Construction du panel Enfants EDP

Ce document retrace la manière dont une base Enfant a été construite à partir de l'Echantillon Démographique Permanent, millésime 2019. Il a été rédigé par Claire Vandendriesche et validé par Carole Bonnet et Anne Solaz. On présente la construction des variables et les codes Stata associés qui permettent de construire cette base Enfants[1].

Carole Bonnet, Anne Solaz et Claire Vandendriessche[2]

Table des matières

[1] Ce travail a bénéficié d'un financement de France Stratégie, du Haut Conseil de la Famille, de l'Enfance et de l'Âge (HCFEA), ainsi que du soutien de l'équipex LifeObs (ANR-21-ESRE-0037) et du projet Big_stat (ANR-16-CE41-0007).

[2] Merci de citer ce travail de la manière suivante : Bonnet Carole, Solaz Anne et Claire Vandendriessche, 2022, « Construction du panel Enfants EDP », Note technique.

Déclaration et formatage de la base

Dans Stata, on déclare le répertoire contenant tous les fichiers de la base comme répertoire de travail et on traduit en Unicode les fichiers de données Stata pour tenir compte des accentuations françaises dans les labels de variables et de valeurs de variables. Ce formatage Unicode n'a besoin d'être réalisé qu'une seule fois.

```
global basesdir C:\Users\Public\Documents\Bases Brutes STATA
global wd C:\Users\Public\Documents\Claire VDD
cd "$basesdir"
clear
unicode encoding set Latin1
unicode analyze edp_be2019_*.dta
unicode translate edp_be2019_*.dta
```

(Note : les extraits de code comme celui-ci peuvent être automatiquement colorés en langage Stata grâce au logiciel Visual Studio Code).

Construction des variables fixes et délimitation aux enfants

Ces variables construites sont invariantes dans le temps du futur panel. Il s'agit des variables suivantes :

- Date de naissance
- Année de naissance
- Mois de naissance
- Jour de naissance
- Date de décès
- Âge en 2019
- Âge exact fin 2019
- Naissance à l'étranger
- Dates de naissance des parents

On se sert de l'année de naissance construite pour ne garder que les individus nés à partir de 1994, c'est-à-dire ceux observables au moins une fois pendant leur minorité dans les années fiscales 2011-2019. Cela réduit la taille de la base de 3 865 553 individus à 960 543 enfants.

```
*Ouverture de la table individu :
use edp_be2019_individu.dta,clear

*On construit les variables fixes :
gen ddn=date(nai_date,"YMD")
format ddn %td
gen ddd=date(dec_date,"YMD")
format ddd %td
gen anaisi=year(ddn)
gen mnaisi=month(ddn)
gen jnaisi=day(ddn)
la var ddn "Date de naissance"
la var ddd "Date de décès"
la var anaisi "Année de naissance"
la var mnaisi "Mois de naissance"
la var jnaisi "Jour de naissance"
drop if anaisi<1994 //on délimite la base aux individus qui seront au moins
une fois mineurs sur la période 2011-2019
order anaisi mnaisi jnaisi ddn,after(nai_date)
gen age=2019-year(ddn)
la var age "Âge de l'enfant EDP en 2019"
gen ageexact=age+(365-doy(ddn))/365
la var ageexact "Âge exact de l'enfant EDP fin 2019"
gen byte bornabroad=(substr(nai_lieu,1,2)=="99") if !mi(nai_lieu)
la var bornabroad "Personne née à l'étranger"
la var nbnais "Existence d'un acte de naissance ou d'adoption de l'individu
EDP"
*Dates de naissance des parents:
replace mere_ind_nais_date="1975-04-25" if id_diff=="0000202906" //correction
: 1875->1975
```

```
replace mere_ind_nais_date="" if id_diff=="0000202906" //année de naissance
mère >= année de naissance enfant EDP
replace pere_ind_nais_date="" if
inlist(id_diff,"0000593913","0000876543","0002247145") //année de naissance
père >= année de naissance enfant EDP
gen ddnm=date(mere_ind_nais_date,"YMD")
gen ddnp=date(pere_ind_nais_date,"YMD")
format ddnm ddnp %td
gen anaism=year(ddnm)
gen anaisp=year(ddnp)
order ddnm ddnp anaism anaisp,after(pere_ind_nais_date)
la var ddnm "Date de naissance de la mère"
la var ddnp "Date de naissance du père"
la var anaism "Année de naissance de la mère"
la var anaisp "Année de naissance du père"
```

On apparie la table « naissance » et toutes les variables qu'elle contient, que l'on préfixe par « n_ » :

```
*Coller avec la table naissance (--préfixe N--), pour récupérer les infos du
bulletin de naissance:
tempfile naissance
preserve
use edp_be2019_naissance.dta,clear
ren * n_*
ren n_id_diff id_diff
save `naissance'
restore
merge 1:1 id_diff using `naissance',gen(_merge_naissance) keep(1 3)
la var _merge_naissance "Appariement avec la table Naissance"
```

Parmi les variables importées depuis la table naissance, les conditions d'accouchement permettent d'identifier la présence de jumeaux :

```
*Récupération d'info sur les naissances gémellaires/multiples:
encode n_ctx_catacc,generate(gemellaire)
recode gemellaire (1 2 3 5=.d "Jumeau décédé à la naissance")(4 6 7=1
"Jumeau")(8 9=.t "Tri-/Quadruplets")(10=0 "Naissance simple"),gen(n_jumeaux)
drop gemellaire
la var n_jumeaux "L'individu EDP est issu d'une naissance gemellaire"
order n_* _merge_naissance,after(bornabroad)
```

On apparie ensuite la table « descendance » pour récupérer les identifiants EDP des parents EDP, que l'on préfixera par « d_ ». Ceci modifie la structure de la base que nous avons, car le nombre de lignes passera d'une par enfant ayant 0 ou 1 parent EDP, à deux par enfant ayant les 2 parents EDP. On passe alors à 961 650 lignes.

```
*Coller avec la table descendance (--préfixe D--), pour récupérer les
identifiants EDP des parents EDP
tempfile descendance app
preserve
use edp_be2019_descendance.dta,clear
keep id_evt_diff id_diff ope_type nb_edp
ren (id_diff ope_type nb_edp)(d_parent_id_diff d_ope_type d_nb_edp)
ren id_evt_diff n_id_evt_diff
save `descendance'
restore
preserve
keep if nbnais==0
gen str10 d_parent_id_diff=""
gen str10 d_ope_type=""
gen double d_nb_edp=.
save `app'
restore
drop if nbnais==0
merge 1:m n_id_evt_diff using `descendance',gen(_merge_descendance) keep(1 3)
//c'est un merge one-to-many, on va donc accroître le nb de lignes par enfant
EDP par le nb de parents EDP impliqués dans sa naissance (généralement x1 mais
parfois x2)
label define nbedpn 1 "Seul l'enfant est EDP" 2 "Un des deux parents est EDP"
3 "Les deux parents sont EDP"
label define nbedpd 1 "Seul le parent est EDP" 2 "L'autre parent ou l'enfant
est EDP" 3 "L'autre parent et l'enfant sont EDP"
label values n_nb_edp nbedpn
label values d_nb_edp nbedpd
la var _merge_descendance "Appariement avec la table Descendance"
la var n_nb_edp "Nombre d'individus EDP associés à l'évènement"
la var d_nb_edp "Nombre d'individus EDP associés à l'évènement"
append using `app'
```

On reforme la base pour ne garder qu'une ligne par enfant, c'est-à-dire 960 543 lignes. Pour cela, on met l'identifiant EDP des deux parents sur la même ligne :

```
egen byte parent_type=group(d_ope_type),lname(parent_type)
tempfile noedpparent
preserve
keep if parent_type==.
ren d_parent_id_diff d_parent_id_diff1
ren d_ope_type d_ope_type1
clonevar d_parent_id_diff2=d_parent_id_diff1
clonevar d_ope_type2=d_ope_type1
drop parent_type
save `noedpparent'
restore
keep if parent_type<.
```

```
reshape wide d_parent_id_diff d_ope_type, i(id_diff) j(parent_type)
append using `noedpparent'
order d_* _merge_descendance,after(bornabroad)
```

Panélisation de la base

Appariement aux données fiscales

On apparie à la base les tables fiscales individuelles de 2011 à 2019. A raison d'un enfant pour 9 années fiscales, la base voit son nombre de lignes multiplié par au moins 9. En réalité, chaque enfant pouvant faire l'objet de plus d'une déclaration fiscale, il va y avoir plus d'une ligne par individu*année pour le moment. Les variables des tables fiscales individuelles sont préfixées par « fi_ » pour « fisc individu ». A l'issue de cette première phase de la panélisation, il y aura 8 861 986 lignes.

```
*Début de la panélisation de la base : on colle les tables fisc_individu 2011-
2019 (--préfixe FI--).
set more off
tempfile base
save `base',replace
forvalues i=2011/2019 {
    tempfile be`i' fi`i'
    preserve
    use edp_be2019_fisc_individu_`i'.dta,clear
    ren * fi_*
    ren fi_id_diff id_diff
    ren fi_an_fisc an_fisc
    save `fi`i'',replace
    restore
    merge 1:m id_diff using `fi`i'',gen(_merge_fi) keep(1 3)
    duplicates tag id_diff ,gen(fi_multiple_fiscs)
    save `be`i'',replace
    use `base',clear
}
append using `be2011' `be2012' `be2013' `be2014' `be2015' `be2016' `be2017'
`be2018' `be2019',gen(_app_fi)
drop if _app_fi==0
label define _app_fi 1 "2011" 2 "2012" 3 "2013" 4 "2014" 5 "2015" 6 "2016" 7
"2017" 8 "2018" 9 "2019"
label values _app_fi _app_fi
la var fi_multiple_fiscs "Nombre de déclarations fiscales excédentaires dans
l'année"
la var _app_fi "Vague de panel (1=2011 ... 9=2019)"
la var _merge_fi "Appariement avec les données socio-fiscales"
```

Elimination des déclarations fiscales multiples

On crée par la suite la variable clé de foyer fiscal *fi_idfoy*, ainsi qu'une variable permettant d'identifier si un enfant a été déclaré en résidence alternée une année donnée (*fi_atleast1h*). Ces variables seront utiles pour éliminer les observations fiscales excédentaires du même enfant.

```
*On crée une indicatrice de foyer fiscal:
egen fi_idfoy=concat(fi_id_fisc_foy_diff fi_ordrefip)
```

```
la var fi_idfoy "Identifiant de foyer fiscal (id_fisc_foy_diff+ordrefip)"

*On construit une variable pour filtrer les individus*années qui ont été
déclarés à charge par au moins un des deux déclarants:
gen byte typecharge=(fi_t_charge=="H") & fi_t_charge!=""
bys id_diff _app_fi: egen fi_atleast1h=max(typecharge) //identifie les
individus ayant au moins une entrée en tant que garde alternée pour une année
donnée (t_charge=H)
drop typecharge
la var fi_atleast1h "Au moins une garde alternée déclarée pour un
enfant*année"
```

On identifie les enfants qui ont été l'objet de plusieurs déclarations fiscales, pour chaque année fiscale :

```
duplicates tag id_diff _app_fi, gen(tag)
ta _app_fi tag
```

Le résultat, indiqué dans le Tableau 1, fait constater qu'à chaque année fiscale, il existe un certain nombre de doublons d'enfant*année. En 2011, il y a 958 862 enfants observés fiscalement au mieux une fois (voire pas observés), et 1 681 enfants qui font l'objet de deux déclarations fiscales, soit 3 362 observations-doublons. En 2013, 21 enfants sont observés fiscalement trois fois, soit 63 observations-doublons. Et en 2019, 1 enfant est observé fiscalement six fois, soit 6 observations-doublons.

Tableau 1- Observations fiscales excédentaires par année fiscale

Année fiscale	Nombre DE DOUBLONS						Total
	0	1	2	3	4	5	
2011	958 862	3 362	0	0	0	0	**962 224**
2012	958 247	4 592	0	0	0	0	**962 839**
2013	956 211	8 622	63	0	0	0	**964 896**
2014	952 881	15 178	216	4	0	0	**968 279**
2015	937 972	44 058	1 509	156	0	0	**983 695**
2016	931 230	57 002	2 280	208	0	0	**990 720**
2017	919 200	79 894	4 014	224	10	0	**1 003 342**
2018	918 938	80 584	3 687	328	10	0	**1 003 547**
2019	900 878	115 122	5 937	476	25	6	**1 022 444**
Total	**8 434 419**	**408 414**	**17 706**	**1 396**	**45**	**6**	**8 861 986**

L'objectif, par la suite, sera d'utiliser une méthode permettant de ne garder qu'**au plus deux observations fiscales du même enfant par année**. C'est-à-dire de réduire ce tableau à ses deux premières colonnes, en éliminant les observations excédentaires des individus triplement, quadruplement, quintuplement, ou sextuplement identifiés. Pour cela, on utilisera la variable *type_pres*, une variable de la base EDP, pour nous permettre d'éliminer les observations « hors champ » (*type_pres*=9) des individus EDP multi-identifiés une année donnée.

