

La régression logistique en épidémiologie

Code Stata

Introduction	3
Commandes préalables	3
Fonctions (.ado files)	4
Fonction logit_crb.ado.....	4
Fonction ORcl_fp.ado	5
Fonction ORcl_fp.ado	5
Fonction chi2_res.ado.....	6
Chapitre 1	7
Chapitre 2	8
II. Estimation d'un pourcentage par la méthode du maximum de vraisemblance.....	8
III. Application au modèle logistique	8
IV. Tests des paramètres du modèle logistique.....	8
VI. Annexe 2 : Les 3 tests issus de la méthode du maximum de vraisemblance	9
Chapitre 3	11
I. Règle générale d'interprétation du coefficient d'une variable (tableau 1)	11
III. Variable qualitative nominale à plus de 2 classes	11
IV. Variable qualitative ordinale	11
IV.1. Modélisation de l'association GEU – tabac	11
IV.2. Choix des valeurs de X.....	12
IV.3. Choix entre variables indicatrices et modélisation linéaire (test de linéarité)	12
VI. Prise en compte d'une interaction	13
Chapitre 4	14
IV. Courbes lowess avec 2 valeurs de bandwidth	14
V. Graphe des logits observés pour la régression accouchement en fonction de l'âge	14
VI. Modélisation avec une fonction en escalier.....	14
VII. Modélisation avec des polynômes.....	16
VIII. Modélisation avec des polynômes fractionnaires	16
VIII.2 Choix des puissances d'un polynôme fractionnaire de degré donnéFP	16
VIII.3. Choix du meilleur polynôme fractionnaire pour une variable : Tableau 5.....	17
IX. Modélisation avec des fonctions splines	17
IX.4 Splines linéaires (Figures 25 et 26)	17
IX.5. Splines cubiques (Figure 27).....	18
IX.6 Splines cubiques restreints (Figure 28)	18
IX.7. Choix du modèle avec des fonctions splines (tableaux 8 et 9).....	18

X. Présentation des résultats issus de la modélisation	18
X.3. Présentation quantitative des résultats dans un tableau (tableau 11).....	18
XII.2 Représentation graphique des données observées avec la courbe modélisée (Figure 31)	19
Chapitre 5	20
IV.3 Colinéarité	20
V.3. Sélection fondée sur le changement de l'estimation de l'odds-ratio.....	20
V.4. Méthodes de sélection pas à pas (stepwise).....	20
Chapitre 6	21
II. Régression logistique multinomiale	21
II.2. Exemple et interprétation des résultats.....	21
II.3. Régression logistique multinomiale versus plusieurs régressions binomiales.....	21
II.4. Comparaison des OR associés à X selon les catégories de Y	21
II.5. Changement de classe de référence pour Y	21
IV. Modèle cumulative-odds.....	21
IV.2. Exemple : Fragilité chez les personnes vivant avec le VIH	22
IV.3. Résultats avec l'hypothèse des odds proportionnels.....	22
IV.4. Regroupement de classes de Y et test de l'hypothèse des odds proportionnels	22
IV.5. Présentation des résultats	22
IV.6. Plusieurs variables indépendantes	22
V. Modèle continuation-ratio	22
V.2. Test de l'hypothèse des odds proportionnels	22
V.3. Modélisation de la durée d'infécondité	23
VI. Modèle adjacent-category	23
VII. Choix du modèle.....	23
VII.2. Un peu d'humilité sur l'importance du choix ...	23
Chapitre 7	24
II. Tests d'adéquation.....	24
III.3. Exemple	24
IV. Courbes ROC	24
IV.1. Régression logistique et classement	24
IV.2. Aire sous la courbe ROC	24
V. Diagnostics de régression	24
V.2. Influence d'un profil sur les statistiques d'adéquation	24
V.3. Influence sur l'estimation des coefficients	25
V.4. Examens des points influents.....	25

Introduction

Ce document donne les codes Stata des résultats et figures de chaque chapitre du livre "La régression logistique en épidémiologie" édité par EDP Sciences. Il ne contient que les codes des commandes Stata, les sorties figurent dans les chapitres eux-mêmes.

Les commandes d'enregistrement des figures sur disque ("graph export ...") sont le plus souvent précédées d'une * qui marque une ligne de commentaire dans Stata. Il faut enlever cette * pour que la figure soit effectivement enregistrée.

De façon générale, les commentaires dans Stata sont indiqués de la façon suivante =

- * : toute la ligne qui suit est un commentaire
- /* puis */ : tout ce qui entre ces deux couples de signes est un commentaire
- // : quand le double slash précédé d'un blanc est en fin de ligne tout ce qui suit sur la ligne est un commentaire
- /// : quand le triple slash précédé d'un blanc est en fin de ligne tout ce qui suit sur la ligne est un commentaire et le saut de ligne est ignoré (c'est-à-dire que la commande se poursuit à la ligne suivante)

Les bases de données nécessaires sont disponibles à l'adresse indiquée dans le chapitre 1 du livre.

<https://laboutique.edpsciences.fr/produit/1504/9782759838189/la-regression-logistique-en-epidemiologie>.

Elles existent selon différents formats. Le plus pratique pour une exploitation à l'aide de Stata est le format .dta : Stata est directement ouvert en double-cliquant sur le fichier. Mais il est aussi possible de lire le fichier .csv dans le programme Stata avec les commandes appropriées.

Commandes préalables

Les codes sont présentés par chapitre du livre avec quelques commentaires. Pour chaque chapitre, ils peuvent être précédés d'un bloc de "commandes préalables" destinées à homogénéiser les figures et à préciser l'emplacement de certains fichiers.

Il s'agit des commandes suivantes :

* Commandes pour enregistrer les résultats dans un fichier.

```
capt log close
cd "/Directory de travail/chapitre x"
log using "code chapx -`c(current_date)'.smcl",replace
.....(lignes de codes de programme)....
log close // à mettre à la fin du programme
```

* le fichier est enregistré dans le *directory de travail* avec le nom "code chapx - date". La date est celle du jour et l'option "replace" permet que ne soit conservé qu'un fichier par jour. Le fichier est au format smcl qui est propre à Stata. Si on en veut une copie en format txt, la commande (à placer à la fin du programme) est :

```
translate "code chapx -`c(current_date)'.smcl" "code chapx -`c(current_date)'.txt",
translator(smcl2txt) replace
```

* Ouverture du fichier de données. L'option clear ferme le fichier actuellement ouvert (s'il y en a un). Attention, les modifications éventuelles de ce fichier ne sont pas enregistrées.

```
use "/Directory/fic.dta", clear
```

* Commande pour éviter que le programme ne s'arrête à chaque page d'écran de résultats et qu'il faille lui dire de continuer

```
set more off
```

* Directory où se trouvent les fonctions. La liste et l'objectif de ces fonctions sont donnés plus loin. Ce sont des fichiers avec l'extension .ado

```
quiet adopath + "Directory" // quiet(ly) évite que le résultat de commande soit affiché.
```

* Options générales pour que les graphiques soient homogènes (elles pourront être modifiées pour des figures particulières si nécessaire)

```
set scheme stcolor // scheme de base qui s'appliquera pour toutes les figures
```

```

* étendue de l'axe des ordonnées
global min=-6
global max=2

* couleur des points observés
global pcol="sienna"

* courbe modélisée
global ccol="forest_green" // couleur
global cl="medthick" // épaisseur du trait

```

Fonctions (.ado files)

De façon générale, on utilise des fonctions pour exécuter des tâches répétitives. Une fonction est un programme Stata particulier avec l'extension .ado (lisible en txt) et des commandes spécifiques au début pour indiquer les arguments (paramètres) nécessaires. Une fonction est appelée avec son nom dans un programme Stata. Il faut placer le fichier dans un directory et indiquer son emplacement au début du programmes avec la commande "adopath +" (voir plus haut).

