Méthode de programmation des fonctions renvoyant des fonctions : Application au package SPlmficmcm et son app-shiny.

Molière Nguile-Makao, Vincent Vradet
&
Alexandre Bureau

Unité méthodologique du Centre Clinique et Évaluative en Oncologie
Hôtel-Dieu de Québec Laval Université

26 mai 2017

Plan

1. Concept/Interêts
   - Concept/Intérêt
   - Illustration
2. Application au package SPmlficmcm
   - Problématique du SPmlficmcm
   - Package SPmlficmcm
3. SPmlfcmcm Shiny-App

Concept/intérêts

- **Programmation R**
  - Idée générale

    - **Arguments**
      →(Scalaires, Caractères,· · ·)
    - **Résultats**
      Objet (Scalaires, Caractères, · · ·)

- **Concept : Fonctions par des fonctions**
  - Autre façon de programmer en R

    - **Arguments**
      → (Scalaires, Caractères, Fonctions· · ·)
    - **Résultats**
      Objet (Scalaires, Caractères, Programme R, Fonction,· · ·)

- **Intérêts**
  - Meilleur contrôle avec une fonction renvoyée
  - Rendre des programmes portables

Illustration 1

- **Exemple** : Estimation paramétrique de la fonction de vraisemblance.
  - Générer une variable de survie $T \sim weibull$

Programme

```
# Simulation de la variable de Survie
set.seed(13200)
X1<-rweibull(500,1.5)
X2<-rweibull(500,4)
C=ifelse(X1<X2,1,0)
T=ifelse(C==1,X1,X2)
obs<-100+c(1:length(C))
dat<-data.frame(obs,T,C)
head(dat)
```

```
  obs         T C
1 101 0.9365323 1
2 102 0.2322323 1
3 103 0.5450104 1
4 104 0.8810166 0
5 105 0.2266389 1
6 106 0.5157231 1
```

Equation de log-vraisemblance

$$L(\theta) = -\sum_i (c_i log(f(t_i,\theta)) + (1-c_i) log(S(t_i,\theta)))$$

Programme qui retourne la fonction de vraissemblance

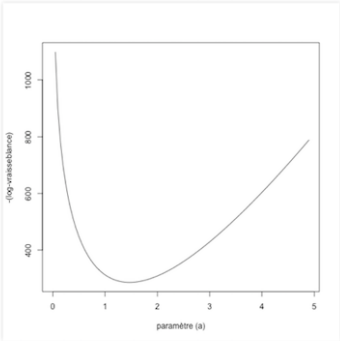
```
# Programme d'estimation du paramètre
ProSurv<-function(dat){
    # Etape 1 construction de la fct de vraisemblance
    logV<-Vectorize(function(a){
                    return((-1)*sum(dat[["C"]]*log(dweibull(dat[["T"]],a))
                    +(1-dat[["C"]])*log(1-pweibull(dat[["T"]],a)),na.rm = FALSE))
    },'a')
    # retourne la fonction de vraisemblance
    return(logV)
}
```

Illustration 1(suite)

- **Exemple** : Estimation paramétrique de la fonction de vraisemblance.
  - Représentation de la fonction de log-vraisemblance

Représentation graphique de la fonction de -log-vraisemblance

```
# Application
likf<-ProSurv(dat)

curve(likf(x),xlim=c(0,4.9),xlab="paramètre (a)",ylab="-(log-vraisseblance)")
```



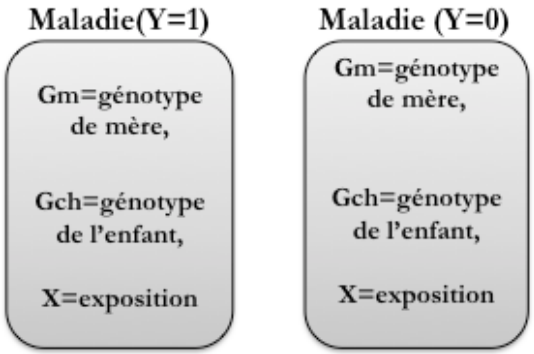
Optimisation du paramètre θ

```
nlm(likf,0.1)
```

```
$minimum
[1] 286.4191

$estimate
[1] 1.467679

$gradient
[1] 0.0001925276

$code
[1] 1

$iterations
[1] 12
```