Le but de cette procédure est de préserver l'information la plus fraîche et celle relative aux résidences alternées.

```
1. gsort id_diff _app_fi fi_type_pres - fi_t_charge fi_ordrefip
   fi_type_fisc
2. list id_diff fi_type_pres fi_t_charge fi_ordrefip if tag>1 & _app_fi==3
   //21 individus avec 1 observation excédentaire
3. by id_diff _app_fi: gen ntag=_n
4. drop if ntag>2 //6631 obs. excédentaires éliminées
5. drop tag ntag
```

Nous faisons la démonstration qui suit pour l'année 2013. Le résultat de l'exécution de la ligne de code 2 fait apparaître le Tableau 2, où les observations barrées correspondent aux observations excédentaires éliminées par l'exécution de la ligne 4 :

Tableau 2 - Exemple de multi-identifications fiscales

id_diff	fi_type_pres	fi_t_charge	fi_ordrefip
82874	1		A
82874	2	F	A
~~**82874**~~	~~9~~		~~A~~
400089	1		A
400089	2		A
~~**400089**~~	~~9~~	~~F~~	~~B~~
644623	1		A
644623	2	H	A
~~**644623**~~	~~9~~	~~H~~	~~A~~
720082	1		A
720082	2	F	A
~~**720082**~~	~~9~~		~~A~~
731982	1		A
731982	2	H	A
~~**731982**~~	~~9~~	~~H~~	~~B~~
762720	1		A
762720	2	H	A
~~**762720**~~	~~9~~	~~F~~	~~A~~
836812	1		D
836812	2	F	A
~~**836812**~~	~~9~~		~~C~~
1061534	1		A
1061534	2		B
~~**1061534**~~	~~9~~	~~F~~	~~A~~
1169206	1		A
1169206	2	F	B
~~**1169206**~~	~~9~~		~~A~~
1436693	1		A
1436693	2	H	A
~~**1436693**~~	~~9~~	~~H~~	~~A~~
1556173	1		A
1556173	2	H	A
~~**1556173**~~	~~9~~		~~A~~
1596522	1		A
1596522	2		B
~~**1596522**~~	~~9~~	~~F~~	~~A~~
1687379	1		A
1687379	2		A
~~**1687379**~~	~~9~~		~~A~~
1963530	1		A
1963530	2		B
~~**1963530**~~	~~9~~		~~C~~
2510661	1		A
2510661	2		B
~~**2510661**~~	~~9~~	~~F~~	~~A~~
2755551	1		A
2755551	2		A
~~**2755551**~~	~~9~~	~~F~~	~~B~~
2978608	1		A
2978608	2	H	A
~~**2978608**~~	~~9~~	~~F~~	~~A~~
3218969	1		A
3218969	2		B
~~**3218969**~~	~~9~~	~~F~~	~~A~~
3419695	1		A
3419695	2		B
~~**3419695**~~	~~9~~	~~F~~	~~B~~
3475138	1		A

3475138	2	F	A
~~**3475138**~~	~~9~~		~~A~~
4398486	1	F	B

4398486	9		B
~~**4398486**~~	~~9~~		~~C~~

A l'issue de ces éliminations, la base est réduite à 8 855 355 observations. Il y a toutefois toujours 960 543 enfants, ce que l'on peut savoir en exécutant la ligne suivante :

```
unique id_diff
```

On supprime par la suite l'observation excédentaire des individus apparaissant la même année fiscale 2 fois sur un foyer fiscal identique. On éliminera l'observation excédentaire ayant la valeur de *type_fisc* la plus grande.

```
1. duplicates tag id_diff fi_idfoy _app_fi,gen(tagfoy)
2. ta _app_fi tagfoy //509 individus ont 1 obs. excédentaire du même foyer
3. sort _app_fi tagfoy id_diff fi_type_pres
4. by _app_fi tagfoy id_diff : gen ntag=_n
5. drop if ntag==2 & tagfoy==1 //509 obs. excédentaires éliminées
6. drop tagfoy ntag
```

L'exécution de la ligne 2 fait apparaître le Tableau 3 :

Tableau 3 - Observations excédentaires de foyers fiscaux

	Nombre de foyers-doublons		
Année fiscale	**0**	**1**	**Total**
2011	962 222	2	**962 224**
2012	962 835	4	**962 839**
2013	964 873	2	**964 875**
2014	968 191	14	**968 205**
2015	983 100	14	**983 114**
2016	989 836	20	**989 856**
2017	1 001 586	300	**1 001 886**
2018	1 001 780	368	**1 002 148**
2019	1 019 914	294	**1 020 208**
Total	**8 854 337**	**1 018**	**8 855 355**

A l'issue de la ligne de code 5, on a 8 854 846 observations, et toujours 960 543 enfants.

Au sujet des foyers fiscaux (*id_fisc_foy_diff* + *ordrefip*) et des logements fiscaux (*id_fisc_log_diff*) plusieurs remarques :

Il ne peut désormais y avoir deux fois le même foyer chez un individu*année. Il peut y avoir toutefois deux fois le même logement pour un individu*année. Et il peut y avoir 1 ou plusieurs logements distincts chez le même individu*année. Dit autrement : il ne peut y avoir qu'un logement unique par foyer, mais il peut y avoir plusieurs foyers distincts dans un logement. Un individu peut être identifié,

pour une année donnée, dans un ou deux foyers fiscaux distincts, il ne peut jamais être identifié deux fois dans le même foyer. Un individu peut être identifié, pour une année donnée, dans un ou deux logements fiscaux distincts ou identiques. Si ses logements sont distincts, ses foyers sont distincts aussi ; si ses logements sont identiques, ses foyers sont distincts quand même.

Reformation du panel

On reforme alors la base, de telle sorte à n'avoir plus qu'une ligne par enfant*année. Les informations fiscales relatives au 2e foyer fiscal se retrouvant sur la même ligne que celle relative au 1er foyer fiscal. On modifie toutefois les préfixes des variables fiscales pour distinguer les deux (« fi1_ » et « fi2_ ») :

```
1. //On reshape pour supprimer les doublons d'obs en dédoublant le nb de
   colonnes par foyer fiscal:
2. sort id_diff _app_fi
3. egen long id=group(id_diff _app_fi)
4. sort id fi_type_pres //Cette ligne permet d'ordonner les variables fi1_
   et fi2_ en fonction de la valeur prise par type_pres.
5. by id : gen byte foyid=_n
6. qui reshape wide fi_* , i(id) j(foyid)
7. fre _app_fi //normalement 960543 enfants pour chaque vague fiscale
8. ren (fi_*1 fi_*2) (fi1_* fi2_*)
9. order fi?_*,after(an_fisc)
```

A l'exécution de la ligne de code 7, le Tableau 4 doit apparaître, permettant de vérifier qu'il y a bien 960 543 enfants pour chaque année fiscale, soit au total 8 644 887 observations.

Tableau 4 - Nombre d'observations par année fiscale

Année fiscale	Effectifs
2011	960 543
2012	960 543
2013	960 543
2014	960 543
2015	960 543
2016	960 543
2017	960 543
2018	960 543
2019	960 543
Total	**8 644 887**

On construit la variable *fisum*, qui indique pour chaque enfant, le nombre d'années fiscales auxquelles il a pu être apparié :

```
//Nombre d'années fiscales appariées:
gen byte fi=(_merge_fi>1)
bys id_diff: egen byte fisum=total(fi)
drop fi
la var fisum "Nombre d'années fiscales appariées"
```

Construction de variables du nombre de logements et foyers par individu*année :

Le code suivant permet d'identifier dans combien de logements distincts l'enfant vit, et dans combien de foyers fiscaux il est déclaré.

```
1. *On construit des indicatrices de vie dans 1 ou 2 logements/foyers
   fiscaux:
2. gen byte fi_nbfoys=2 if fi1_idfoy!=fi2_idfoy & !mi(fi1_idfoy,fi2_idfoy)
3. replace fi_nbfoys=1 if (!mi(fi1_idfoy) & mi(fi2_idfoy)) | (mi(fi1_idfoy)
   & !mi(fi2_idfoy))
4. gen byte fi_nblogs=2 if fi1_id_fisc_log_diff!=fi2_id_fisc_log_diff &
   !mi(fi1_id_fisc_log_diff,fi2_id_fisc_log_diff)
5. replace fi_nblogs=1 if ( !mi(fi1_id_fisc_log_diff) &
   mi(fi2_id_fisc_log_diff) & mi(fi2_idfoy)) | (mi(fi1_id_fisc_log_diff) &
   mi(fi1_idfoy) & !mi(fi2_id_fisc_log_diff)) |
   (fi1_id_fisc_log_diff==fi2_id_fisc_log_diff &
   !mi(fi1_id_fisc_log_diff,fi2_id_fisc_log_diff))
6. replace fi_nblogs=1 if fi_nblogs==. & fi_nbfoys==1 //on suppose qu'il
   n'y a qu'un logement dans la mesure où on n'a pu apparier qu'un seul
   foyer
7. ta fi_nblogs fi_nbfoys,m //pour 14013 observations, on ne peut pas
   savoir avec confiance s'ils sont dans 1 ou 2 logements fiscaux (dont
   l'un des identifiants ou les deux sont missings)
8. order fi_nblogs fi_nbfoys,before(_merge_fi)
9. la var fi_nblogs "Nombre de logements distincts identifiés"
10.la var fi_nbfoys "Nombre de foyers distincts identifiés"
```

La ligne de code 7 fait apparaître le tableau suivant :

Tableau 5- Nombre d'observations par nombre de logements et foyers

		Nombre de foyers fiscaux distincts			
		1	**2**	**.**	**Total**
Nombre de logements fiscaux distincts	**1**	4 920 105	69 795	0	4 989 900
	2	0	126 150	0	126 150
	.	0	14 013	3 514 824	3 528 837
	Total	4 920 105	209 958	3 514 824	8 644 887

Par construction, ayant éliminé les declarations triples, quadruples, etc., il ne peut y avoir au minimum qu'un, et pas plus de deux foyers fiscaux par enfant*année. Idem pour le nombre de logements.

Enrichissement du panel des données socio-fiscales (DSF)

Revenus du ménage

Par la suite, on vient apparier les données de la table “fisc revenus” au panel, logement fiscal par logement fiscal et année par année. Les variables issues de cette table seront préfixées “fr_1” et “fr_2” selon qu’elles correspondent aux données du logement 1 ou du logement 2 :

```
*On peut alors coller la base fisc revenu sur chaque logement fiscal (--
préfixe FR--):
set more off
tempfile fr1 fr2
gen byte _merge_fr1=.
gen byte _merge_fr2=.
forvalues i=2011/2019 {
    local i2=`i'-2010
    preserve
    use edp_be2019_fisc_revenu_`i'.dta,clear
    ren * fr1_*
    ren fr1_id_fisc_log_diff fi1_id_fisc_log_diff
    ren fr1_an_fisc an_fisc
    save `fr1',replace
    ren fr1_* fr2_*
    ren fi1_id_fisc_log_diff fi2_id_fisc_log_diff
    save `fr2',replace
    restore
    merge m:1 fi1_id_fisc_log_diff an_fisc using `fr1',keep(1 3 4) update
    replace _merge_fr1=_merge if _app_fi==`i2'
    drop _merge
    merge m:1 fi2_id_fisc_log_diff an_fisc using `fr2',keep(1 3 4) update
    replace _merge_fr2=_merge if _app_fi==`i2'
    drop _merge
}
label values _merge_fr1 _merge_fr2 _merge
format an_fisc %ty
order _merge_fr? ,last
la var _merge_fr1 "Appariement avec la table revenus du logement 1"
la var _merge_fr2 "Appariement avec la table revenus du logement 2"
```

Numérisation des variables

Afin de commencer à économiser de la mémoire, on peut numériser les variables, qui pour l’instant sont toutes au format *string* :

```
destring *,replace
```

Type de ménage

À l’aide de la variable *typmen9* appariée, on peut construire une variable de type de ménage simplifiée pour les deux logements :

```
*On construit des indicatrices sur le type de ménage :
recode fr1_typmen9 (11 12 30=.s "Sans enfant")(21 22=1 "Famille
monoparentale")(41 42 43=2 "Couple avec enfant(s)")(50=.c "Ménage
complexe")(99=.),gen(fr1_typmen)
recode fr2_typmen9 (11 12 30=.s "Sans enfant")(21 22=1 "Famille
monoparentale")(41 42 43=2 "Couple avec enfant(s)")(50=.c "Ménage
complexe")(99=.),gen(fr2_typmen)
order fr1_typmen,after(fr1_typmenr)
order fr2_typmen,after(fr2_typmenr)
la var fr1_typmen "Type de ménage simplifié - logement 1"
la var fr2_typmen "Type de ménage simplifié - logement 2"
```