Dans ce livre, il s'agit de réaliser les courbes, ou de calculer des OR après modélisation d'une variable quantitative comme cela est présenté notamment dans le chapitre 4.

4 fonctions ont été programmées :

Fonction logit_crb.ado

Fonction permettant de construire les courbes du livre, notamment pour le chapitre 4 (Modélisation des variables quantitatives). Elle peut être suivi de la commande *graph export "fig XXX.png"* si on veut enregistrer la figure au format png.

Ce programme est appelé par la commande logit_crb après un modèle de régression logistique avec une seule variable X

La syntaxe est logit.crb varlist [, options]

lin(#) obs(#) crb(#) ic(#)

- varlist doit contenir les 3 variables : Y (en 0/1), X, X' (X' est la variable X en classes pour lesquelles on veut marquer les points observés)

- les options sont (il peut y en avoir aucune, c'est ce qu'indique [] :

min(#) et max(#) : valeurs extrêmes de y=Logit P pour les graphes.

A choisir empiriquement selon les exemples. Les valeurs par défaut sont -6 et 2.

lin(#) : tracé du modèle linéaire entre X et Y

0 pas de tracé (valeur par défaut), 1 tracé du modèle linéaire

obs(#)

0 observations non figurées, 1 observations présentes (par défaut)

crb(#) : courbe

0 pas de courbe ; 1 : avec courbe (par défaut).

Cela peut être utile si on ne veut que les points observés

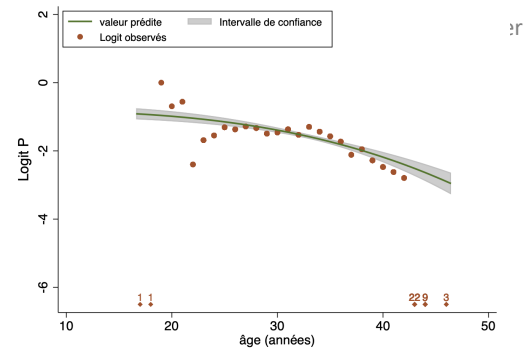
ic(#) : intervalle de confiance

0 pas d'IC ; 1 : présence de l'IC (par défaut)

Utile si on veut "simplifier" la figure.

Exemple (avec les variables du fichier geu)

```
fp <age>, replace dim(1) nocompare : logit acc <age>
logit_crb acc age agea ,min(-6) max(3) lin(0) obs(1)
graph export "fig 17 fp1.png",replace
```



Fonction ORcl_fp.ado

Fonction de calcul des OR par classes et de leur intervalle de confiance après modélisation une commande mfp, fp ou fracpoly

La fonction est appelée par la commande ORcl_pf après un modèle de régression logistique où une variable X est remplacée par un polynôme fractionnaire. Ce modèle doit être créé par une des commandes mfp, fp ou fracpoly.

La syntaxe est : ORcl_pf, ref(#) cl(numlist)

- ne pas oublier la virgule après ORcl_pf

- ref(#) est le centre de la classe de référence et # doit être remplacé par une seule valeur

- cl est la liste des centres de classes, séparés par un blanc ou une virgule, pour lesquels on veut avoir l'OR et son IC (il doit y avoir au moins une classe).

Le programme est limité aux cas où le polynôme fractionnaire de X est de degré ≤ 2 (une ou deux puissances)

Exemple

```
mfp : logistic acc age
ORcl_pf, ref(27) cl(17 22 32 37 42)
```

Résultat

Modélisation : mfp : logistic acc age

variables dans le modèle final : fracpoly: logistic acc age 3 3

OR et IC pour la variable age

ref = 2.7	classe = 17	OR = 0.573	95% CI = [0.380 - 0.865]
ref = 2.7	classe = 22	OR = 0.801	95% CI = [0.657 - 0.976]
ref = 2.7	classe = 32	OR = 0.964	95% CI = [0.846 - 1.098]
ref = 2.7	classe = 37	OR = 0.606	95% CI = [0.509 - 0.721]
ref = 2.7	classe = 42	OR = 0.206	95% CI = [0.145 - 0.291]

Fonction ORcl_sp.ado

Fonction de calcul des OR par classes et de leur intervalle de confiance après une transformation de X par des fonctions splines

Ce programme est appelé par la commande ORcl_sp après un modèle de régression logistique où une variable X est a été remplacée par des fonctions splines cubiques restreints avec 3 à 7 noeuds. Ces fonctions splines doivent avoir été créées par une des commandes makespline ou mkspline.

La syntaxe est : ORcl_sp varname, [knots(numlist max=1)] [knotslist(numlist max=7)] [knplace(string)] ref(real) cl(numlist min=1) [ajust(varlist fv)]

Les arguments entre [] ne sont pas obligatoires

- varname : nom de la variable transformée en splines
- [knots(numlist max=1)] : le nombre de nœuds
ou [knotslist(numlist max=7)] : liste des emplacements des nœuds (séparés par des blancs ou une virgule)
Une et une seule de ces options doit être fournie, celle qui a été donnée dans la commande initiale makespline (ou mkspline).
Il y a une contrainte : il doit y avoir de 3 à 7 nœuds (c'est lié à la possibilité de l'option harrell)
- [knplace(string)] max=7]] : méthode de placements des nœuds
harrell (les nœuds sont placés selon les recommandations de Harrell)
ou uniformknots (les nœuds sont placés en des points équidistants sur l'étendue de X). Par défaut les nœuds sont placés aux percentiles de X.
- ref(#) est le centre de la classe de référence et # doit être remplacé par une seule valeur
- cl est la liste des centres de classes, séparés par des blancs ou des virgules, pour lesquels on veut avoir l'OR et son IC (il peut n'y avoir qu'une seule classe).
- [(ajust)] : liste des variables d'ajustement dans le modèle logistique (éventuellement aucune)

Le programme se limite au cas où une seule variable X est transformée en splines

Exemple

```
makespline rcs age, basis(vsp) knots(3) harrell replace
logit acc `r(regressors)' ovo i.qual
ORcl_sp age, ref(27) cl(17 22 32 37 42) knots(3) knplace(harrell) ajust(ovo i.qual)
```

Résultat

OR et IC pour la variable age ajustée sur les variable ovo i.qual

Avec un modèle logistique où age est remplacé par 2 fonctions splines (dont la 1ère est age) : logit acc (2 variables splines) ovo i.qual

ref = 27	classe = 17	OR = 0.725	95% CI = [0.512 - 1.026]
ref = 27	classe = 22	OR = 0.851	95% CI = [0.716 - 1.013]
ref = 27	classe = 32	OR = 1.078	95% CI = [0.930 - 1.249]
ref = 27	classe = 37	OR = 0.663	95% CI = [0.555 - 0.792]
ref = 27	classe = 42	OR = 0.296	95% CI = [0.219 - 0.401]

Fonction chi2_res.ado

Fonction pour donner des statistiques d'adéquation d'un modèle logistique (chapitre 7)

Il sert notamment à calculer le khi2 de la déviance que Stata ne donne pas.

