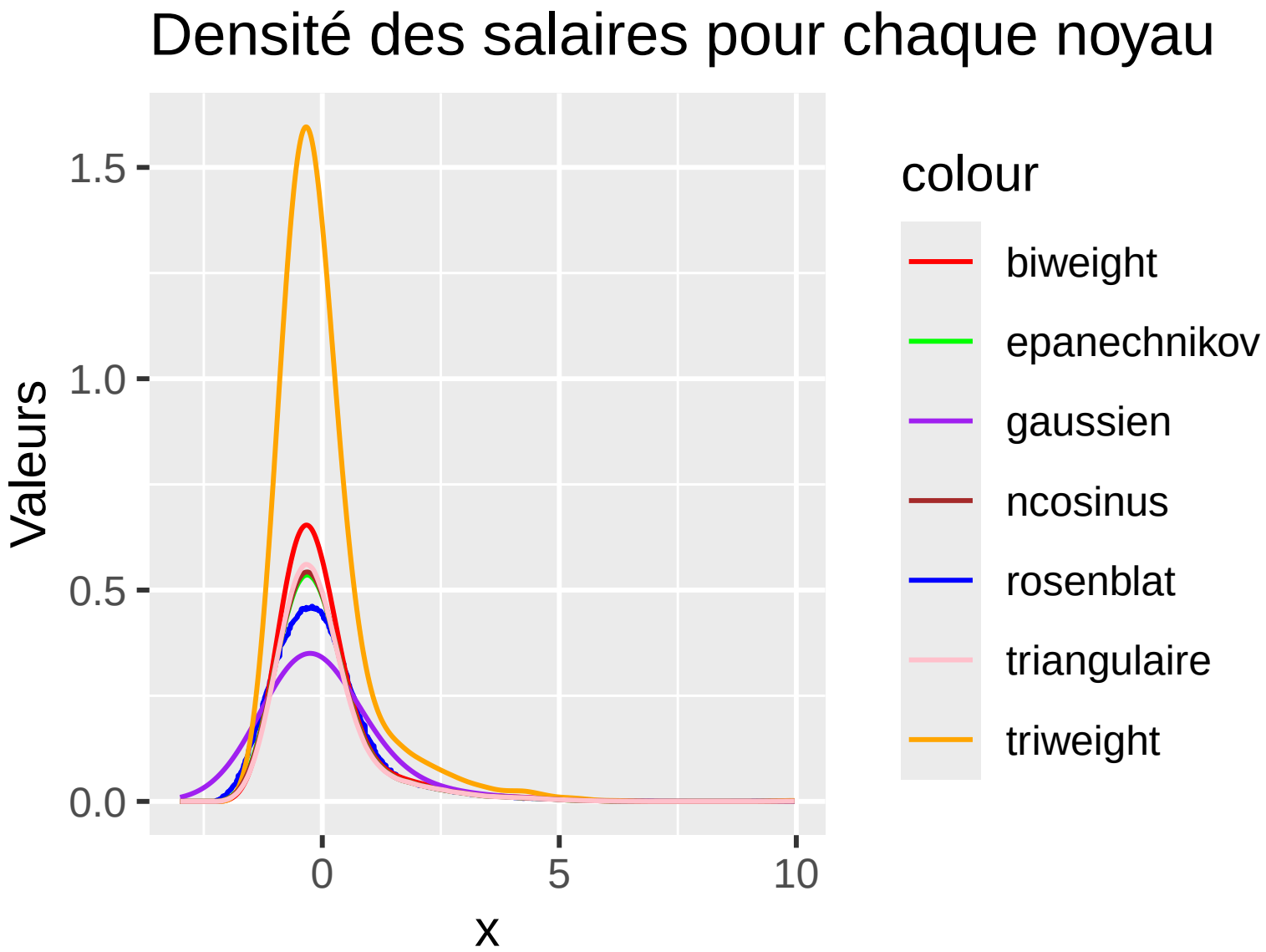
Représentation de la base centrée réduite

A gauche la distribution des salaires et et à droite la distribution des salaires centrée réduite.





Application avec données réelles



Interprétation

End

La densité d'une distribution de salaires est le plus souvent déséquilibrée (elle n'est pas symétrique) du fait de la présence de valeurs extrêmes ou encore parce que les salaires diffèrent toujours dans une entreprise. Il y aura toujours des salaires élevés et d'autres non. D'où le graphe ci-dessous. Le noyau adéquat pour la distribution des salaires est celui qui à une courbe plus lissée et bien formée. Mais cela peut varier en fonction de la valeur de la fenêtre optimale et d'autres paramètres.