Informations fiscales relatives aux déclarants de l'enfant

Par la suite, on apparie aux enfants*années les informations relatives à son ou ses deux déclarants fiscaux, pour chaque foyer fiscal où il est identifié. Nous récupérons donc dans les tables « fisc individu » toutes les variables concerant au maximum 4 personnes : le déclarant du premier foyer fiscal (préfixe « fi1d_ »), le partenaire déclarant du premier foyer fiscal (préfixe « fi1p_ »), le déclarant du deuxième foyer fiscal (préfixe « fi2d_ »), le partenaire déclarant du deuxième foyer fiscal (préfixe « fi2p_ »).

```
*Search for declarants' information (or "declarant & partner", for fiscal
years 2011 & 2012):
/*L'appariement se fait sur le foyer fiscal et l'année*/
//Match declarant 1 :
set more off
gen byte _merge_fi1d=.
gen byte _merge_fi2d=.
tempfile fid
forvalues i=2011/2019 {
    di "`i'"
    local i2=`i'-2010
    preserve
    use edp_be2019_fisc_individu_`i'.dta,clear
    format an_fisc %ty
    keep if inlist(type_fisc,"1","01")
    gen ddnstring=anais+"-"+mnais+"-"+jnais if !mi(anais,mnais,jnais)
    gen ddn=date(ddnstring,"YMD")
    format ddn %td
    egen fi1_idfoy=concat(id_fisc_foy_diff ordrefip)
    clonevar fi2_idfoy=fi1_idfoy
    drop if mi(fi1_idfoy)
    duplicates drop fi1_idfoy type_fisc,force
    unab toloop : id_diff sexe lien_familial ddn type_fisc cideci zoxyzd
dacoed t_charge codnais i_fisc_* type_decl type_pres poids_fideli nbptr
    foreach v of varlist `toloop' {
        ren `v' fi1d_`v'
        clonevar fi2d_`v'=fi1d_`v'
```

```
    }
    save `fid',replace
    restore
    merge m:1 fi1_idfoy an_fisc using `fid',keep(1 3 4 5) keepusing(fi1d_*)
update
    replace _merge_fi1d=_merge if _app_fi==`i2'
    drop _merge
    merge m:1 fi2_idfoy an_fisc using `fid',keep(1 3 4 5) keepusing(fi2d_*)
update
    replace _merge_fi2d=_merge if _app_fi==`i2'
    drop _merge
}
label values _merge_fi1d _merge_fi2d _merge
order _merge_fi1d,after(fi1d_ddn)
order _merge_fi2d,last
la var _merge_fi1d "Appariement avec le déclarant 1 du foyer 1"
la var _merge_fi2d "Appariement avec le déclarant 1 du foyer 2"

//Match declarant 2 (partner):
set more off
gen byte _merge_fi1p=.
gen byte _merge_fi2p=.
tempfile fip
forvalues i=2011/2019 {
    di "`i'"
    local i2=`i'-2010
    preserve
    use edp_be2019_fisc_individu_`i'.dta,clear
    format an_fisc %ty
    keep if inlist(type_fisc,"2","02") //Select declarant 2 (formerly
"partner")
    gen ddnstring=anais+"-"+mnais+"-"+jnais if !mi(anais,mnais,jnais)
    gen ddn=date(ddnstring,"YMD")
    format ddn %td
    egen fi1_idfoy=concat(id_fisc_foy_diff ordrefip)
    clonevar fi2_idfoy=fi1_idfoy
    drop if mi(fi1_idfoy)
    duplicates drop fi1_idfoy type_fisc,force
    unab toloop : id_diff sexe lien_familial ddn type_fisc cideci zoxyzd
dacoed t_charge codnais i_fisc_* type_decl type_pres poids_fideli nbptr
    foreach v of varlist `toloop' {
        ren `v' fi1p_`v'
        clonevar fi2p_`v'=fi1p_`v'
    }
    save `fip',replace
    restore
    merge m:1 fi1_idfoy an_fisc using `fip',keep(1 3 4 5) keepusing(fi1p_*)
update
```

```
    replace _merge_fi1p=_merge if _app_fi==`i2'
    drop _merge
    merge m:1 fi2_idfoy an_fisc using `fip',keep(1 3 4 5) keepusing(fi2p_*)
update
    replace _merge_fi2p=_merge if _app_fi==`i2'
    drop _merge
}
label values _merge_fi1p _merge_fi2p _merge
order _merge_fi1p,after(fi1p_ddn)
order _merge_fi2p,last
la var _merge_fi1p "Appariement avec le déclarant 2 du foyer 1"
la var _merge_fi2p "Appariement avec le déclarant 2 du foyer 2"
```

Déclaration du panel

L'analyse en panel requiert la déclaration de celui-ci à l'aide de la commande *xtset*. Les variables de déclaration du panel seront *idi* (variable numérique issue de l'identifiant individu *id*_diff) et *annee*, correspondant à l'année fiscale. On crée également la variable *_base1* qui est une sous-base, permettant d'identifier tous les enfants EDP qui ont au moins un parent EDP.

```
*Déclarer le panel :
count if mi(real(id_diff)) //13671 observations of people whose id_diff is
nonnumeric (starts with an X), i.e. some individuals born before 2010
gen str10 idi=subinstr(id_diff,"X",".",1) //recode the leading "X" in id_diff
with a "." in order to be numeric
destring idi,replace
gen int annee=2010+_app_fi
xtset idi annee,yearly
ren an_fisc fi_an_fisc
order fi_an_fisc ,before(_merge_fi)
la var id "Clé enfant EDP*année fiscale"
la var id_diff "Clé enfant EDP (texte)"
la var idi "Clé enfant EDP (numérique)"
la var annee "Clé année fiscale"
gen byte _base1=(_merge_descendance==3)
la var _base1 "Enfant EDP ayant au moins 1 parent EDP"
order id_diff idi id annee _app_fi _base1,first
```

Par la suite, il sera possible qu'utiliser des opérateurs temporels tels que « L. » ou « F. » fasse apparaître l'erreur 5 suivante « not sorted ». Pour y remédier, il suffit généralement de trier la base selon la clé du panel :

```
sort idi annee
```

Âge à une année fiscale donnée

On construit une variable d'âge de l'enfant, qui varie avec l'année fiscale d'observation, et qui tienne compte du décès éventuel de l'enfant.

```
*Âge en panel :
gen int yeardc=year(ddd)
gen int age_t=annee-year(ddn)
replace age_t=.u if age_t<0 //unborn, pas encore nés
replace age_t=.d if annee>yeardc //décédés
drop yeardc
order age_t,after(ageexact)
la var age_t "Âge à année fiscale donnée"
```

Revenus détaillés par déclarant fiscal

Chacun des déclarants fiscaux déclarant des revenus individuels (salaires, allocations chômage, etc.), il est possible d'apparier le panel à la table « fisc revdet » pour collecter ces revenus détaillés. On utilisera les préfixes suivants pour clarifier l'origine de ces variables :

- « fi1drd_ » : Revenus détaillés du déclarant du premier foyer fiscal
- « fi1prd_ » : Revenus détaillés du partenaire déclarant du premier foyer fiscal
- « fi2drd_ » : Revenus détaillés du déclarant du deuxième foyer fiscal
- « fi2prd_ » : Revenus détaillés du partenaire déclarant du deuxième foyer fiscal

```
*Rajouter les revenus détaillés par déclarant fiscal:
set more off
destring fi1d_type_fisc fi1p_type_fisc fi2d_type_fisc fi2p_type_fisc,replace
egen fi1d_id=concat(fi1_id_fisc_foy_diff fi1_ordrefip fi1d_type_fisc) //clés
d'appariement
egen fi1p_id=concat(fi1_id_fisc_foy_diff fi1_ordrefip fi1p_type_fisc)
egen fi2d_id=concat(fi2_id_fisc_foy_diff fi2_ordrefip fi2d_type_fisc)
egen fi2p_id=concat(fi2_id_fisc_foy_diff fi2_ordrefip fi2p_type_fisc)
preserve
tempfile revdet
use edp_be2019_fisc_revdet_2011.dta,clear
append using edp_be2019_fisc_revdet_2012.dta edp_be2019_fisc_revdet_2013.dta
edp_be2019_fisc_revdet_2014.dta edp_be2019_fisc_revdet_2015.dta
edp_be2019_fisc_revdet_2016.dta edp_be2019_fisc_revdet_2017.dta
edp_be2019_fisc_revdet_2018.dta edp_be2019_fisc_revdet_2019,gen(_app_fi)
replace _app_fi=_app_fi+1
destring id_fisc_foy_diff ordrefip type_fisc,replace
egen fi1d_id=concat(id_fisc_foy_diff ordrefip type_fisc)
clonevar fi1p_id=fi1d_id
clonevar fi2d_id=fi1d_id
clonevar fi2p_id=fi1d_id
save `revdet',replace
restore
merge m:1 fi1d_id _app_fi using `revdet', keep(1 3) keepusing(rev_princ ysali
ychoi yrsti yalri yragi ybici ybnci) gen(_merge_fi1drd)
```

```
ren (rev_princ ysali ychoi yrsti yalri yragi ybici ybnci) (fi1drd_rev_princ
fi1drd_ysali fi1drd_ychoi fi1drd_yrsti fi1drd_yalri fi1drd_yragi fi1drd_ybici
fi1drd_ybnci)
merge m:1 fi1p_id _app_fi using `revdet', keep(1 3) keepusing(rev_princ ysali
ychoi yrsti yalri yragi ybici ybnci) gen(_merge_fi1prd)
ren (rev_princ ysali ychoi yrsti yalri yragi ybici ybnci) (fi1prd_rev_princ
fi1prd_ysali fi1prd_ychoi fi1prd_yrsti fi1prd_yalri fi1prd_yragi fi1prd_ybici
fi1prd_ybnci)
merge m:1 fi2d_id _app_fi using `revdet', keep(1 3) keepusing(rev_princ ysali
ychoi yrsti yalri yragi ybici ybnci) gen(_merge_fi2drd)
ren (rev_princ ysali ychoi yrsti yalri yragi ybici ybnci) (fi2drd_rev_princ
fi2drd_ysali fi2drd_ychoi fi2drd_yrsti fi2drd_yalri fi2drd_yragi fi2drd_ybici
fi2drd_ybnci)
merge m:1 fi2p_id _app_fi using `revdet', keep(1 3) keepusing(rev_princ ysali
ychoi yrsti yalri yragi ybici ybnci) gen(_merge_fi2prd)
ren (rev_princ ysali ychoi yrsti yalri yragi ybici ybnci) (fi2prd_rev_princ
fi2prd_ysali fi2prd_ychoi fi2prd_yrsti fi2prd_yalri fi2prd_yragi fi2prd_ybici
fi2prd_ybnci)

order fi1d_id fi1drd_rev_princ fi1drd_ysali fi1drd_ychoi fi1drd_yrsti
fi1drd_yalri fi1drd_yragi fi1drd_ybici fi1drd_ybnci _merge_fi1drd,
after(_merge_fi1d)
order fi1p_id fi1prd_rev_princ fi1prd_ysali fi1prd_ychoi fi1prd_yrsti
fi1prd_yalri fi1prd_yragi fi1prd_ybici fi1prd_ybnci _merge_fi1prd,
after(_merge_fi1p)
order fi2d_id fi2drd_rev_princ fi2drd_ysali fi2drd_ychoi fi2drd_yrsti
fi2drd_yalri fi2drd_yragi fi2drd_ybici fi2drd_ybnci _merge_fi2drd,
after(_merge_fi2d)
order fi2p_id fi2prd_rev_princ fi2prd_ysali fi2prd_ychoi fi2prd_yrsti
fi2prd_yalri fi2prd_yragi fi2prd_ybici fi2prd_ybnci _merge_fi2prd,
after(_merge_fi2p)
la var fi1d_id "Identifiant du déclarant 1, foyer 1
(id_fisc_foy_diff+ordrefip+type_fisc)"
la var fi1p_id "Identifiant du déclarant 2, foyer 1
(id_fisc_foy_diff+ordrefip+type_fisc)"
la var fi2d_id "Identifiant du déclarant 1, foyer 2
(id_fisc_foy_diff+ordrefip+type_fisc)"
la var fi2p_id "Identifiant du déclarant 2, foyer 2
(id_fisc_foy_diff+ordrefip+type_fisc)"
la var _merge_fi1drd "Appariement avec les revenus détaillés du déclarant 1,
foyer 1"
la var _merge_fi1prd "Appariement avec les revenus détaillés du déclarant 2,
foyer 1"
la var _merge_fi2drd "Appariement avec les revenus détaillés du déclarant 1,
foyer 2"
la var _merge_fi2prd "Appariement avec les revenus détaillés du déclarant 2,
foyer 2"
save "$wd\be2019.dta",replace
```

Variables fiscales de logement

On apparie ensuite les variables du logement fiscal des foyers auxquels est rattaché l'enfant EDP :

Ces variables seront préfixées par fl1_ et fl2_ selon qu'il s'agit du logement du foyer 1 ou du foyer 2. Si le logement est identique, alors les variables préfixées fl1_ et fl2_ seront identiques.

```
*Importer les variables de logement (exécuter toutes les lignes d'un coup) :
set more off
use "$wd\be2019.dta",clear
preserve
tempfile fl
forvalues i=2011/2019 {
    tempfile fl`i'
    use edp_be2019_fisc_logement_`i'.dta,clear
    ren an_fisc fi_an_fisc
    destring id_fisc_log_diff,replace
    ren id_fisc_log_diff fi1_id_fisc_log_diff
    clonevar fi2_id_fisc_log_diff=fi1_id_fisc_log_diff
    save `fl`i'',replace
}
use `fl2011',clear
append using `fl2012' `fl2013' `fl2014' `fl2015' `fl2016' `fl2017' `fl2018'
`fl2019'
save `fl',replace
restore
merge m:1 fi1_id_fisc_log_diff fi_an_fisc using `fl', keep(1 3)
gen(_merge_fl1)
foreach v of varlist dep cne seccad datachev hlmsem dateacte datepers occ
dnatlc natloc tax logement nloc4 eau egout elec gaz chauff ascenseur nbniv
surftot nbpiec nbpp nbsdb nbbaig nbdouc nbwc nbcuis8 nbcuis9 nbch nbsam
lgt_social num_iris num_qpv {
    ren `v' fl1_`v'
}
merge m:1 fi2_id_fisc_log_diff fi_an_fisc using `fl', keep(1 3)
gen(_merge_fl2)
foreach v of varlist dep cne seccad datachev hlmsem dateacte datepers occ
dnatlc natloc tax logement nloc4 eau egout elec gaz chauff ascenseur nbniv
surftot nbpiec nbpp nbsdb nbbaig nbdouc nbwc nbcuis8 nbcuis9 nbch nbsam
lgt_social num_iris num_qpv {
    ren `v' fl2_`v'
}
la var _merge_fl1 "Appariement avec la table fiscale logement 1"
la var _merge_fl2 "Appariement avec la table fiscale logement 2"
order fl?_* _merge_fl? , after(_merge_fi2prd)
destring fl?_*,replace
compress
save "$wd\be2019.dta",replace
```

Appariement aux enquêtes annuelles de recensement (EAR)

On utilise deux méthodes concurrentes pour apparier aux données de recensement. La première se veut robuste aux différences de logement : il s'agit de s'assurer que le logement recensé (celui disponible dans les tables EAR) est le même que le logement fiscal (celui disponible dans les tables fiscales). La deuxième méthode ne s'impose pas cette contrainte : il en résulte qu'elle permet plus d'appariements possibles.