Les autres résultats : nombre de profils et khi2 de Pearson sont donnés mais la commande estat gof de Stata le fait aussi.

La syntaxe est simplement chi2_res. La fonction s'applique au modèle de régression logistique qui précède

Exemple avec le fichier geu.dta

```
logit ct ctub agea i.tabfc afcs ainf
chi2_res
```

Résultats

Statistiques d'adéquation

nombre de profils = 358

khi2 de Pearson (350) = 331.07 p = 0.12

khi2 de la déviance (350) = 380.67 p = 0.12

Chapitre 1

Pas de code pour ce chapitre

Chapitre 2

II. Estimation d'un pourcentage par la méthode du maximum de vraisemblance

```
clear

set obs 5000

**** Figure 1

** génération de la densité de la vraisemblance
gen x=(_n/_N)
gen y=455*(x^3)*((1-x)^12)

** trait vertical au maximum de la vraisemblance
gen x1=.2 in 1/2
gen y1= 0 in 1
replace y1=.25 in 2

** axes
gen axe1x=0 in 1
replace axe1x=1 in 2
gen axe1y=0 in 1/2
gen axe2x=0 in 1/2
gen axe2y=0 in 1
replace axe2y=.3 in 2

tw (line y x, lpat(solid)) ///
   (line y1 x1, lpat(solid) lw(vthin)), ///
text(0.01 0.98 "P") ///
text(0.3 0 "Vraisemblance", place(e) size(medium) fcolor(black)) ///
yscale(range(0 .3)) ylabel(0 (.05) .3, nogrid angle(horizontal) format(%4,2fc)) ///
xscale(range(0 1)) xtitle("") xlabel(0 (0.1) 1, format(%3,1fc)) ///
legend(off)

* graph export chap2-fig1.png,replace
```

III. Application au modèle logistique

```
use "geu.dta", clear
```

```
**** Tableaux 1 à 3
```

```
tab ct salp
logit ct salp
logistic ct salp
```

IV. Tests des paramètres du modèle logistique

```
**** Tableau 4 : Test du rapport des vraisemblances maximum
logistic ct salp
est store a
logistic ct if salp!=.
/* !=. veut dire différent de donnée manquante et permet d'estimer le modèle b
sur les mêmes sujets que le modèle a */
est store b
lrtest a b
```

```
**** Tableau 5 : Test du khi2
```

```
tab ct salp,chi
```

```
**** Test simultané de plusieurs paramètres
```

```
*** Tableau 6
```

La régression logistique en épidémiologie – Codes Stata – Chapitre 2


```
logistic ct salp univf fprof
```

```
**** Test de Wald (Tableau 7, partie 1)
testparm univf fprof
```

```
**** Test du rapport des vraisemblances (Tableau 7, partie 2)
est store a // cette commande met en mémoire le modèle précédent avec les 3 variables salp,
univf,fprof
logistic ct salp if univf!=. & fprof!=.
/* if univf!=. & fprof!=. peut être remplacé par if e(sample) qui veut dire
qu'on limite l'analyse aux sujets utilisés pour le modèle de régression
précédent */
est store b // cette commande met en mémoire le modèle précédent avec la seule variable salp
lrtest a b
```

VI. Annexe 2 : Les 3 tests issus de la méthode du maximum de vraisemblance

```
clear
set obs 5000

* valeur du tableau à 4 cases ct*salp
local e1m1=91
local e0m1=466
local e1m0=26
local e0m0=1122
local alpha=-0.9
* constante
local lnc=lnfactorial(1705)-lnfactorial(557)-lnfactorial(1148)
* valeurs pour l'axe des abscisses (noté beta à la place de x)
gen beta=((_n/37)-(_N/100))/10+3

* ln vraisemblance en fonction de beta
gen lnV=-`e1m0'*(`alpha'+beta)-`e0m0'*`alpha'-`e1m1'+`e1m0'*ln(1+1/exp(`alpha'+beta))-
`e0m1'+`e0m0'*ln(1+1/exp(`alpha'))
* vraisemblance en fonction de beta
gen V=exp(lnV+`lnc')

* valeur testée (beta = 0)
local b0=0
local maxV=-1023.05 // max de la vraisemblance
local beta_est=2.131 // valeur de l'estimation de beta

* forme de la vraisemblance avec la tangente au maximum
* coordonnées pour les barres avec flèches
gen ysc5=`maxV'
gen xsc5=`beta_est'-2
gen xsc6=`beta_est'+2

graph tw (line lnV beta) ///
(pcbarrow ysc5 xsc5 ysc5 xsc6, lw(vthin) col(black)) ///
, /// xlabel(none) ylabel(none)
ytitle (ln vraisemblance) ///
xline(`beta_est') ///
text(-1275 2.131 "{&beta}") ///
text(-1275 2.131 "^", size(`saxe') place(n)) ///
legend(off) ///
scheme(s2mono_livre)
* graph export chap2-vraisemblance.png,replace

* test de Wald
* coordonnées pour les barres avec flèches
gen yw1= -1100
gen xw1=`beta_est'
gen xw2=`b0'

graph tw (line lnV beta) ///
```

```

        (pcbarrow yw1 xw1 yw1 xw2, lw(*2.5) col(black)) ///
, /// xlabel(none) ylabel(none)
ytitle (ln vraisemblance) ///
xline(`beta_est') ///
xline(`b0') ///
text(-1275 2.131 "{&beta;}") ///
text(-1275 2.131 "^", size(`saxe') place(n)) ///
legend(off)
* graph export chap2-Wald.png,replace

* test du rapport de vraisemblance
local ylr=lnV[740] // n_=740 => beta=0 (obtenu dans le fichier par list lnV if beta==0)
* coordonnées pour les barres avec flèches
gen ylr1=`ylr'
gen ylr2=`maxV'
gen xlr1=0

graph tw (line lnV beta) ///
        (pcbarrow ylr1 xlr1 ylr2 xlr1, lw(*2.5) col(black)) ///
, /// xlabel(none) ylabel(none)
ytitle (ln vraisemblance) ///
yline(`maxV') ///
yline(`ylr') ///
xline(`beta_est') ///
xline(`b0') ///
text(-1275 2.131 "{&beta;}") ///
text(-1275 2.131 "^", size(`saxe') place(n)) ///
legend(off)
* graph export chap2-LR.png,replace

* test du score
local p=55 // pente de la tangente pour beta=0 (obtenue par essais successifs)
local l=0.7 // longueur de la tangente

* coordonnées pour la tangente pour beta=0
gen xsc1=`b0'-`l'
gen xsc2=`b0'+`l'
gen ysc1=`p'*(xsc1-`b0')+`ylr'
gen ysc2=`p'*(xsc2-`b0')+`ylr'

* coordonnées pour la tangente au maximum
gen ysc3=`maxV'
gen xsc3=`beta_est'-`l'-1
gen xsc4=`beta_est'+`l'+1

graph tw (line lnV beta) ///
        (pcbarrow ysc1 xsc1 ysc2 xsc2, lw(*2.5) col(black)) ///
        (pcbarrow ysc3 xsc3 ysc3 xsc4, lw(*2.5) col(black)) ///
, /// xlabel(none) ylabel(none)
ytitle (ln vraisemblance) ///
xline(`beta_est') ///
xline(`b0') ///
text(-1275 2.131 "{&beta;}") ///
text(-1275 2.131 "^", size(`saxe') place(n)) ///
legend(off)
* graph export chap2-score.png,replace