Problématique

- **Objectif :**
  - Analyser l'impact de l'intéraction gène-environnement sur la survenue des maladies obstétriques dans un design couple mère-enfant cas et mère-enfant témoins.
  - Design couple mère-enfant cas et témoins



- **Modèle de Regréssion logistique**
  - Inéficace : problème du lien parental (genotype enfant 1/2 génotype mère.)
- **Nouvelle approche : Modèle semi-paramétrique**
  - Modèle proposé par Chen et al (2012)
  - Extension proposé par M. Nguile-Makao et A. Bureau (2015)

Modèle semi-paramétrique

- **Modèle de Chen et al (2012)**
  - Modèle sans données manquantes

$$\begin{aligned} P_{ijmc}(\beta) &= Pr(Y = i|X = j, G^M = m, G^C = c; \beta) \\ P_{c|m}(\theta) &= Pr(G^C = c|G^M = m; \theta) \end{aligned} \tag{1}$$

  - Ecriture de la log-vraisemblance : $(Y = i, X = j, G^M = m)$

$$\ell(\beta, \theta, \delta_{jm}, u_{im}(\beta, \theta))$$

avec $\delta_{jm} = Pr(X = j|G^M = m)$

- **Estimation de β et θ**
  - Etape 1 estimation de δ_{jm} utilisant le multiplicateur lagrangien $\sum_j \delta_{jm} = 1$.
  - Etape 2 Resolution du U_{im}.

$$\forall i, m \quad u_{im}(\beta, \theta) = 1 - \frac{d_i P_m(\theta)}{N_m^*(\beta, \theta) Pr(Y = i)}, \tag{2}$$

  - Etape 3 Ecriture de la fonction de vraisemblance

$$\ell^p(\beta, \theta, \widehat{u}_{im}(\beta, \theta)) \tag{3}$$

Modèle semi-paramétrique

- **Extension de M. Nguile-Makao et A. Bureau (2015)**
  - Modèle avec les données manquantes sur le génotype de l'enfant

$$
\begin{aligned}
l^p(\beta,\theta,\widehat{u}_{im}(\eta)) = & \sum_{ijmc\in S^1_{IJMC}} n_{ijmc}\log(h_{ijmc}(\eta)) + \sum_{ijm\in S^2_{IJM}} \overline{n}_{ijm}\log\left(\sum_{c\in\mathbf{G}^M} h_{ijmc}(\eta)\right) \\
& + \sum_{jm} C_{jm}\log\left(\frac{P_m(\theta)}{N^*_m(\eta)f_{jm}(\eta,\widehat{u}_m(\eta))}\right) \qquad (4) \\
& + \sum_{i}\left[d_i\log\left\{\sum_m \frac{P_m(\theta)}{N^*_m(\eta)}\sum_j \frac{C_{jm}}{f_{jm}(\eta,\widehat{u}_m(\eta))}h_{ijm}(\eta)\right\}\right].
\end{aligned}
$$

- **Estimation de θ et β**
  - Procédure d'estimation identique.

Fonction principale Spmlficmcm

- **Arguments**
  - Arguments de la fonction

```
#=====================================================
# Arguments
 Spmlficmcm <-
 function(fl,N,gmname,gcname,DatfE,typ,start,p=NULL)
#=====================================================
```

- **Corps du programme**
  - Etape 1 : Estimation via des fonctions `Est.Inpar`

```
#============================================================
# Arguments
Est.Inpar <-
function(fl,N,gnma,gnch,tab1,typ,p=NULL){
#============================================================

#============================================================
# Calcul de la valeur initiale de la distribution du genotype
d=c(N0/n0,N1/n1)
# Ecriture de l equation (3)
fgt<-fgp_mf1(tab,d,gmname=gnma,gcname=gnch,outc=outc)
ss <- nleqslv(0.1,fgt)
theta.start=ss$x;
#============================================================
# solution des parametres finaux
parms=c(beta.start,theta.start)
# calcul des valeurs intiales du systeme non lineair
lst_Matin2<-fct_invCap(tab,N,outc,vrze,gnma,gnch,theta.start)
Mat.sup=lst_Matin2$brs
#============================================================
```

<fgp_mf1> et <fct_invCap> Fonctions qui renvoient des fonctions.