Méthode robuste

Si l'enfant est déclaré fiscalement dans un logement qui est différent de celui où il est recensé, il faut pouvoir identifier cette différence. Pour cela, on considère qu'un logement fiscal est différent d'un logement recensé lorsque la *somme des années de naissance* des membres du logement diffère selon qu'il s'agit du logement recensé ou fiscal, et on encode dans la variable *hash* cette somme.

Dans cette méthode, par soucis d'économie de taille, on ne récupère que l'identifiant logement des tables EAR (variable *id_log_diff*) que l'on apparie avec *id_diff* et *hash* à chacun des logements fiscaux (identifiés par les variables *fi1_id_fisc_log_fiff* et *fi2_id_fisc_log_diff*). On préfixera cet identifiant logement par « earfi1_ » et « earfi2_ » selon que le logement recensé correspond au premier logement fiscal ou au deuxième.

```
*On sauvegarde la base EDP dans le répertoire de travail :
save "${wd}\be2019.dta",replace

/*Recherche de correspondance entre id_log_diff et id_diff+id_fisc_log_diff
On utilisera la somme des années de naissance du logement recensé et du
logement fiscal de l'individu EDP pour apparier le logement fiscal au logement
recensé :*/
set more off
tempfile ear
forvalues i=2011/2019 {
    tempfile app`i'
    use edp_be2019_ear`i'_individu.dta,clear
    bys id_diff id_log_diff: gen int n=_n if !mi(id_diff,id_log_diff)
    drop if n>1 & n<. //retirer les bulletins individuels excédentaires
    drop n
    bys id_log_diff: egen long hash=total(anai) //somme des années de
naissance du logement recensé
    contract id_diff id_log_diff hash
    drop if mi(id_diff)
    drop _freq
    save `ear',replace

    use edp_be2019_fisc_individu_`i'.dta,clear
    bys id_diff id_fisc_log_diff: gen int n=_n if
!mi(id_diff,id_fisc_log_diff)
    drop if n>1 & n<. //retirer les déclarations fiscales exécentaires
    drop n
    destring anais,replace
```

```
    bys id_fisc_log_diff: egen long hash=total(anais) //somme des années de
naissance du logement fiscal
    contract id_diff id_fisc_log_diff hash
    drop if mi(id_diff,id_fisc_log_diff)
    drop _freq
    merge m:m id_diff hash using `ear', keep(1 3) keepusing(id_log_diff)
    duplicates drop id_diff id_fisc_log_diff,force
    save `app`i''
}
use `app2011',clear
append using `app2012' `app2013' `app2014' `app2015' `app2016' `app2017'
`app2018' `app2019',gen(_app_fi)
drop if mi(id_log_diff)
drop _merge
la var hash "Somme des années de naissance du logement fiscal recensé"
ren id_fisc_log_diff fi1_id_fisc_log_diff
clonevar fi2_id_fisc_log_diff=fi1_id_fisc_log_diff
destring fi?_id_fisc_log_diff,replace
replace _app_fi=_app_fi+1 //uniformise les valeurs prises par _app_fi
compress
save "$wd\correspondance_earfi.dta",replace /*Cette table de correspondance ne
fait pas de distinction entre adultes EDP et enfants EDP
Appariement de cette table avec la base EDP :*/
use "${wd}\be2019.dta",clear
merge 1:1 id_diff fi1_id_fisc_log_diff _app_fi using
"$wd\correspondance_earfi.dta",keep(1 3) keepusing(hash id_log_diff)
gen(_merge_earfi1)
ren hash earfi1_hash
ren id_log_diff earfi1_id_log_diff
merge 1:1 id_diff fi2_id_fisc_log_diff _app_fi using
"$wd\correspondance_earfi.dta",keep(1 3) keepusing(hash
id_log_diff)  gen(_merge_earfi2)
ren hash earfi2_hash
ren id_log_diff earfi2_id_log_diff
la var _merge_earfi1 "Appariement du foyer 1 avec la table des EAR"
la var _merge_earfi2 "Appariement du foyer 2 avec la table des EAR"
```

Méthode simple

On fait un autre appariement avec les EAR, cette fois directement entre l'enfant EDP et son logement recensé, qui peut être différent du logement fiscal. Le nombre de logements appariés sera nécessairement plus grand que dans la procédure précédente, car précédemment, on se contraignait à ce qu'il y ait eu un appariement avec les DSF, et à ce que les logements fiscal et recensé soient identiques, pas ici.

Ici, on se contentera de ne sélectionner qu'un seul logement recensé, en évinçant les doublons de recensement. On ne récupère que les variables identifiant le logement recensé (*id_log_diff*) et le code commune du logement recensé (*depcom_code*), préfixés par « ear_ ».

```
gen int _merge_ear=.
tempfile temp
forvalues i=2011/2019 {
    preserve
    use edp_be2019_ear`i'_individu.dta,clear
    duplicates drop id_diff,force
    gen int annee=`i'
    save `temp',replace
    restore
    merge m:1 id_diff annee using `temp',keep(1 3 4) update
keepusing(id_log_diff depcom_code)
    replace _merge_ear=_merge*10+`i'-2010 if _merge_ear<20 | _merge_ear==.
    drop _merge
}
replace _merge_ear=10 if _merge_ear<20
ren id_log_diff ear_id_log_diff
ren depcom_code ear_depcom_code
order ear_id_log_diff ear_depcom_code _merge_ear,after(_merge_earfi2)
la var _merge_ear "Appariement de l'enfant EDP avec la table des EAR"
```

Comparaison des méthodes et perspectives d'évolution du codage

On peut dénombrer les différences entre les clés d'identification des logements recensés selon les deux méthodes, et comparer les possibilités d'appariements ouvertes par chacune d'elles :

```
1. *comptage des différences entre les deux manières d'apparier aux EAR :
2. count if (earfi1_id_log_diff!=ear_id_log_diff &
   !mi(ear_id_log_diff,earfi1_id_log_diff))|(earfi2_id_log_diff!=ear_id_log
   _diff & !mi(ear_id_log_diff,earfi2_id_log_diff))
3. count if (earfi1_id_log_diff==ear_id_log_diff &
   !mi(ear_id_log_diff,earfi1_id_log_diff))|(earfi2_id_log_diff==ear_id_log
   _diff & !mi(ear_id_log_diff,earfi2_id_log_diff))
4. ta _merge_ear _merge_earfi1
```

Les lignes 2 et 3 permettent de constater 2 314 occurrences de différences entre les logements fiscal et recensé ; pour 555 703 similarités, soit 0.41% de différences. L'exécution de la ligne 4 donne un aperçu des appariements qu'il est possible de réaliser avec l'une ou l'autre des méthodes :

Tableau 6 - Appariements aux EAR selon les 2 méthodes

		Appariement aux EAR (méthode robuste)		
		Pas d'appariement	Appariements	**Total**
Appariement aux EAR (méthode simple)	Pas d'appariement	7 730 522	0	**7 730 522**
	Appariement vague 1	33 212	51 665	**84 877**
	Appariement vague 2	35 023	53 904	**88 927**
	Appariement vague 3	37 762	57 824	**95 586**
	Appariement vague 4	39 569	59 530	**99 099**
	Appariement vague 5	41 223	62 072	**103 295**
	Appariement vague 6	42 564	63 136	**105 700**

	Appariement vague 7	42 002	65 834	**107 836**
	Appariement vague 8	44 278	67 598	**111 876**
	Appariement vague 9	46 406	70 763	**117 169**
	Total	**8 092 561**	**552 326**	**8 644 887**

On observe 552 326 appariements du premier logement fiscal dans le recensement, grâce à la méthode robuste. Toutefois, l'appariement aux EAR avec la méthode simple est nettement plus fructueux : l'ensemble des appariements sur les 9 vagues se cumule à 914 365 individus*années (8644887-7730522).

On regrette que la clé *hash* ne permette pas de trouver autant de correspondances qu'espéré, ce qui laisse supposer que la somme des années de naissance des membres du logement était une contrainte trop forte. Par la suite, on pourra assouplir ce code pour n'utiliser que la commune de résidence (*depcom*) comme clé de correspondance.

Injection de variables d'EAR (2011-2019) dans la base

Ici nous donnons un exemple de comment peut être utilisée la clé de la méthode simple (*ear_id_log_diff*) pour obtenir d'autres variables de l'EAR.

Pour chaque personne de référence du ménage (lprm=1) ou son conjoint (lprm=2), on va récupérer :

- L'identifiant de bulletin individuel de recemensement (*id_bi_diff*)
- Le genre de la personne (*sexe*)
- La date de naissance de la personne (*jnai, mnai, anai*)
- Le lien de la personne avec la personne de référence du ménage (*lprm*)
- Le nombre de personnes à charge dans le logement de la personne (*tlprm3,* variable construite)
- La situation de vie en couple ou non de la personne (*couple*)
- Le statut conjugal de la personne (*matr*)
- Le mode de cohabitation de la personne (*moco*)
- Le diplôme (*dipl*)
- La catégorie sociale (*cs*)

On commence par compresser et sauvegarder la base EDP :

```
*Destring, compression et sauvegarde:
set more off
destring *,replace
replace  fi1p_sexe="" if fi1p_sexe=="C"
replace  fi2p_sexe="" if fi2p_sexe=="C"
destring fi1p_sexe fi2p_sexe,replace
gen byte genre=1 if sexe=="M"
replace genre=2 if sexe=="F"
la var genre "Genre de l'individu EDP 1=garçon ; 2=fille"
compress
save "${wd}\be2019.dta",replace
```

Dans une première étape, on constitue une base *ear.dta* contenant toutes ces informations. Les variables seront préfixées par ear1_ et ear2_ selon qu'il s'agit de la personne de référence du ménage (ear1_) ou son conjoint (ear2_) :

```
//Compiler les infos côté EAR :
set more off
tempfile master using
forvalues i=2011/2019 {
    tempfile app`i'
    use "${basesdir}\edp_be2019_ear`i'_individu.dta",clear
    gen int annee=`i'
    gen byte lprm3=(lprm=="3") if lprm!=""
    bys id_log_diff: egen int tlprm3=total(lprm3) //total de PAC
    drop lprm3
    keep if inlist(lprm,"1","2")
    sort lprm
    gen ddn=mdy(mnai,jnai,anai)
    format ddn %td
    keep id_log_diff id_bi_diff annee sexe ddn couple matr *lprm* poids*
depcom_code moco dipl cs
    sort id_log_diff lprm
    ren (id_bi_diff sexe ddn couple matr *lprm* moco dipl cs)(ear1_id_bi_diff
ear1_sexe ear1_ddn ear1_couple ear1_matr ear1_*lprm* ear1_moco ear1_dipl
ear1_cs)
    ren poids* ear1_poids*
    preserve
    bys id_log_diff: keep if _n==1
    save `master',replace
    restore
    ren (ear1_id_bi_diff ear1_sexe ear1_ddn ear1_couple ear1_matr ear1_*lprm*
ear1_moco ear1_dipl ear1_cs)(ear2_id_bi_diff ear2_sexe ear2_ddn ear2_couple
ear2_matr ear2_*lprm* ear2_moco ear2_dipl ear2_cs)
    ren ear1_poids* ear2_poids*
    bys id_log_diff: keep if _n==2
    save `using',replace
    use `master',clear
    merge 1:1 id_log_diff using `using',nogenerate
    save `app`i'',replace
}
use `app2011',clear
append using `app2012' `app2013' `app2014' `app2015' `app2016' `app2017'
`app2018' `app2019'
ren id_log_diff ear_id_log_diff
ren depcom_code ear_depcom_code
destring *,replace
compress
save "$wd\ear.dta",replace
```

Dans la deuxième étape, nous rouvrons la base EDP, et nous injectons les nouvelles variables EAR contenues dans *ear.dta*. Etant donné que la clé d'appariement est le logement recensé (*ear_id_log_diff*), c'est-à-dire, celui issu de l'appariement par méthode simple, il peut être différent du logement fiscal. Pour repérer un peu mieux les éventuelles non-correspondances entre logement fiscal et logement recensé, on n'utilisera pas cette fois la clé *hash*, mais on se propose de vérifier si les communes du logement fiscal et celles du logement recensé sont les mêmes :

```
1. *Injecter l'EAR dans l'EDP :
2. use "$wd\be2019.dta",clear
3. //on vérifie d'abord que le logement recensé est le même que le logement
   fiscal, en comparant les codes de commune :
4. gen str6 fi1_depcom=fi1_csdep+strofreal(fi1_cne1,"%03.0f") if
   fi1_csdep!="B3" & fi1_cne1<.
5. gen str6 fi2_depcom=fi2_csdep+strofreal(fi2_cne1,"%03.0f") if
   fi2_csdep!="B3" & fi2_cne1<.
6. destring ear_id_log_diff,replace
7. cap n assert (ear_depcom_code==fi1_depcom | ear_depcom_code==fi2_depcom)
   if !mi(fi1_depcom,ear_depcom_code)
8. //On injecte les données EAR dans la base EDP :
9. merge m:1 ear_id_log_diff ear_depcom_code annee using
   "$wd\ear.dta",keepusing(ear*) nogenerate keep(1 3)
10.//On filtre les logements EAR qui ne sont pas identiques à un logement
   fiscal connu :
11.gen byte ear_samelogfisc=(ear_depcom_code==fi1_depcom |
   ear_depcom_code==fi2_depcom) if !mi(fi1_depcom,ear_depcom_code)
12.label define ear_samelogfisc 1 "Le logement recensé est dans la même
   commune qu'un logement fiscal" 0 "Pas de commune identique entre
   logements fiscaux et recensé"
13.label values ear_samelogfisc ear_samelogfisc
14.la var ear_samelogfisc "Logement recensé identique au logement fiscal"
```

L'exécution de la ligne 7, et un *tabulate* sur la variable *ear_samelogfisc*, permettent de conclure qu'il y a 47 877 contradictions entre logement fiscal et recensé, sur 734 986 observations, soit 6.51% de contradictions.