```

Chapitre 3

```
use "geu.dta", clear
```

I. Règle générale d'interprétation du coefficient d'une variable (tableau 1)

```
logit ct age
```

```
***** § II. Variable dichotomique (tableau 2)
```

```
gen salp0_2=2*salp
logit ct salp
logit ct salp0_2
logistic ct salp0_2
```

III. Variable qualitative nominale à plus de 2 classes

```
**** Tableau 4
```

```
logistic ct i.gind // aide sur la syntaxe i. : help fvvarlist (factor variables)
```

```
**** Tableau 5
```

```
tab ct gind
```

```
**** Tableaux 6 et 7 : changement de catégorie de référence
```

```
logistic ct ib(2).gind
logistic ct ib(3).gind
```

```
***** Tableau 8
```

```
quiet logistic ct i.gind
lincom 3.gind-2.gind, or
```

* nb : "quiet" permet de ne pas afficher le résultat de la commande qui a été donné avant, mais qui est nécessaire pour la ligne suivante

```
**** Tableau 9
```

```
quiet logistic ct i.gind
testparm i.gind
```

```
**** Tableau 10
```

```
quiet logistic ct i.gind
est store a
quiet logistic ct if gind!=.
est store b
lrtest a b
```

IV. Variable qualitative ordinale

```
**** Tableau 12
```

```
tab ct tabfc
```

IV.1. Modélisation de l'association GEU – tabac

```
**** Figure 1
```

** nb : cette figure peut aussi être obtenue par `logit_crb ct tabfc tabfc,obs(1) min(-1.5) max(1) ic(0)`, même si c'est un peu un détournement de la fonction `logit_crb`

```
qui logit ct i.tabfc
predict xp1, xb
qui logit ct tabfc
predict lp1,xb
```

```
local saxe "vlarge"
```

```
tw (line lp1 tabfc) ///
(scatter xp1 tabfc, msymbol(0) mcolor(bk)), ///
text(1 0 "Logit P", size(`saxe') place(e)) ///
text(-1.45 3 "Tabac", place(w) size(`saxe')) ///
```

```

yscale(range(-1.5 1)) ylabel(-1.5 (.5) 1, nogrid angle(horizontal) format(%4,1fc) labsize(`saxe`))
///
xlabel(labsize(`saxe`)) ///
xtitle("") ytitle("") ///
legend(off) ///
ysize(4) xsize(10) ///
scheme(s2mono_livre)
* graph export ct_tab_lineaire.png,replace

```

```

**** Tableau 13
logistic ct i.tabfc
logistic ct tabfc
lincom 2*tabfc,or
lincom 3*tabfc,or

```

IV.2. Choix des valeurs de X

```

recode tabfc 1=5 2=15 3=25, gen(tabfc2)

```

```

**** Tableau 14
logistic ct tabfc2
lincom 5*tabfc2,or
lincom 15*tabfc2,or
lincom 25*tabfc2,or

```

```

predict lp2,xb
qui logistic ct i.tabfc2
predict xp2, xb

```

```

**** Figure 2
tw (line lp2 tabfc2) ///
    (scatter xp2 tabfc2, msymbol(0) mcolor(bk)), ///
text(1 0 "Logit P", size(`saxe`) place(e)) ///
text(-1.45 25 "Tabac (centre des classes)", place(w) size(`saxe`)) ///
yscale(range(-1.5 1)) ylabel(-1.5 (.5) 1, nogrid angle(horizontal) format(%4,1fc) labsize(`saxe`))
///
xscale(range(0 25)) xlabel(0(5)25, labsize(`saxe`)) ///
xtitle("") ytitle("") ///
legend(off) ///
ysize(4) xsize(10) ///
scheme(s2mono_livre)
* graph export ct_tab_codage2.png,replace

```

IV.3. Choix entre variables indicatrices et modélisation linéaire (test de linéarité)

* Programme pour tracer les 3 courbes descriptives (linéaires et en escalier). La programmation est compliquée. Et c'est juste pour faire un joli dessin ...

```

qui logit ct tabfc
predict lin1,xb
qui logit ct i.tabfc
predict yesc,xb

```

```

* yesc2 est là ainsi que tcont2 pour une meilleure allure des courbes dans le premier intervalle
(tabfc=0)
clonevar yesc2=yesc
replace yesc2=yesc +(yesc-lin1)*30 if tabfc==0

```

```

* génération d'intervalles autour des valeurs de tabfc
* tcont intervalles pour courbes en escalier
gen tcont=uniform()/2
replace tcont =uniform()+tabfc-0.5 if tabfc!=0

```

```

* Figure 3 avec la droite et les 2 fonctions en escalier ("linéaire" et indicatrices)
local saxe "medium"
tw (scatter yesc2 tabfc, sort msymbol(0)) ///
    (line lin1 tcont,sort c(stairstep) lwidth(medthick) lpattern(solid) lcolor(red)) ///
    (line yesc2 tcont,sort c(stairstep)lwidth(medthick) lpattern(dash) lcolor(blue)) ///
    (line lin1 tabfc,sort lwidth(medthick) lpattern(shortdash) lcolor(red)) , ///
text(1 0 "Logit P", size(`saxe`) place(e)) ///
ylabel(-1 (.5) .5, nogrid angle(horizontal) format(%4,1fc) labsize(`saxe`)) ///

```

```

yttitle("") ///
xttitle(,size(`saxe`)) xlabel(0(1)3) ///
legend(order(1 2 3 4) label(1 "Logit observés") label(2 "Modélisation linéaire") label(3 "Variables
indicatrices") label(4 "Droite") ///
pos(4) ring(0) size(small) symxsize(*.5) symysize(*.5)) xttitle("Tabac en classes") ///
ysize(5) xsize(6) ///
scheme(s2mono_livre)

```

```

* graph export ordinal_3crb.png,replace

```

```

**** Test de linéarité (tableau 15)

```

```

qui logit ct tabfc
est store lin
qui logit ct i.tabfc
est store indic
lrtest indic lin

```

```

test (2.tabfc=2*1.tabfc) (3.tabfc=3*1.tabfc)

```

VI. Prise en compte d'une interaction

```

**** Tableau 16

```

```

logit ct i.clomid##i.age30
testparm 1.clomid#1.age30

```

```

**** Tableau 17

```

```

qui logistic ct i.clomid##i.age30
lincom 1.clomid + 1.clomid#1.age30

```

```

**** Tableau 18 : interaction et facteur à plus de 2 classes

```

```

logit ct i.clomid##i.agec3
lincom 1.clomid,or
lincom 1.clomid + 1.clomid#2.agec3,or
lincom 1.clomid + 1.clomid#3.agec3,or
testparm (1.clomid#2.agec3) (1.clomid#3.agec3) // ou testparm 1.c*#1.a* 1.c*#2.a*