Fonction principale Spmlficmcm

- **Corps du programme**
  - Etape 1 : Estimation des valeurs initiales via la fonction (suite)

```
#=========================================================================
# Etape 1
# Conditions verifiées
# Estimation des paramètres initiaux
            if(typ==1){
            vIn<-Est.Inpar(fl,N,gmname,gcname,DatfE,1)
            }else{
            vIn<-Est.Inpar(fl,N,gmname,gcname,DatfE,2)
             }
  if(missing(start)){parms<-vIn$parms
                          }else{parms<-start}
  p = length(parms)
  beta.start=parms[1:(p-1)];
  theta.start=parms[p]
# Valeurs initiales du systeme d'équation non-linéaire
  ma.u<-vIn$ma.u
  vecma.u=c(ma.u[1,],ma.u[2,])

#=========================================================================
```

  - Etape 2 : Construction et resolution du système non linéaire

```
# Valeurs initiales du systeme d quation non linaire
ma.u<-vIn$ma.u
vecma.u=c(ma.u[1,],ma.u[2,])
#=============================================================
# Etape 2 resolution du systeme d equation
RSeq<-Nlsysteq(fl,DatfE,N,gmname,gcname,yname,beta.start,theta.start)
SS<-nleqslv(vecma.u,RSeq)
vecma.u<-SS$x
#-------------------------------------------------------------
```

  `<Nlsysteq>` : renvoie une fonction qui dépend des U_{im}

Fonction principale Spmlficmcm

- **Corps du programme (suite)**
  - Ecriture de la log-vraissemblance

```
#=================================================================
# Etape 3 ecriture de la vraisemblance profile
if(typ==1){
  ftlh<-ft_likhoodCas1(fl,DatfE,N,gmname,gcname,yname,vecma.u)
          }else{
  ftlh<-ft_likhoodCasM(fl,DatfE,N,gmname,gcname,yname,vecma.u)
                }
#=================================================================
```

  - Estimation des paramètres

```
#=========================================================================

# Etape 4
# Estimation des paramètres
# Construction de la fonction gradient
  if(typ==1){
          fctgrad<-ft_gradientCas1(fl,DatfE,N,gmname,gcname,yname,vecma.u)
   }else{
        fctgrad<-ft_gradientCasM(fl,DatfE,N,gmname,gcname,yname,vecma.u)
  }
  Grad<-fctgrad(parms)
# Calcul de la hessian
  delta <- 1e-5;
  hess <- matrix(0, p, p);
  for(gg in 1:p){
  delta_ggamma <- parms;
  delta_ggamma[gg]<- delta_ggamma[gg] + delta;
  hess[, gg]<-(fctgrad(delta_ggamma) - Grad)/delta;
  }

# Estimateur des parametres
  Parms.est=parms-solve(hess)%*%Grad

# Calcul de la variance
  Grad1<-fctgrad(Parms.est)
  Hes1 <- matrix(0, p, p);
  for(gg in 1:p){
  delta_ggamma <- Parms.est;
  delta_ggamma[gg] <- delta_ggamma[gg] + delta;
  Hes1[, gg] <- (fctgrad(delta_ggamma) - Grad1)/delta;
  }
 matv<-(-1)*solve(Hes1)
 var.par<-sqrt(diag(matv))

 #=========================================================================
```

Fonction principale Spmlficmcm

- **Valeurs rétournées**
  - Résultats

```
#=============================================================================
# Etape 5
# Préparation des resultats
  mats<-cbind(Parms.est,var.par)
  nma<-c("Intercept",attr(terms.formula(fl),"term.labels"),"theta")
  nac<-c("Estimate","Std.Error")
  colnames(mats)<-nac
  rownames(mats)<-nma
  loglik<-ftlh(mats[,"Estimate"])
  rr<-list(N=N,Uim=vecma.u,MatR=mats,Matv=matv,Lhft=ftlh,Value_loglikh=loglik)
  return(rr)
 }
#====================== Fin ===================================================
```

  - La fonction de vraisemblance renvoyée <Lhft> dépend du paramètre $\eta = c(\beta, \theta)$

Application shiny : app_SPmlficmcm

- https://moli12.shinyapps.io/app_SPmlficmcm/

...


Merci ...
mortderire.com