Enfin, on sauvegarde à nouveau la base :

```
compress
save "${wd}\be2019.dta",replace
```

Récupération des informations les plus fraîches des EAR

Dans la sous-section précédente, nous avons montré comment on pouvait récupérer des informations de l'EAR une année fiscale donnée. Il est certain toutefois qu'un individu EDP ne puisse être recensé sur toutes les années fiscales où il est observé (2011-2019), laissant des « trous » d'observation entre les années. Par ailleurs, il est possible qu'il ait été recensé les années précédant l'observation fiscale (2004-2010). On se propose ici de remplir les « trous » des informations les plus fraîches disponibles. Nous illustrons d'un exemple le changement attendu :

Tableau 7 - Exemple d'information EAR disponible

id_diff	annee	age_t	ear1_id_bi_diff	ear1_lprm	ear1_sexe	ear1_ddn	ear1_couple	ear2_id_bi_diff	ear2_lprm	ear2_sexe	ear2_ddn	ear2_couple
0002056882	2011	6	.	.	.	.	.	.	.	.	.	.
0002056882	2012	7	.	.	.	.	.	.	.	.	.	.
0002056882	2013	8	848010	1	1	31Aug1979	1	847708	2	2	17Oct1979	1
0002056882	2014	9	.	.	.	.	.	.	.	.	.	.
0002056882	2015	10	.	.	.	.	.	.	.	.	.	.
0002056882	2016	11	.	.	.	.	.	.	.	.	.	.
0002056882	2017	12	.	.	.	.	.	.	.	.	.	.
0002056882	2018	13	818954	1	1	31Aug1979	1	818955	2	2	17Oct1979	1
0002056882	2019	14	.	.	.	.	.	.	.	.	.	.

Dans ce tableau, l'individu EDP apparaît comme vivant avec ses deux parents en couple, de sexe différent. Le logement a été recensé 2 fois sur la période, à 5 ans d'intervalle. On propose de créer une nouvelle série de variables, de même nom, mais suffixées par *_r*, pour extrapoler l'information manquante à partir de l'information la plus fraîche. C'est-à-dire de répliquer les valeurs obtenues à l'EAR 2013 pour les années 2014 à 2017, et celles obtenues à l'EAR 2018 pour l'année 2019. Pour les années 2011 et 2012, on ira chercher dans les EAR 2004-2010 les valeurs du recensement le plus récent. La variable *_merge_ear_r* permet d'informer l'origine de la vague EAR. On produira ainsi le résultat suivant :

Tableau 8- Exemple d'informations EAR rafraîchies

id_diff	annee	age_t	ear1_id_bi_diff_r	ear1_lprm_r	ear1_sexe_r	ear1_ddn_r	ear1_couple_r	ear2_id_bi_diff_r	ear2_lprm_r	ear2_sexe_r	ear2_ddn_r	ear2_couple_r	_merge_ear_r
0002056882	2011	6	791934	1	1	31Aug1979	1	791934	2	2	17Oct1979	1	Rafraîchi depuis l'EAR 2008
0002056882	2012	7	791934	1	1	31Aug1979	1	791934	2	2	17Oct1979	1	Rafraîchi depuis l'EAR 2008
0002056882	2013	8	848010	1	1	31Aug1979	1	847708	2	2	17Oct1979	1	Apparié à l'EAR 2013
0002056882	2014	9	848010	1	1	31Aug1979	1	847708	2	2	17Oct1979	1	Rafraîchi depuis l'EAR 2013
0002056882	2015	10	848010	1	1	31Aug1979	1	847708	2	2	17Oct1979	1	Rafraîchi depuis l'EAR 2013
0002056882	2016	11	848010	1	1	31Aug1979	1	847708	2	2	17Oct1979	1	Rafraîchi depuis l'EAR 2013
0002056882	2017	12	848010	1	1	31Aug1979	1	847708	2	2	17Oct1979	1	Rafraîchi depuis l'EAR 2013
0002056882	2018	13	818954	1	1	31Aug1979	1	818955	2	2	17Oct1979	1	Apparié à l'EAR 2018
0002056882	2019	14	818954	1	1	31Aug1979	1	818955	2	2	17Oct1979	1	Rafraîchi depuis l'EAR 2018

Le codage nécessaire à la production de ce résultat est le suivant :

```
*Rafraîchissement des EAR:
//récupération des identifiants logements des EAR 2004-2010 pour l'année 2011:
gen int _merge_ear_r=. //récupération des identifiants logements
tempfile temp
forvalues i=2010(-1)2004 {
    preserve
    use edp_be2019_ear`i'_individu.dta,clear
    duplicates drop id_diff,force
    gen int annee=2011
    save `temp',replace
    restore
    merge m:1 id_diff annee using `temp',keep(1 3 4 5) update
keepusing(id_log_diff depcom_code)
    replace _merge_ear_r=_merge*100+(`i'-2000) if _merge_ear_r<200 |
_merge_ear_r==.
    drop _merge
}
replace _merge_ear_r=100 if _merge_ear_r<200
ren id_log_diff ear_id_log_diff_r
ren depcom_code ear_depcom_code_r
destring ear_id_log_diff_r ear_depcom_code_r,replace
la var _merge_ear_r "Appariement de l'enfant EDP avec la table des EAR"
//récupération des informations des EAR 2004-2010
set more off
tempfile master using be2019 ear
forvalues i=2010(-1)2004 {
    save `be2019',replace
    use "${basesdir}\edp_be2019_ear`i'_individu.dta",clear
    gen int annee=2011
    gen byte lprm3=(lprm=="3") if lprm!=""
    bys id_log_diff: egen int tlprm3=total(lprm3) //total de PAC
    drop lprm3
    keep if inlist(lprm,"1","2")
    sort lprm
    gen ddn=mdy(mnai,jnai,anai)
    format ddn %td
    keep id_log_diff id_bi_diff annee sexe ddn couple matr *lprm* poids*
depcom_code moco dipl cs
    sort id_log_diff lprm
    ren (id_bi_diff sexe ddn couple matr *lprm* moco dipl
cs)(ear1_id_bi_diff_r ear1_sexe_r ear1_ddn_r ear1_couple_r ear1_matr_r
ear1_*lprm*_r ear1_moco_r ear1_dipl_r ear1_cs_r)
    ren poids* ear1_poids*_r
    preserve
    bys id_log_diff: keep if _n==1
    save `master',replace
    restore
    ren (ear1_id_bi_diff_r ear1_sexe_r ear1_ddn_r ear1_couple_r ear1_matr_r
ear1_*lprm*_r ear1_moco_r ear1_dipl_r ear1_cs_r)(ear2_id_bi_diff_r ear2_sexe_r
```

```
ear2_ddn_r ear2_couple_r ear2_matr_r ear2_*lprm*_r ear2_moco_r ear2_dipl_r
ear2_cs_r)
    ren ear1_poids* ear2_poids*
    bys id_log_diff: keep if _n==2
    save `using',replace
    use `master',clear
    merge 1:1 id_log_diff using `using',nogenerate
    ren id_log_diff ear_id_log_diff_r
    ren depcom_code ear_depcom_code_r
    save `ear',replace
    use `be2019',clear
    merge m:1 ear_id_log_diff_r ear_depcom_code_r annee using
`ear',keepusing(ear*) nogenerate keep(1 3 4 5) update
}
//extrapolation des vagues EAR manquantes:
replace ear1_id_bi_diff_r=subinstr(ear1_id_bi_diff_r,"X",".",1) //recode the
leading "X" in id_bi_diff with a "." in order to be numeric
replace ear2_id_bi_diff_r=subinstr(ear1_id_bi_diff_r,"X",".",1)
destring ear?_*_r,replace
replace ear1_cs="" if ear1_cs=="ZZ" //hors champs
replace ear2_cs="" if ear2_cs=="ZZ"
destring ear1_cs ear2_cs,replace
sort idi annee
foreach v of varlist ear?_id_bi_diff ear?_couple ear?_cs ear?_dipl ear?_lprm
ear?_matr ear?_moco ear?_poids_ea ear?_sexe ear?_poids_ea_cale
ear?_poids_ea_cale_oct ear?_poids_panel_5 ear?_tlprm3 ear?_ddn {
    replace `v'_r=`v' if !mi(`v')
    replace `v'_r=L.`v'_r if `v'_r==.
}
//codage de la variable _merge_ear_r
replace _merge_ear_r=L._merge_ear_r if L._merge_ear_r>100 & L._merge_ear_r<.
replace _merge_ear_r=311 if !mi(ear1_id_bi_diff) & annee==2011
replace _merge_ear_r=312 if !mi(ear1_id_bi_diff) & annee==2012
replace _merge_ear_r=111 if mi(ear1_id_bi_diff) & !mi(L.ear1_id_bi_diff) &
annee==2012
replace _merge_ear_r=313 if !mi(ear1_id_bi_diff) & annee==2013
replace _merge_ear_r=111 if mi(ear1_id_bi_diff) & !mi(L2.ear1_id_bi_diff) &
annee==2013
replace _merge_ear_r=112 if mi(ear1_id_bi_diff) & !mi(L.ear1_id_bi_diff) &
annee==2013
replace _merge_ear_r=314 if !mi(ear1_id_bi_diff) & annee==2014
replace _merge_ear_r=111 if mi(ear1_id_bi_diff) & !mi(L3.ear1_id_bi_diff) &
annee==2014
replace _merge_ear_r=112 if mi(ear1_id_bi_diff) & !mi(L2.ear1_id_bi_diff) &
annee==2014
replace _merge_ear_r=113 if mi(ear1_id_bi_diff) & !mi(L.ear1_id_bi_diff) &
annee==2014
replace _merge_ear_r=315 if !mi(ear1_id_bi_diff) & annee==2015
```

```
replace _merge_ear_r=111 if mi(ear1_id_bi_diff) & !mi(L4.ear1_id_bi_diff) &
annee==2015
replace _merge_ear_r=112 if mi(ear1_id_bi_diff) & !mi(L3.ear1_id_bi_diff) &
annee==2015
replace _merge_ear_r=113 if mi(ear1_id_bi_diff) & !mi(L2.ear1_id_bi_diff) &
annee==2015
replace _merge_ear_r=114 if mi(ear1_id_bi_diff) & !mi(L.ear1_id_bi_diff) &
annee==2015
replace _merge_ear_r=316 if !mi(ear1_id_bi_diff) & annee==2016
replace _merge_ear_r=111 if mi(ear1_id_bi_diff) & !mi(L5.ear1_id_bi_diff) &
annee==2016
replace _merge_ear_r=112 if mi(ear1_id_bi_diff) & !mi(L4.ear1_id_bi_diff) &
annee==2016
replace _merge_ear_r=113 if mi(ear1_id_bi_diff) & !mi(L3.ear1_id_bi_diff) &
annee==2016
replace _merge_ear_r=114 if mi(ear1_id_bi_diff) & !mi(L2.ear1_id_bi_diff) &
annee==2016
replace _merge_ear_r=115 if mi(ear1_id_bi_diff) & !mi(L.ear1_id_bi_diff) &
annee==2016
replace _merge_ear_r=317 if !mi(ear1_id_bi_diff) & annee==2017
replace _merge_ear_r=111 if mi(ear1_id_bi_diff) & !mi(L6.ear1_id_bi_diff) &
annee==2017
replace _merge_ear_r=112 if mi(ear1_id_bi_diff) & !mi(L5.ear1_id_bi_diff) &
annee==2017
replace _merge_ear_r=113 if mi(ear1_id_bi_diff) & !mi(L4.ear1_id_bi_diff) &
annee==2017
replace _merge_ear_r=114 if mi(ear1_id_bi_diff) & !mi(L3.ear1_id_bi_diff) &
annee==2017
replace _merge_ear_r=115 if mi(ear1_id_bi_diff) & !mi(L2.ear1_id_bi_diff) &
annee==2017
replace _merge_ear_r=116 if mi(ear1_id_bi_diff) & !mi(L.ear1_id_bi_diff) &
annee==2017
replace _merge_ear_r=318 if !mi(ear1_id_bi_diff) & annee==2018
replace _merge_ear_r=111 if mi(ear1_id_bi_diff) & !mi(L7.ear1_id_bi_diff) &
annee==2018
replace _merge_ear_r=112 if mi(ear1_id_bi_diff) & !mi(L6.ear1_id_bi_diff) &
annee==2018
replace _merge_ear_r=113 if mi(ear1_id_bi_diff) & !mi(L5.ear1_id_bi_diff) &
annee==2018
replace _merge_ear_r=114 if mi(ear1_id_bi_diff) & !mi(L4.ear1_id_bi_diff) &
annee==2018
replace _merge_ear_r=115 if mi(ear1_id_bi_diff) & !mi(L3.ear1_id_bi_diff) &
annee==2018
replace _merge_ear_r=116 if mi(ear1_id_bi_diff) & !mi(L2.ear1_id_bi_diff) &
annee==2018
replace _merge_ear_r=117 if mi(ear1_id_bi_diff) & !mi(L.ear1_id_bi_diff) &
annee==2018
replace _merge_ear_r=319 if !mi(ear1_id_bi_diff) & annee==2019
```

```
replace _merge_ear_r=111 if mi(ear1_id_bi_diff) & !mi(L8.ear1_id_bi_diff) &
annee==2019
replace _merge_ear_r=112 if mi(ear1_id_bi_diff) & !mi(L7.ear1_id_bi_diff) &
annee==2019
replace _merge_ear_r=113 if mi(ear1_id_bi_diff) & !mi(L6.ear1_id_bi_diff) &
annee==2019
replace _merge_ear_r=114 if mi(ear1_id_bi_diff) & !mi(L5.ear1_id_bi_diff) &
annee==2019
replace _merge_ear_r=115 if mi(ear1_id_bi_diff) & !mi(L4.ear1_id_bi_diff) &
annee==2019
replace _merge_ear_r=116 if mi(ear1_id_bi_diff) & !mi(L3.ear1_id_bi_diff) &
annee==2019
replace _merge_ear_r=117 if mi(ear1_id_bi_diff) & !mi(L2.ear1_id_bi_diff) &
annee==2019
replace _merge_ear_r=118 if mi(ear1_id_bi_diff) & !mi(L.ear1_id_bi_diff) &
annee==2019
label define _merge_ear_r 100 "Non-apparié aux EAR" 111 "Rafraîchi depuis
l'EAR 2011" 112 "Rafraîchi depuis l'EAR 2012" 113 "Rafraîchi depuis l'EAR
2013" 114 "Rafraîchi depuis l'EAR 2014" 115 "Rafraîchi depuis l'EAR 2015" 116
"Rafraîchi depuis l'EAR 2016" 117 "Rafraîchi depuis l'EAR 2017" 118 "Rafraîchi
depuis l'EAR 2018" 310 "Rafraîchi depuis l'EAR 2010" 311 "Apparié à l'EAR
2011" 312 "Apparié à l'EAR 2012" 313 "Apparié à l'EAR 2013" 314 "Apparié à
l'EAR 2014" 315 "Apparié à l'EAR 2015" 316 "Apparié à l'EAR 2016" 317 "Apparié
à l'EAR 2017" 318 "Apparié à l'EAR 2018" 319 "Apparié à l'EAR 2019" 404
"Rafraîchi depuis l'EAR 2004" 405 "Rafraîchi depuis l'EAR 2005" 406 "Rafraîchi
depuis l'EAR 2006" 407 "Rafraîchi depuis l'EAR 2007" 408 "Rafraîchi depuis
l'EAR 2008" 409 "Rafraîchi depuis l'EAR 2009" ,replace
label values _merge_ear_r _merge_ear_r
save "${wd}\be2019.dta",replace
```