```

```

**** Tableau 19 : analyses séparées

```

```

logit ct clomid if age30==0
logit ct clomid if age30==1

```

Chapitre 4

```
use "cycles3.dta",clear
```

IV. Courbes lowess avec 2 valeurs de bandwidth

```
** variables nécessaires ici car les courbes lowess ne peuvent pas être obtenues avec logit.crb
bys agea: egen pacc=mean(acc)
bys agea: egen ycount=count(acc)
gen yobs=ln(pacc/(1-pacc))
gen yobs1=$min-0.5 if pacc==0
gen yobs2=$max-0.5 if pacc==1
label var yobs "Logit observés"

* valeurs min et max particulières pour les courbes lowess
tempvar yobs1b
local min=-8
gen `yobs1b'=`min'-0.5 if pacc==0

foreach bd of numlist .8 .1 { // boucle pour les 2 valeurs de bandwidth
    local bd2=subinstr("`bd'", ".", "0", 1) // pour nommer le graphe
    lowess acc age, ///
    logit bw(`bd') lineopts(lcolor($ccol) lw($cl)) ///
    addplot(scatter yobs `yobs1b' yobs2 agea, ///
    msymbol(0 D D) mcolor($pcol $pcol $pcol) ///
    sort msize(1 .5 .5) mlabel(. ycount ycount) mlabpos(12 12 12)) ///
    ylabel(-8 (2) $max,nogrid) title("") subtitle("") ///
    xtitle("âge (années)") ytitle("Logit P") ///
    legend(order(2 3) label(2 "Courbe lowess") ///
    label(3 "Logit observés") pos(1) ring(0) size(small) symxsize(*.5) symysize(*.5)) $gcol

    * graph export "fig 8 lowless`bd2'.png",replace
}
```

V. Graphe des logits observés pour la régression accouchement en fonction de l'âge

```
quiet logit acc age // "quiet" (ou en abrégé qui) évite que le résultat s'affiche
logit_crb acc age agea, min($min) max($max) obs(1) lin(0) ic(0) crb(0)

* graph export "fig 9 logits_obs.png",replace
```

VI. Modélisation avec une fonction en escalier

```
**** Modèle 1 : linéaire (Figure 11)

qui logit acc age
est store M1 // pour garder les résultats en mémoire en vue de comparaisons ultérieures
logit_crb acc age agea,min($min) max($max) lin(0) obs(1)
* graph export "fig 11 lineaire.png",replace

**** Histogrammes sans et avec regroupements (Tableau 1)
logit acc i.agea
tab agea acc,miss
recode agea (min/19=18) (42/max=44), gen(age2) // regroupement des catégories extrêmes
tab age2 acc, miss

**** Coefficients des modèles A et B
logit acc agea
est store A
logit acc ib30.age2, cformat(%9.3f)
La régression logistique en épidémiologie – Codes Stata – Chapitre 4
```

est store B

**** Les 3 courbes descriptives (linéaires et en escalier) Figures 12 et 13
 * changements d'échelle des Logit pour ces 2 figures

local min=-3.5

local max=.5

gen yobs1b=`min'-0.5 if pacc==0

gen yobs2b=`max'-0.5 if pacc==1

* Logits prédits

qui logit acc age

predict lin1,xb

qui logit acc i.age2

predict yesc2,xb

graph twoway (scatter yobs yobs1b yobs2b agea, sort msymbol(0 D D) ///

msize(1 .5 .5) mlabel(. ycount ycount) mlabpos(12 12 12) ///

mcol(\$pcol \$pcol \$pcol)) ///

(line lin1 age,lwidth(medium) lcolor(black) lpattern(solid)) ///

(line lin1 agea,c(stepstair) lwidth(medthick) lcolor(cyan) lpattern(solid)) ///

(line yesc2 age2,c(stepstair) lw(\$cl) lcol(\$ccol) lpattern(dash)) ///

, ylabel(`min' (2) `max',nograd) ytitle("Logit P") ///

legend(order(1 4 5 6) label(1 "Logit observés") label(4 "Modèle A") label(5 "Modèle 1") label(6 "Modèle B") pos(1) ring(0) size(small) symxsize(*.5) symysize(*.5)) xtitle("âge en année")

* graph export "fig 12 modeles_1_A_B.png",replace

**** Comparaison des modèles 1 et A et 1 et B (pour voir les résultats obtenus quand les modèles ne sont pas emboîtés)

* capture = pas d'arrêt du programme même s'il y a une erreur

* noisily = afficher quand même le résultat (ici, l'erreur)

capture noisily lrtest M1 A

lrtest M1 B

**** Comparaison modèles A et B

lrtest A B

**** Courbes en escalier avec age en classes de 1 an et de 5 ans

** Tableau 2

logistic acc ib30.age2, cformat(%9.2f)

recode agea (min/19=1) (20/24=2) (25/29=3) (30/34=4) (35/39=5) (40/max=6), gen(age3)

label var age3 "age cl 5 ans"

* tab agea age3 //si besoin pour vérifier l'exactitude de la commande précédente

logistic acc ib3.age3, cformat(%9.2f)

qui logit acc i.age2

predict ypred,xb

qui logit acc i.age3

est store C

predict yesc3,xb

graph twoway (scatter yobs yobs1b yobs2b agea, sort msymbol(0 D D) ///

msize(1 .5 .5) mlabel(. ycount ycount) mlabpos(12 12 12) ///

mcol(\$pcol \$pcol \$pcol)) ///

(line yesc2 age,c(stepstair) lpattern(dash) lcolor(\$ccol) lw(\$cl)) ///

(line yesc3 age2,c(stepstair) lcolor(red) lwidth(medthick)), ///

ylabel(`min' (2) `max',nograd) ytitle("Logit P") ///

legend(order(1 4 5) label(1 "Logit observés") label(4 "Modèle B") label(5 "Modèle C (classes d'âge de 5 ans)") pos(1) ring(0) size(small) symxsize(*.5) symysize(*.5)) xtitle("Age (années)")

* graph export "fig 13 age1-5ans.png",replace

La régression logistique en épidémiologie – Codes Stata – Chapitre 4

```
*** comparaison des modèles B et C
lrtest B C
```

VII. Modélisation avec des polynômes

```
foreach p of numlist 3 5 7 { // boucle pour des polynômes de degré 3, 5 et 7
    capt drop age_* // pour éviter une erreur au 2ème tour de la boucle
    local poly`p'="age"
    foreach k of numlist 2/`p' { // boucle pour générer les variables puissance de age
        gen age_`k' = age^`k'
        local poly`p'="poly`p'" + " " + "age_`k'"
    }

    logit acc `poly`p''
    testparm age_*

    logit_crb acc age agea,min(-7) max(3) lin(0) obs(1)

    * graph export "figure 14 poly`p'.png",replace
}
```

VIII. Modélisation avec des polynômes fractionnaires

```
use "/Volumes/Macintosh_HD/Enseignement/Livres/Livre Reg log/Fichiers de donnees livre/Fichiers de
donnees publics/cycles3/cycles3.dta",clear
```