Collecte des bulletins de naissance dans les bases de l'INSEE

Pour les années 1998 à 2003, nous pouvons récupérer dans les bases exhaustives de naissances de l'INSEE les informations de naissance de l'enfant EDP qui nous manquent sur les mois de naissance de janvier, avril, et juillet. À partir de 2004, les enfants nés l'un des 16 jours EDP font systématiquement l'objet d'une collecte des informations de naissance dans la table « naissance ». Avant 1998, il manque dans les bases exhaustives de l'INSEE des clés d'appariement essentielles (dates et lieux de naissance des parents) pour apparier l'enfant.

Pour les cohortes nées entre 1998 et 2003 incluses, on va chercher dans la table « naissance » les enfants EDP n'ayant pas d'acte de naissance dans la base EDP, mais qui ont le même genre, naissent le même jour dans la même commune, et on va vérifier si :

- parmi toutes les femmes déclarantes dans l'un ou l'autre des foyers fiscaux de l'enfant, une femme née le même jour dans la même commune apparaît dans le bulletin de naissance. Si c'est le cas c'est la mère
- parmi tous les hommmes déclarant dans l'un ou l'autre des foyers fiscaux de l'enfant, un homme né le même jour dans la même commune apparaît dans le bulletin de naissance. Si c'est le cas c'est le père

```
*Collecte des bulletins de naissance INSEE :
forvalues i=1998/2003 {
 tempfile insee`i'
 use nais`i'.dta,clear
 gen ddnstring=anais+"-"+mnais+"-"+jnais if !mi(anais,mnais,jnais)
 gen ddn=date(ddnstring,"YMD")
 format ddn %td
 keep if (inlist(month(ddn),4,7,10) & day(ddn)<5) | (month(ddn)==1 &
inlist(day(ddn),2,3,4,5)) //on ne garde que les naissances EDP
 keep sexe anais mnais jnais ddn depnais comnais anaism mnaism jnaism
depnaism comnaism anaisp mnaisp jnaisp depnaisp comnaisp catacc
 save `insee`i''
}
use `insee1998',clear
append using `insee1999' `insee2000' `insee2001' `insee2002' `insee2003'
gen ddnmstring=anaism+"-"+mnaism+"-"+jnaism if !mi(anaism,mnaism,jnaism)
gen ddnm=date(ddnmstring,"YMD")
format ddnm %td
gen ddnpstring=anaisp+"-"+mnaisp+"-"+jnaisp if !mi(anaisp,mnaisp,jnaisp)
gen ddnp=date(ddnpstring,"YMD")
format ddnp %td
drop ddnmstring ddnpstring
drop if mi(ddnm,ddnp)
gen nai_lieu=depnais+comnais if !mi(depnais,comnais)
gen depcomm=depnaism+comnaism if !mi(depnaism,comnaism)
gen depcomp=depnaisp+comnaisp if !mi(depnaisp,comnaisp)
destring *,replace
```

```
26. duplicates tag ddn sexe nai_lieu ddnm depcomm ,gen(tagm) //doublons
    enfant-mère
27. fre catacc if tagm>0 //351 naissances simples (X) sont issues d'une même
    paire enfant-mère doublonnée. On élimine toutes ces naissances simples,
    et on élimine les 2139 gemellités excédentaires :
28. drop if catacc=="X" & tagm>0
29. duplicates drop ddn sexe nai_lieu ddnm depcomm,force //élimination des
    gemellités excédentaires
30. duplicates tag ddn sexe nai_lieu ddnp depcomp ,gen(tagp) //doublons
    enfant-père
31. fre catacc if tagp>0 //757 naissances simples (X) sont issues d'une même
    paire enfant-père doublonnée. On élimine toutes ces naissances simples,
    ainsi que les 21 autres cas :
32. drop if tagp>0
33. drop tag?
34. ren sexe genre
35. compress
36. save "$wd\insee.dta",replace
```

L'exécution des lignes 27 et 31 fait apparaître le tableau suivant (colonne de gauche pour les paires enfant-mère ; colonne de droite pour les paires enfant-père). Considérant qu'on ne pourrait apparier correctement à l'EDP les doublons qui ne correspondent pas à des gemellités (« naissances simples »), on élimine ces cas (351 observations). Afin de pouvoir travailler avec une base INSEE complètement dédoublonnée, on élimine également les observations excédentaires correspondant à des naissances multiples (2139 observations). On élimine également tous les cas de doublons enfants-pères (778 observations).

Tableau 9- Catégorie d'accouchement chez les paires enfant-mère et enfant-père doublonnées

	Effectifs D'enfants-Mères	**EFFECTifs d'enfants-pères**
X – Naissance simple	351	757
0 - 2 garçons vivants	2006	7
1 - 1 garçon vivant, 1 garçon mort	44	0
2 - 2 garçons morts	18	0
3 – 1 garçon vivant, 1 fille vivante	1	6
4 – 1 garçon vivant, 1 fille morte	0	0
5 – 1 garçon mort, 1 fille vivante	0	0
6 – 1 garçon mort, 1 fille morte	0	0
7 – 2 filles vivantes	1996	6
8 – 1 fille vivante, 1 fille morte	22	0
9 – 2 filles mortes	20	0
A – 3 garçons vivants ou morts	31	2
B – 2 garçons + 1 fille vivants ou morts	28	0
C – 1 garçon + 2 filles vivants ou morts	34	0
D – 3 filles vivantes ou mortes	45	0
E – 4 garçons vivants ou morts	4	0
F – 3 garçons + 1 fille vivants ou morts	0	0
G – 2 garçons + 2 filles vivants ou morts	2	0
H – 1 garçon + 3 filles vivants ou morts	0	0

I – 4 filles vivantes ou mortes	0	0
J - + de 4 enfants vivant ou morts	0	0
Total	4602	778

Depuis la base INSEE, on prépare les clés d'appariement de la mère et du père, pour qu'elles aient le même nom que les variables correspondantes dans l'EDP :

```
/*Préparation des clés d'appariement : mère*/
use "$wd\insee.dta",clear
clonevar fi1d_ddn=ddnm
clonevar fi1p_ddn=ddnm
clonevar fi2d_ddn=ddnm
clonevar fi2p_ddn=ddnm
clonevar fi1d_codnais=depcomm
clonevar fi1p_codnais=depcomm
clonevar fi2d_codnais=depcomm
clonevar fi2p_codnais=depcomm
gen byte fi1d_sexe=2
gen byte fi1p_sexe=2
gen byte fi2d_sexe=2
gen byte fi2p_sexe=2
save "${wd}\inseem.dta",replace
/*Préparation des clés d'appariement : père*/
use "$wd\insee.dta",clear
clonevar fi1d_ddn=ddnp
clonevar fi1p_ddn=ddnp
clonevar fi2d_ddn=ddnp
clonevar fi2p_ddn=ddnp
clonevar fi1d_codnais=depcomp
clonevar fi1p_codnais=depcomp
clonevar fi2d_codnais=depcomp
clonevar fi2p_codnais=depcomp
gen byte fi1d_sexe=1
gen byte fi1p_sexe=1
gen byte fi2d_sexe=1
gen byte fi2p_sexe=1
save "${wd}\inseep.dta",replace
```

On procède ensuite à l'appariement de la base EDP avec la base INSEE selon la procédure décrite plus haut. On récupère du côté de l'INSEE les variables d'intérêt : les dates et lieux de naissance du père et de la mère (*ddnm ddnp depcomm depcomp*).

```
/*Appariement des mères:*/
use  "${wd}\be2019.dta",clear
merge m:1 ddn genre nai_lieu fi1d_ddn fi1d_codnais fi1d_sexe using
"${wd}\inseem.dta", keep(1 3 4 5) gen(_merge_fi1dmere) update keepusing(ddnm)
merge m:1 ddn genre nai_lieu fi1p_ddn fi1p_codnais fi1p_sexe using
"${wd}\inseem.dta", keep(1 3 4 5) gen(_merge_fi1pmere) update keepusing(ddnm)
```

```
merge m:1 ddn genre nai_lieu fi2d_ddn fi2d_codnais fi2d_sexe using
"${wd}\inseem.dta", keep(1 3 4 5) gen(_merge_fi2dmere) update keepusing(ddnm)
merge m:1 ddn genre nai_lieu fi2p_ddn fi2p_codnais fi2p_sexe using
"${wd}\inseem.dta", keep(1 3 4 5) gen(_merge_fi2pmere) update keepusing(ddnm)
/*Appariement des pères:*/
merge m:1 ddn genre nai_lieu fi1d_ddn fi1d_codnais fi1d_sexe using
"${wd}\inseep.dta", keep(1 3 4 5) gen(_merge_fi1dpere) update keepusing(ddnp)
merge m:1 ddn genre nai_lieu fi1p_ddn fi1p_codnais fi1p_sexe using
"${wd}\inseep.dta", keep(1 3 4 5) gen(_merge_fi1ppere) update keepusing(ddnp)
merge m:1 ddn genre nai_lieu fi2d_ddn fi2d_codnais fi2d_sexe using
"${wd}\inseep.dta", keep(1 3 4 5) gen(_merge_fi2dpere) update keepusing(ddnp)
merge m:1 ddn genre nai_lieu fi2p_ddn fi2p_codnais fi2p_sexe using
"${wd}\inseep.dta", keep(1 3 4 5) gen(_merge_fi2ppere) update keepusing(ddnp)
/*Codage de variables reconstruites:*/
bys id_diff: egen ddnmm=max(ddnm) //on fixe, par enfant, les dates de
naissance des parents
bys id_diff: egen ddnpm=max(ddnp)
replace ddnm=ddnmm
replace ddnp=ddnpm
drop ddnmm ddnpm
replace anaism=year(ddnm) if mi(anaism)
replace anaisp=year(ddnp) if mi(anaisp)
gen str5 depcomm=mere_lieu_nais_depcom
replace depcomm=fi1d_codnais if _merge_fi1dmere==4
replace depcomm=fi1p_codnais if _merge_fi1pmere==4
replace depcomm=fi2d_codnais if _merge_fi2dmere==4
replace depcomm=fi2p_codnais if _merge_fi2pmere==4
gen str5 depcomp=pere_lieu_nais_depcom
replace depcomp=fi1d_codnais if _merge_fi1dpere==4
replace depcomp=fi1p_codnais if _merge_fi1ppere==4
replace depcomp=fi2d_codnais if _merge_fi2dpere==4
replace depcomp=fi2p_codnais if _merge_fi2ppere==4
sort id_diff depcomm //on fixe, par enfant, les lieux de naissance des parents
by id_diff: gen str5 depcommm=depcomm[_N]
sort id_diff depcomp
by id_diff: gen str5 depcompm=depcomp[_N]
replace depcomm=depcommm
replace depcomp=depcompm
drop depcommm depcompm
gen byte _insee_update=1 if _merge_fi1dmere==4
replace _insee_update=1 if _merge_fi1pmere==4
replace _insee_update=1 if _merge_fi2dmere==4
replace _insee_update=1 if _merge_fi2pmere==4
replace _insee_update=0 if _insee_update==.
bys idi: egen byte maxupdate=max(_insee_update) //on fixe, par enfant, la
variable _insee_update
replace _insee_update=maxupdate
drop maxupdate
order genre,after(sexe)
```

```
order depcomm ,after(mere_lieu_nais_depcom)
order depcomp ,after(pere_lieu_nais_depcom)
la var depcomm "Département de naissance de la mère"
la var depcomp "Département de naissance du père"
order _merge_fi1dmere _merge_fi1dpere,after(_merge_fi1d)
order _merge_fi1pmere _merge_fi1ppere,after(_merge_fi1p)
order _merge_fi2dmere _merge_fi2dpere,after(_merge_fi2d)
order _merge_fi2pmere _merge_fi2ppere,after(_merge_fi2p)
la var _merge_fi1dmere "Le déclarant 1 du foyer 1 est la mère (3 ou 4 si vrai
; 1 nsp ou non)"
la var _merge_fi1dpere "Le déclarant 1 du foyer 1 est le père (3 ou 4 si vrai
; 1 nsp ou non)"
la var _merge_fi1pmere "Le déclarant 2 du foyer 1 est la mère (3 ou 4 si vrai
; 1 nsp ou non)"
la var _merge_fi1ppere "Le déclarant 2 du foyer 1 est le père (3 ou 4 si vrai
; 1 nsp ou non)"
la var _merge_fi2dmere "Le déclarant 1 du foyer 2 est la mère (3 ou 4 si vrai
; 1 nsp ou non)"
la var _merge_fi2dpere "Le déclarant 1 du foyer 2 est le père (3 ou 4 si vrai
; 1 nsp ou non)"
la var _merge_fi2pmere "Le déclarant 2 du foyer 2 est la mère (3 ou 4 si vrai
; 1 nsp ou non)"
la var _merge_fi2ppere "Le déclarant 2 du foyer 2 est le père (3 ou 4 si vrai
; 1 nsp ou non)"
la var _insee_update "Appariement sur la table Naissances de l'INSEE"
```

Correction des erreurs de sexe des déclarants

Pour identifier les familles homoparentales, ou pour récupérer les informations fiscales des parents de naissance, une correction des erreurs de sexe des déclarants fiscaux sera nécessaire. On utilise deux méthodes : la première consiste à corriger les erreurs de sexe des déclarants fiscaux lorsqu'il est évident qu'il s'agit des parents de naissance en raison de leur date de naissance concordante (méthode transversale) ; la deuxième consiste à corriger les erreurs de sexe en tenant compte des variations dans le temps des identificatrices de sexe (méthode longitudinale). On utilise ces deux méthodes successivement.

Correction transversale

Tout d'abord on crée des dummies pour déceler si les déclarants fiscaux de chaque foyer de l'enfant sont de même sexe ou non, dans le but ultérieur de dénombrer les corrections qui ont pu être réalisées. Ces variables sont préfixées par « fi1_ » et « fi2_ » selon le foyer considéré, ont un radical « mmsx » (même sexe), et sont suffixées par « _a » (avant correction des erreurs de sexe). Une fois les corrections transversales réalisées, on crée à nouveau des dummies similaires, cette fois suffixées par « _b » (après correction transversale).

```
1. /*On corrige en premier les erreurs "transversales" <- incohérences
   entre le sexe du déclarant fiscal et celui du parent de naissance ayant
   la même date de naissance. On fait confiance au sexe renseigné à l'état
   civil.*/
2. gen byte fi1_mmsx_a=(fi1d_sexe==fi1p_sexe) if !mi(fi1d_sexe)
3. gen byte fi2_mmsx_a=(fi2d_sexe==fi2p_sexe) if !mi(fi2d_sexe)
4. gen byte fi_mmsx_a=max(fi1_mmsx_a,fi2_mmsx_a)
5. fre fi_mmsx_a //4032 enfants*années
6. replace fi1d_sexe=2 if fi1d_ddn==ddnm & !mi(ddnm) & ddnm!=ddnp
7. replace fi2d_sexe=2 if fi2d_ddn==ddnm & !mi(ddnm) & ddnm!=ddnp
8. replace fi1p_sexe=2 if fi1p_ddn==ddnm & !mi(ddnm) & ddnm!=ddnp
9. replace fi2p_sexe=2 if fi2p_ddn==ddnm & !mi(ddnm) & ddnm!=ddnp
10.replace fi1d_sexe=1 if fi1d_ddn==ddnp & !mi(ddnp) & ddnm!=ddnp
11.replace fi2d_sexe=1 if fi2d_ddn==ddnp & !mi(ddnp) & ddnm!=ddnp
12.replace fi1p_sexe=1 if fi1p_ddn==ddnp & !mi(ddnp) & ddnm!=ddnp
13.replace fi2p_sexe=1 if fi2p_ddn==ddnp & !mi(ddnp) & ddnm!=ddnp
14.gen byte fi1_mmsx_b=(fi1d_sexe==fi1p_sexe) if !mi(fi1d_sexe)
15.gen byte fi2_mmsx_b=(fi2d_sexe==fi2p_sexe) if !mi(fi2d_sexe)
16.gen byte fi_mmsx_b=max(fi1_mmsx_b,fi2_mmsx_b)
17.fre fi_mmsx_b //2652 enfants*années
18.ta fi_mmsx_a fi_mmsx_b
```

L'exécution de la ligne 18 fait apparaître le tableau suivant :

Tableau 10 - Genres des couples parentaux avant et après correction transversale

		Après correction des erreurs transversales de sexe		
		De sexe différent	De même sexe	Total
Avant correction des erreurs transversales de sexe	De sexe différent	5 125 500	338	5 125 838
	De même sexe	1 718	2 314	4 032
	Total	5 127 218	2 652	5 129 870

On note que la correction des erreurs de sexe fait disparaître un tiers des couples de même sexe. Plus précisément, ce sont 1 718 enfants*années dont les parents sont identifiés de même sexe qui sont en réalité des couples de sexe différent, et 338 qui avaient des parents incorrectement identifiés de sexe différents qui sont en réalité de même sexe.

Correction longitudinale

On corrige ensuite les erreurs "longitudinales" : les incohérences d'une année sur l'autre du sexe du déclarant fiscal, en supposant qu'il n'y ait pas d'interversion d'une année sur l'autre entre les déclarants 1 et 2. Si des incohérences sont constatées, on passe toutes les valeurs de sexe en valeurs manquantes. On supposera qu'il n'y a pas de non-conformité de genre chez les déclarants fiscaux qui justifient ces variations de sexe. À l'issue de cette correction, on crée des dummies de déclarants de même sexe, suffixées cette fois « _c » (après correction longitudinale).

```
sort idi annee
set more off
forvalues i=1/8 {
    di "i=`i'"
    replace fi1d_sexe=.v if fi1d_sexe!=L`i'.fi1d_sexe &
fi1d_ddn==L`i'.fi1d_ddn & !mi(fi1d_sexe,fi1d_ddn,L`i'.fi1d_sexe)
    replace fi1d_sexe=.v if F`i'.fi1d_sexe==.v & !mi(fi1d_sexe)
    replace fi1p_sexe=.v if fi1p_sexe!=L`i'.fi1p_sexe &
fi1p_ddn==L`i'.fi1p_ddn & !mi(fi1p_sexe,fi1p_ddn,L`i'.fi1p_sexe)
    replace fi1p_sexe=.v if F`i'.fi1p_sexe==.v & !mi(fi1p_sexe)
    replace fi2d_sexe=.v if fi2d_sexe!=L`i'.fi2d_sexe &
fi2d_ddn==L`i'.fi2d_ddn & !mi(fi2d_sexe,fi2d_ddn,L`i'.fi2d_sexe)
    replace fi2d_sexe=.v if F`i'.fi2d_sexe==.v & !mi(fi2d_sexe)
    replace fi2p_sexe=.v if fi2p_sexe!=L`i'.fi2p_sexe &
fi2p_ddn==L`i'.fi2p_ddn & !mi(fi2p_sexe,fi2p_ddn,L`i'.fi2p_sexe)
    replace fi2p_sexe=.v if F`i'.fi2p_sexe==.v & !mi(fi2p_sexe)
}
gen byte fi1_mmsx_c=(fi1d_sexe==fi1p_sexe) if !mi(fi1d_sexe,fi1p_sexe)
gen byte fi2_mmsx_c=(fi2d_sexe==fi2p_sexe) if !mi(fi2d_sexe,fi2p_sexe)
gen byte fi_mmsx_c=max(fi1_mmsx_c,fi2_mmsx_c)
fre fi_mmsx_c //2594 enfants*années
ta fi_mmsx_b fi_mmsx_c,m
drop fi1_mmsx_a fi2_mmsx_a fi1_mmsx_b fi2_mmsx_b fi1_mmsx_c fi2_mmsx_c
order fi_mmsx_a fi_mmsx_b fi_mmsx_c,after(_merge_fi)
```

```
la var fi_mmsx_a "Effectifs initiaux d'enfants vivant avec des co-déclarants de même sexe"
la var fi_mmsx_b "Effectifs après correction transversales des erreurs de sexe"
la var fi_mmsx_c "Effectifs après correction longitudinale des erreurs de sexe"
compress
save "${wd}\be2019.dta",replace
```

L'exécution du *tabulate* fait apparaître le tableau suivant :

Tableau 11 - Genres des couples parentaux avant et après correction longitudinale

		Après suppression des erreurs longitudinales de sexe			
		De sexe différent	De même sexe	Valeur manquante	Total
Avant suppression des erreurs longitudinales de sexe	De sexe différent	3 393 699	0	1 733 419	5 127 218
	De même sexe	0	2 594	58	2 652
	Total	3 393 699	2 594	5 248 494	8 644 887

On note assez peu d'incohérences : seuls 58 enfants*années sont concernés par des incohérences temporelles de la mention de sexe de leurs parents.

Appariement des informations fiscales des parents de naissance

Dans cette étape, on va identifier qui, dans le foyer et le logement de l'enfant, est son parent de naissance. Dans la plupart des cas, on s'attend à ce que les co-déclarants fiscaux (variables préfixées fi1d_ fi1p_ fi2d_ et fi2p_) soient les parents de naissance de l'enfant. Toutefois, il arrive parfois que ce ne soit pas le cas. Soit que le déclarant fiscal soit le grand-père par exemple, ou que le père soit le déclarant mais que la mère soit une adulte non-déclarante déclarée à charge dans le foyer, voire déclarée sur un autre foyer fiscal que l'enfant, mais dans le même logement. Ou, dans la plupart des cas, parce que la co-déclarante ou le co-déclarant n'est pas la mère ou le père de naissance, mais la belle-mère ou le beau-père.

Pour cela, l'appariement se fera sur :

- La date de naissance des parents de naissance : on comparera celles-ci, telles que données sur le bulletin de naissance, aux dates de naissance des personnes présentes sur les tables fiscales individuelles
- Le genre des parents de naissance : là encore, on comparera le genre de tous les habitants du foyer/logement de l'enfant, avec ceux de parents de naissance (mère ou père)
- Les identifiants de foyer et de logement fiscaux :
    - Dans un premier temps, on appariera par le foyer fiscal seulement
    - Dans un second temps, si l'appariement est infructueux, on appariera uniquement sur le logement fiscal.