VIII.2 Choix des puissances d'un polynôme fractionnaire de degré donnéFP

```
*** Figure 17
fp <age>, replace dim(1) nocompare : logit acc <age>
notes // commande quasi indispensable pour interpréter le résultat de fp
logit_crb acc age agea ,min($min) max($max) lin(0) obs(1)
* graph export "fig 17 fp1.png",replace

*** Figure 18
fp <age>, replace dim(2) nocompare : logit acc <age>
notes
qui predict pred_fp2,xb // inutile ici, servira pour la figure 21
label var pred_fp2 "FP{sub:2}"
logit_crb acc age agea ,min($min) max($max) lin(0) obs(1)
* graph export "fig 18 fp2.png",replace

**** Tableau 4 : Les deviances de tous les FP1 et FP2
fracpoly logit acc agea, adjust(no) log

**** Figure 19 : 3 couples de puissances
capt drop age_*
foreach pow in "3 3" "3 1" "3 2" {
    fp generate age^(`pow'), scale(0 10) replace
    logit acc age_*
    logit_crb acc age agea,min(-6) max(2) lin(0) obs(0)
    * graph export "fig 19 fp (`pow').png",replace
}

**** Figure 20 : FP 4
fp <age>, replace dim(4) nocompare : logit acc <age>, iter(15)
* iter(15) pour que fp puisse converger avec degré 4 (option nécessaire depuis Stata 11 et le chgt
dans la procédure ml)
notes
qui predict pred_fp4,xb // inutile ici, servira pour la figure 21
label var pred_fp4 "FP{sub:4}"
logit_crb acc age agea ,min($min) max($max) lin(0) obs(1)
* graph export "fig 20 fp4.png",replace
```

```
**** Figure 21 : les deux graphes fp2 et fp4 ensembles
```



```

* cela nécessite des calculs préalables car cette figure ne peut pas être obtenue avec la fonction
logit_crb
* Il faut en particulier les lignes de code pour avoir points observés
bys agea: egen pacc=mean(acc)
bys agea: egen ycount=count(acc)
gen yobs=ln(pacc/(1-pacc))
gen yobs1=$min-0.5 if pacc==0
gen yobs2=$max-0.5 if pacc==1
label var yobs "Logit observés"

twoway (scatter yobs yobs1 yobs2 agea, sort msymbol(0 D D) ///
msize(1 .5 .5) mlabel(. ycount ycount) mlabpos(12 12 12) ///
mcol($pcol $pcol $pcol)) ///
(line pred_fp2 pred_fp4 age, sort lcol($cc dkorange) ///
lpat(solid dash)) ///
, ylabel($min (2) $max,nogrid) xlab(15 25 to 45) l1title("Logit P") xtitle("âge (années)") legend
(order(4 5 1) pos(11) ring(0) size(small) symxsize(*.5) symysize(*.5) col(1))

* graph export "fig 21 fp 2_4.png",replace

```

VIII.3. Choix du meilleur polynôme fractionnaire pour une variable : Tableau 5

```
mfp, select(.05) center(no) : logit acc age
```

**** VIII.4. Modélisation simultanée de plusieurs variables quantitatives, procédure mfp

```

* Figure 22 : odélisation de ovo et emb
mfp, select(.05) center(no) : logit acc ovo
recode ovo (31/35=33) (36/40=38) (41/max=50),gen(ovoc)
logit_crb acc ovo ovoc,min(-6) max(2) lin(0) obs(1)
* graph export "fig 22 ovo.png",replace

```

```

mfp, select(.05) center(no) : logit acc emb
recode emb (16/max=19),gen(embc)
logit_crb acc emb embc,min(-6) max(2) lin(0) obs(1)
* graph export "fig 22 emb.png",replace

```

* Tableau 6 : procédure mfp avec plusieurs variables

```
mfp, center(no) select(0.05) : logit acc age ovo emb
```

```

**** Exemple avec une variable qualitative fictive (qual)
xi : mfp, center(no) select(0.05) : logit acc age ovo emb (i.qual)

```

IX. Modélisation avec des fonctions splines

IX.4 Splines linéaires (Figures 25 et 26)

** Option marginal avec mkspline, équivalente à ne pas mettre local avec makespline mais il faut mettre order(1)

```

foreach nk of numlist 1 4 { // nk nombre de noeuds
    makespline linear age, order(1) basis(vsp) knots(`nk') replace
    mat li r(knots) // pour avoir l'emplacement des noeuds

    foreach k of numlist 1/`nk' {
        label var vsp_1_`k' "(X - `=round(A[1,`k'],.1)')+ " // cela permet d'avoir les bonnes
légendes dans les figures
    }

    * graphe des fonctions splines de base
    twoway (line age vsp_1* age, sort), /// // vsp_1* désigne toutes les variables dont le nom
commence par vsp_1 (elles sont créées par makespline)
    xtitle("age") ytitle("") ///
    legend(pos(11) ring(0) label(1 "X = age")) $gcol
    * graph export "fig 25-26 splines_bases `nk'.png",replace

    * courbes splines
    logit acc age vsp_1*
    logit_crb acc age agea,min(-6) max(2) lin(0) obs(1)

```

```

    * graph export "fig 25-26 lspline_`nk'.png",replace
}

```

IX.4.c Stepwise pour réduire le nombre de noeuds (Tableau 7)

```

makespline linear age, order(1) basis(vsp) knots(11) replace
matrix l r(knots), format(%6.1f)
sw, pr(.10) lockterm1 : logit acc age vsp_1*

```

IX.5. Splines cubiques (Figure 27)

```

makespline piecewise age, order(3) basis(vsp) knots(3) replace
disp "`r(regressors)'"
matrix l r(knots)

gen age2=age^2
gen age3=age^3

logit acc age age2 age3 vsp_1*

/* on obtient la même chose que la ligne précédente avec
logit acc `r(regressors)'"
En effet, `r(regressors)'" est une macro qui contient c.(c._rs_sp_1##c._rs_sp_1##c._rs_sp_1 vsp_1_1
vsp_1_2 vsp_1_3)
- le c. qui commence la ligne veut dire que toutes les variables entre parenthèses doivent être
considérées comme continues
- les vvariable vsp_1_i sont les fcontions splines créées. Elles peuvent être représentées par
vsp_1* où * désigne n'importe quelle chaîne de caractères
- la variable age est renommée par défaut _rs_sp_1 (et éventuellement changée d'échelle)
- l'ensemble c._rs_sp_1##c._rs_sp_1##c._rs_sp_1 est une utilisation astucieuse de l'opérateur # qui
équivalent à l'ensemble age, age^2 et age^3 (voir help fvvarlist)
*/

logit_crb acc age agea,min(-12) max(3) lin(0) obs(1)
* graph export cspline_fig27.png,replace

```

IX.6 Splines cubiques restreints (Figure 28)

```

makespline rcs age, order(3) basis(vrsp) knots(3) norescalevars replace
disp "`r(regressors)'"
matrix l r(knots)

tway (line age vrsp* age, sort), xtitle("age") ytitle("") ///
legend(pos(11) ring(0) label (1 "X") label(2 {stSymbol:u}{sub:2}))
* graph export "fig 28a fonc_spline_rcs.png",replace

logit acc age vrsp*
/* on obtient la même chose avec
logit acc `r(regressors)'"
à noter que cette fois `r(regressors)'" contientc.(c.age vrsp_1_1), et donc que la variable age n'est
pas renommée
*/

logit_crb acc age agea,min(-6) max(3) lin(0) obs(1)
* graph export rcspline_fig28b.png,replace

```

IX.7. Choix du modèle avec des fonctions splines (tableaux 8 et 9)

```

uvrs logit acc age, alpha(0.05) degree(3) df(4) trace

mvrs logit acc age ovo emb, alpha(0.05) degree(3) df(4) select(0.05)

```

X. Présentation des résultats issus de la modélisation

X.3. Présentation quantitative des résultats dans un tableau (tableau 11)

```

** construction d'une variable âge en classes de 5 ans
gen age5=agea
recode age5 min/19 = 17 20/24 = 22 25/29 = 27 30/34 = 32 35/39 = 37 40/max= 42
label var age5 "âge classes 5 ans"
*char age5[omit]27 // pour que la catégorie de référence soit 25-29 ans

```

```

** variables indicatrices
logistic acc b(27).age5, cformat(%5.2f)

** polynômes fractionnaires
mfp : logistic acc age
ORcl_pf, ref(27) cl(17 22 32 37 42)

** fonctions splines
makespline rcs age, basis(vsp) knots(3) harrell replace
logit acc `r(regressors)'
ORcl_sp age, ref(27) cl(17 22 32 37 42) knots(3) knplace(harrell)

```

XII.2 Représentation graphique des données observées avec la courbe modélisée (Figure 31)

```

mfp, df(2) : logit acc age
fracplot, lineopt(c(1)) ciopts(color(gs12)) title("Age non arrondi, PF1") ylabel(-6(2)2,nogrid) ///
addplot(,legend(pos(11) ring(0) order(3 1 - "" 2) ///
label(2 "Valeurs {c 34}observées{c 34} données par fracplot") ///
label(3 "Logit P") label(1 "Intervalle de confiance"))) $gcol
graph export "fig 31 PF1 non arrondi.png",replace

mfp, df(4) : logit acc age
fracplot, lineopt(c(1)) ciopts(color(gs12)) title("Age non arrondi, PF2") ylabel(-6(2)2,nogrid) ///
addplot(,legend(pos(11) ring(0) order(3 1 - "" 2) ///
label(2 "Valeurs {c 34}observées{c 34} données par fracplot") ///
label(3 "Logit P") label(1 "Intervalle de confiance"))) $gcol
graph export "fig 31 PF2 non arrondi.png",replace

```

Chapitre 5

```
use "geu.dta", clear
```

IV.3 Colinéarité

```
tab ctub ageu  
logistic ct ctub  
logistic ct ageu  
logistic ct ctub ageu
```

V.3. Sélection fondée sur le changement de l'estimation de l'odds-ratio

```
logit ct age30 tabf univf afcs aivg ainf clomid ptub  
chest age30,lockterms(tabf) backward format(%5.3g)  
* graph export chest.png,replace
```

V.4. Méthodes de sélection pas à pas (stepwise)

```
stepwise, lockterm1 pr(.1) : logit ct (age30 tabf) univf afcs aivg ainf clomid ptub  
stepwise, lockterm1 pr(.008) : logit ct (age30 tabf) univf afcs aivg ainf clomid ptub  
stepwise,pr(.1) : logit ct (i.agec) tabf univf afcs aivg ainf clomid ptub
```

Chapitre 6

II. Régression logistique multinomiale

```
use "premafiv.dta", clear
```

II.2. Exemple et interprétation des résultats

```
**** Tableau 1
```

```
mlogit cprema age35
```

```
**** Tableau 2 (les lignes avec * sont les autres régressions logistiques binomiales qui ne sont pas sur le tableau)
```

```
logit rupmemb age35
```

```
*logit spont age35
```

```
*logit prov age35
```

```
**** Tableau 3
```

```
mlogit cprema age35, rrr
```

II.3. Régression logistique multinomiale versus plusieurs régressions binomiales

```
**** Tableau 4 (même remarque que pour le tableau 2)
```

```
mlogit cprema age35 hta
```

```
logit rupmemb age35 hta
```

```
*logit spont age35 hta
```

```
*logit prov age35 hta
```

```
**** Tableau 5 (même remarque que pour le tableau 2)
```

```
mlogit cprema i.age35##i.hta
```

```
logit rupmemb i.age35##i.hta
```

```
*logit spont i.age35##i.hta
```

```
*logit prov i.age35##i.hta
```

II.4. Comparaison des OR associés à X selon les catégories de Y

```
**** Tableau 6
```

```
mlogit cprema age35 hta, rrr
```

```
test [2]hta = [3]hta
```

```
test [4]hta = [3]hta
```

```
test ([2]hta = [3]hta) ([4]hta = [3]hta)
```

```
test [2]age35 = [3]age35
```

```
test [3]age35 = [4]age35
```

```
test ([2]age35 = [3]age35) ([3]age35 = [4]age35)
```

```
/***** autre syntaxe pour la tableau 6
```

```
test [1=2] : age35
```

```
test [1=3] : age35
```

```
test ([1=2] : age35) ([1=3] : age35)
```

```
test [1=2] : hta
```

```
test [1=3] : hta
```

```
test ([1=2] : hta) ([1=3] : hta)
```

```
****/
```

II.5. Changement de classe de référence pour Y

```
**** Tableau 7
```

```
mlogit cprema age35, b(3)
```

IV. Modèle cumulative-odds

```
use "septaviv.dta", clear
```

IV.2. Exemple : Fragilité chez les personnes vivant avec le VIH

**** Tableau 8
tab1 sf5 sf3,m

IV.3. Résultats avec l'hypothèse des odds proportionnels

**** Tableau 9
ologit sf5 precaire, nolog

IV.4. Regroupement de classes de Y et test de l'hypothèse des odds proportionnels

**** Tableau 10
ologit sf3 precaire, nolog

**** Tableau 11
gologit2 sf5 precaire
test ([1=2] : precaire) ([1=3] : precaire) ([1=4] : precaire)

IV.5. Présentation des résultats

**** Tableau 12
qui ologit sf3 precaire, nolog
predict robuste pre_fragile fragile
tabstat robuste pre_fragile fragile, stat(mean) by(precaire) format(%4.2f) nototal
drop robuste pre_fragile fragile

**** Figure 1
* mfp:ologit sf3 epices
ologit sf3 epices
predict p*
label var p1 "robuste"
label var p2 "pré-fragile"
label var p3 "fragile"
tw (line p? epices, sort), ///
legend(pos(0) col(1)) ylabel(,angle(0) format(%4.2f))
* graph export ologit.png,replace

IV.6. Plusieurs variables indépendantes

**** Tableau 13
local dep "precaire cd4_350 mrc parten"
ologit sf5 `dep'
gologit2 sf5 `dep'
foreach v of varlist `dep' {
 test ([1=2] : `v') ([1=3] : `v') ([1=4] : `v')
}
}

**** Tableau 14
gologit2 sf5 precaire cd4_350 mrc parten, npl(mrc)

V. Modèle continuation-ratio

use "/Volumes/Macintosh_HD/Enseignement/Livres/Livre Reg log/Fichiers de donnees livre/Fichiers de données publics/suc_fiv/suc_fiv.dta",clear

**** V.1. Exemple : rang de succès en FIV
**** Tableau 15
tab rgsuc,m

**** Tableau 16
gencrm rgsuc age35,or nolog
est store mprop

V.2. Test de l'hypothèse des odds proportionnels

**** Tableau 17
gencrm rgsuc age35,or nolog free(age35)
est store mfree

**** Tableau 18

```
lrtest mprop mfree
test _b[eq1:age35] = _b[eq2:age35] = _b[eq3:age35] = _b[eq4:age35]
```

V.3. Modélisation de la durée d'infécondité

```
**** Tableau 19
gencrm rgsuc dinf5,or nolog
```

```
**** Tableaux 20 et 21
qui gencrm rgsuc i.dinf,or nolog
est store ind
qui gencrm rgsuc dinf,or nolog
est store lin
lrtest lin ind
gencrm rgsuc dinf,or nolog
```

VI. Modèle adjacent-category

```
use "ado.dta"
```

```
**** Tableau 22
adjcatlogit satis genre
** équivalence avec un modèle multinomial avec contraintes
constraint 1 [3]genre = 2*[2]genre
constraint 2 [4]genre = 3*[2]genre
constraint 3 [5]genre = 4*[2]genre
mlogit satis genre, b(1) constraint(1 2 3)
est store madj
```

```
**** Tableau 23 (modèle multinomiale sans contrainte)
mlogit satis genre, b(1)
est store mlog
test ([3]genre = 2*[2]genre) ([4]genre = 3*[2]genre) ([5]genre = 4*[2]genre)
lrtest mlog madj
```

```
**** Tableau 24 et 25
adjcatlogit satis MC
mlogit satis MC, b(1)

* Test de l'hypothèse des risques proportionnels
qui mlogit satis MC, b(1)
est store mlog
constraint 4 [3]MC = 2*[2]MC
constraint 5 [4]MC = 3*[2]MC
constraint 6 [5]MC = 4*[2]MC
mlogit satis MC, b(1) constraint(4 5 6)
est store madj
lrtest mlog madj
```

VII. Choix du modèle

```
use "suc_fiv.dta",clear
```

VII.2. Un peu d'humilité sur l'importance du choix ...

```
**** Tableau 26
gencrm rgsuc age35 dinf5, nolog

**** Tableau 27
ologit rgsuc age35 dinf5
```

Chapitre 7

```
use "geu.dta",clear
```

II. Tests d'adéquation

III.3. Exemple

```
**** Tableau 2 et 3 : Coefficients du modèle logistique et distribution des profils (covariate
patterns)
logit ct ctub agea i.tabfc afcs ainf
estat gof
deviance

predict pat,number
bys pat : gen npat=_N if _n==1 & pat!=.
tab npat

**** Test de Hosmer et Lemeshow (tableau 4)
estat gof,group(10) table
```

IV. Courbes ROC

IV.1. Régression logistique et classement

```
**** Tableaux 5 et 6
estat classification
estat classification, cutoff(.3)

**** Figure 1
lsens, msymbol(i i) lpattern(dash solid) lcolor(blue red)
*graph export lsens.png,replace
```

IV.2. Aire sous la courbe ROC

```
**** Figure 2
lroc, msymbol(i) rlopts(clpat(dash)) lcolor(blue) lw(thick)
*graph export lroc.png,replace
```

V. Diagnostics de régression

```
**** Création des variables de diagnostics de régression
predict p
format p %5.2f
predict r,residuals
predict d,deviance
predict dd,ddeviance
predict h,hat
predict dx2,dx2
predict db,dbeta
format db dx2 dd %4.2f
```

V.2. Influence d'un profil sur les statistiques d'adéquation

```
* calculs préalables pour avoir la même échelle pour tous les figures 3 et 4
qui sum dx2
local mdx2=round(`r(max)',1)
qui sum dd
local mdd=round(`r(max)',1)
local maxy=max(`mdx2',`mdd')

**** Figure 4 : Delta chi2
```



```

twoway scatter dx2 p,msymbol(Oh) mcolor(black) msize(vsmall) ylab(0 (2) `maxy',angle(hori) nogrid)
xlab(0 (.1) 1) yline(4,lcolor(red) lpat(dash)) xtitle("P{sub:j}") ytitle("") text(`maxy' 0
"{{Delta}}{{chi}}{sub:j}{sup:2}",place(e))
*graph export dx2_P.png,replace

```

```

**** figure 4b graphe avec les "courbes noires"
local c=0.2 // valeur empirique pour que les courbes collent bien
local pmax=.91 // pour limiter l'échelle des y
gen crb_d=p/((1-p)*(1-`c'*p*(1-p))) if p<`pmax'
gen crb_g=(1-p)/(p*(1-`c'*p*(1-p))) if p<`pmax'

```

```

twoway (scatter dx2 p, msymbol(Oh) mcolor(black) msize(vsmall)) ///
(line crb_d crb_g p, sort lcolor(gs0 gs0) lpat(solid solid) lw(thick thick)) ///
, ylab(0 (2) `maxy',angle(hori) nogrid) xlab(0 (.1) 1) yline(4,lcolor(red) lpat(dash)) ///
xtitle("P{sub:j}") ytitle("") legend(off) ///
text(`maxy' 0 "{{Delta}}{{chi}}{sub:j}{sup:2}",place(e))
*graph export dx2_P2_crb.png,replace

```

**** Figure 5 : Delta D2**

```

twoway scatter dd p,msymbol(Oh) mcolor(black) msize(vsmall) ylab(0 (2) `maxy',angle(hori) nogrid)
xlab(0 (.1) 1) yline(4,lcolor(red) lpat(dash)) xtitle("P{sub:j}") l1title(" ") ytitle("")
text(`maxy' 0 "{{Delta}}D{sub:j}{sup:2}",place(e))
*graph export dd_P.png,replace

```

V.3. Influence sur l'estimation des coefficients

******* Figure 6 : Delta beta**

```

qui sum db
local mdb=round(`r(max)',.1)
twoway scatter db p,msymbol(Oh) mcolor(black) msize(vsmall) ylab(0 (.2) `mdb',angle(hori) nogrid)
xlab(0 (.1) 1) yline(.15,lcolor(red) lpat(dash)) xtitle("P{sub:j}") l1title(" ") ytitle("")
text(`mdb' 0 "{{Delta}}{{beta}}{sub:j}",place(e)) text(=`mdb'+.02' 0.03 "^")
*graph export db_P.png,replace

```

V.4. Examens des points influents

****** Tableau 7 : tableau des points influents**

```

*tab pat if dd>4 & dd!=.
bys pat : gen sel=1 if _n==1
bys pat : gen nbpat=_N
bys pat:egen pobs=mean(ct)

format pobs %4.2f
format npat agea ctub tabfc afcs ainf %4.0f

gsort -dd
list pat nbpat ctub agea tabfc afcs ainf db dx2 dd pobs p if sel==1 & (dd>4 | dx2>4) & dd!=. &
dx2!=.,noobs

```

****** Tableau 8 : Variations des coefficients pour les profils les plus influents**

* Les lignes suivantes calculent les éléments nécessaires au tableau, les pourcentages de variations des betas doivent ensuite être calculés "à la main"

```

foreach num in 37 189 61 170 44 251 339 {
    di in red "num pat = " `num'
    logit ct ctub agea i.tabfc afcs ainf if pat!=`num'
    estat gof,group(10)
    chi2_res
}

```

```

di in red "num pat ≠ 37 185 61 169 43 251 260"
logit ct ctub agea i.tabfc afcs ainf if !inlist(pat, 37, 185, 61, 169, 43, 251, 260)
estat gof,group(10)
chi2_res

```