Nous apparions en premier les informations sur les mères de naissance. Une fois intégrées à notre base EDP, ces nouvelles variables concernant la mère de naissance seront préfixées par « fim_ » (pour fisc individu mother) :

```
//Match mother:
set more off
gen byte _merge_fim=.
la var _merge_fim "Code #ijkl: Vague i=[1;9]; Foyer j={1;2}; Merge
k={1;3;4;5}; LogUpdate={0;1}"
gen byte fim_liveswithchild=.
tempfile findmother_foy findmother_log
forvalues i=2011/2019 {
    di "`i'"
    local i2=`i'-2010
    preserve
    use edp_be2019_fisc_individu_`i'.dta,clear
    ren an_fisc fi_an_fisc
    format fi_an_fisc %ty
    keep if sexe=="2" //on ne garde que les femmes (clé d'appariement par
genre)
    gen ddnstring=anais+"-"+mnais+"-"+jnais if !mi(anais,mnais,jnais)
    gen ddnm=date(ddnstring,"YMD") //on prépare la clé d'appariement date de
naissance
    format ddnm %td
    egen fi1_idfoy=concat(id_fisc_foy_diff ordrefip) //on prépare la clé
d'appariement pour les foyers 1 et 2
```

```
    clonevar fi2_idfoy=fi1_idfoy
    clonevar fim_idfoy=fi1_idfoy //le foyer de la mère sera utile pour les
appariements par logement
    clonevar fim_id_fisc_log_diff=id_fisc_log_diff //le logement de la mère
sera utile pour les appariements par foyer
    drop if mi(ddnm,id_fisc_log_diff,fi1_idfoy) //on supprime les lignes où il
est impossible d'apparier
    ren (id_diff cideci lien_familial zoxyzd dacoed type_fisc t_charge codnais
i_fisc_* type_decl type_pres poids_fideli nbptr)(fim_id_diff fim_cideci
fim_lien_familial fim_zoxyzd fim_dacoed fim_type_fisc fim_t_charge fim_codnais
fim_i_fisc_* fim_type_decl fim_type_pres fim_poids_fideli fim_nbptr) //on
récupérera ces variables là
    ren id_fisc_log_diff fi1_id_fisc_log_diff //on prépare la clé
d'appariement pour les logements 1 et 2
    clonevar fi2_id_fisc_log_diff=fi1_id_fisc_log_diff
    destring fi?_id_fisc_log_diff,replace
    duplicates drop fi1_idfoy ddnm,force //on ne veut qu'une obs. par clé
    save `findmother_foy',replace
    duplicates drop fi1_id_fisc_log_diff ddnm,force //idem
    save `findmother_log',replace
    restore
    merge m:1 fi1_idfoy ddnm fi_an_fisc using `findmother_foy',keep(1 3 4 5)
keepusing(fim_*) update
    replace _merge_fim=`i2'*1000+100+_merge*10 if _app_fi==`i2'
    drop _merge
    merge m:1 fi2_idfoy ddnm fi_an_fisc using `findmother_foy',keep(1 3 4 5)
keepusing(fim_*) update
    replace _merge_fim=`i2'*1000+200+_merge*10 if _app_fi==`i2' & _merge>=3
    drop _merge
    merge m:1 fi1_id_fisc_log_diff ddnm fi_an_fisc using
`findmother_log',keep(1 3 4 5) keepusing(fim_*) update
    replace _merge_fim=`i2'*1000+100+_merge*10+1 if _app_fi==`i2' & _merge>=3
    drop _merge
    merge m:1 fi2_id_fisc_log_diff ddnm fi_an_fisc using
`findmother_log',keep(1 3 4 5) keepusing(fim_*) update
    replace _merge_fim=`i2'*1000+200+_merge*10+1 if _app_fi==`i2' & _merge>=3
    drop _merge
    replace fim_liveswithchild=((mod(_merge_fim,100)-mod(_merge_fim,10))>10)
if _app_fi==`i2'
    replace fim_liveswithchild=.m if (_merge_fi==1 | ddnm>=.) & _app_fi==`i2'
    replace fim_liveswithchild=.m if mi(fi1_id_fisc_log_diff) &
mi(fi2_id_fisc_log_diff) & _app_fi==`i2'
}
order _merge_fim,after(fim_id_fisc_log_diff)
```

Nous apparions ensuite les informations sur les pères de naissance. Les nouvelles variables seront préfixées par « fif_ » (pour fisc individu father) :

```
//Match father:
set more off
gen byte _merge_fif=.
la var _merge_fif "Code #ijkl: Vague i=[1;9]; Foyer j={1;2}; Merge
k={1;3;4;5}; LogUpdate={0;1}"
gen byte fif_liveswithchild=.
tempfile findfather_foy findfather_log
forvalues i=2011/2019 {
    di "`i'"
    local i2=`i'-2010
    preserve
    use edp_be2019_fisc_individu_`i'.dta,clear
    ren an_fisc fi_an_fisc
    format fi_an_fisc %ty
    keep if sexe=="1"
    gen ddnstring=anais+"-"+mnais+"-"+jnais if !mi(anais,mnais,jnais)
    gen ddnp=date(ddnstring,"YMD")
    format ddnp %td
    egen fi1_idfoy=concat(id_fisc_foy_diff ordrefip)
    clonevar fi2_idfoy=fi1_idfoy
    clonevar fif_idfoy=fi1_idfoy
    clonevar fif_id_fisc_log_diff=id_fisc_log_diff
    drop if mi(ddnp,id_fisc_log_diff,fi1_idfoy)
    ren (id_diff cideci lien_familial  zoxyzd dacoed type_fisc t_charge
codnais i_fisc_* type_decl type_pres poids_fideli nbptr)(fif_id_diff
fif_cideci fif_lien_familial fif_zoxyzd fif_dacoed fif_type_fisc fif_t_charge
fif_codnais fif_i_fisc_* fif_type_decl fif_type_pres fif_poids_fideli
fif_nbptr)
    ren id_fisc_log_diff fi1_id_fisc_log_diff
    clonevar fi2_id_fisc_log_diff=fi1_id_fisc_log_diff
    destring fi?_id_fisc_log_diff,replace
    duplicates drop fi1_idfoy ddnp,force
    save `findfather_foy',replace
    duplicates drop fi1_id_fisc_log_diff ddnp,force
    save `findfather_log',replace
    restore
    merge m:1 fi1_idfoy ddnp fi_an_fisc using `findfather_foy',keep(1 3 4 5)
keepusing(fif_*) update
    replace _merge_fif=`i2'*1000+100+_merge*10 if _app_fi==`i2'
    drop _merge
    merge m:1 fi2_idfoy ddnp fi_an_fisc using `findfather_foy',keep(1 3 4 5)
keepusing(fif_*) update
    replace _merge_fif=`i2'*1000+200+_merge*10 if _app_fi==`i2' & _merge>=3
    drop _merge
    merge m:1 fi1_id_fisc_log_diff ddnp fi_an_fisc using
`findfather_log',keep(1 3 4 5) keepusing(fif_*) update
    replace _merge_fif=`i2'*1000+100+_merge*10+1 if _app_fi==`i2' & _merge>=3
    drop _merge
```

```
    merge m:1 fi2_id_fisc_log_diff ddnp fi_an_fisc using
`findfather_log',keep(1 3 4 5) keepusing(fif_*) update
    replace _merge_fif=`i2'*1000+200+_merge*10+1 if _app_fi==`i2' & _merge>=3
    drop _merge
    replace fif_liveswithchild=((mod(_merge_fif,100)-mod(_merge_fif,10))>10)
if _app_fi==`i2'
    replace fif_liveswithchild=.m if (_merge_fi==1 | ddnp>=.) & _app_fi==`i2'
    replace fif_liveswithchild=.m if mi(fi1_id_fisc_log_diff) &
mi(fi2_id_fisc_log_diff) & _app_fi==`i2'
}
order _merge_fif,after(fif_id_fisc_log_diff)
la var fim_liveswithchild "L'enfant vit avec sa mère de naissance"
la var fif_liveswithchild "L'enfant vit avec son père de naissance"
```

Les variables construites _merge_fim et _merge_fif permettent de savoir si l'appariement a eu lieu, elles ont le format {ijkl} où i est la vague fiscale (de 1 à 9), j est le numéro du foyer fiscal (le 1er ou le 2e), k est la valeur prise par _merge (c'est à dire 1 s'il n'y a pas eu d'appariement ; et 3, 4 ou 5 s'il y a eu un appariement), et l est une indicatrice pour savoir si l'appariement s'est réalisé sur le logement (1) ou sur le foyer fiscal (0).

On créée également les variables fim_liveswithchild et fif_liveswithchild pour identifier si la mère de naissance et le père de naissance vivent ou non avec l'enfant. Pour cela, on vérifie simplement si l'appariement a eu lieu, c'est-à-dire, si _merge_fim et _merge_fif ont une valeur k=3, 4 ou 5 (appariement), ou 1 (pas d'appariement). Plus difficiles à interpréter sont les valeurs « .m » de ces variables : elles signifient qu'on ne peut pas conclure sur la possibilité du parent de naissance de vivre ou non avec l'enfant. Cela arrive dans l'un ou plusieurs des cas suivants :

- L'enfant n'est pas apparié aux données socio-fiscales
- L'enfant n'a pas de logement fiscal renseigné
- La date de naissance du parent de naissance n'est pas connue.

On compresse la base pour économiser de l'espace et on sauvegarde à nouveau :

```
*Compression et sauvegarde
compress
save "${wd}\be2019.dta",replace
```

Différences d'âge et de sexe entre l'enfant et tous les autres habitants du logement

Dans cette section, on veut connaître, pour chaque identifiant fiscal d'enfant EDP et de vague fiscale, l'ensemble des différences d'âge que cet enfant EDP a avec les autres individus de son logement fiscal. Ainsi qu'établir une base des sexes des autres individus.

Dans un premier temps, on ouvre chacune des tables fiscales individuelles, et on ne conserve que les logements qui ont au plus 6 déclarants fiscaux, ce qui représente environ 99% des cas. Une base des différences d'âge et de sexe serait en effet trop grosse dans les cas où un logement compte un grand nombre de foyers fiscaux (et donc de déclarants fiscaux) différents.

Ensuite on créée un produit cartésien entre tous les individus de chaque logement, de façon à pouvoir comparer les dates de naissance et les sexes de tous ces individus. L'idée est de se retrouver avec une base dont voici un extrait :

Tableau 12 - Exemple d'un produit cartésien réalisé sur la table fiscale individuelle

Id_diff	Id_fisc_log_diff	Anais	Declarant	Ddn	Genre	Id	Id1	Genre1	Diffage1	Id2	Genre2	Diffage2
	1108706	1976	1	08aug1976	1	1384987A1	1384987AA	.	27	1384987A2	2	-8
	1108706	1968	1	26dec1968	2	1384987A2	1384987A1	1	8	1384987AA	.	35
0001192524	1108706	2003	0	.	.	1384987AA	1384987A1	1	-27	1384987A2	2	-35

Il s'agit ici d'un logement de 3 individus, dont 1 enfant EDP né en 2003. Les date de naissance exactes et le genre ne sont habituellement pas renseignés dans les tables fiscales pour les enfants. Les variables relatives aux *autres* individus du logement sont suffixées par 1 (le premier autre individu, en orange dans le tableau) et 2 (le deuxième autre individu, en vert dans le tableau). Comme il peut y avoir plus de 3 individus par logement, les numéros de suffixes peuvent s'étendre bien au-delà de 2. Dans notre base, étant donné les conditions que nous mettons (maximum de 6 déclarants fiscaux dans le logement), on peut observer jusqu'à 32 autres individus dans un même logement (en plus de l'enfant EDP). Dans le tableau, on observe grâce aux variables *diffage1* et *diffage2* que l'enfant a 27 ans de moins que le premier autre individu du logement (vraisemblablement son père), et 35 ans de moins que le deuxième autre individu du logement (probablement sa mère).

Une fois cette base obtenue, on ne conservera que les lignes correspondant à des enfants EDP (année de naissance d'au moins 1994, présence d'un identifiant EDP), qui sera ensuite appariée à la base EDP en cours de construction, par l'identifiant *id_diff*. Dans le Tableau 12 ci-dessus, cela correspondra à la 3^{e} ligne.

On vérifiera au préalable si les enfants EDP appartiennent à la même génération ou à une génération plus jeune que les autres individus, en fixant un critère de **différence d'âge de 15 ans**. Dans notre exemple, cela consistera à calculer la valeur 2 pour la variable *nascgen* et 0 pour la variable *nsamegen*, c'est-à-dire que l'enfant EDP vit avec 2 personnes de génération plus ancienne, et 0 personne de la même génération. On construit également les variables *ypa* et *opa* dans les logements où il n'y a que deux personnes de génération plus ancienne que l'enfant EDP, qui permettent de déterminer l'âge du plus jeune et l'âge du plus âgé, respectivement.

On s'intéresse également au genre des autres individus du logement : on nommera les variables *nmascb* et *nfascb* pour, respectivement, le nombre d'hommes, et de femmes, de génération plus ancienne que l'enfant EDP, après correction des erreurs de sexe.

```
set more off
ssc install gsort
tempfile nsup2
forvalues i=2011/2019 {
    tempfile year`i'
    use edp_be2019_fisc_individu_`i'.dta,clear
    gen byte declarant=inlist(type_fisc,"01","02","1","2")
    bys id_fisc_log_diff: egen int ndeclarants=total(declarant)
    fre ndeclarants
    keep if ndeclarants<=6 //on garde 99% des cas
    bys id_fisc_log_diff: gen int n=_n
    bys id_fisc_log_diff: gen int capn=_N
    gen ddnstring=anais+"-"+mnais+"-"+jnais if !mi(anais,mnais,jnais)
    gen ddn=date(ddnstring,"YMD")
    format ddn %td
    drop ddnstring
    destring anais,replace
    gsort + id_fisc_log_diff - anais - ddn + id_diff //nécessite le package
gsort
    by id_fisc_log_diff: gen rank=_n //on créée une variable équivalant à 1
lorsque la personne est la + jeune du logement, 2 lorsqu'elle est la 2e +
jeune, etc.
    gen byte genre=1 if sexe=="1"
    replace genre=2 if sexe=="2"
    save `year`i'',replace
    keep if capn>=2
    ren anais anais2
    ren genre genre2
    ren ddn ddn2
    egen id2=concat(id_fisc_foy_diff ordrefip type_fisc)
    ren declarant declarant2
    ren rank rank2
    keep id2 id_fisc_log_diff anais2 genre2 ddn2 id_diff declarant2 rank2
    save `nsup2',replace
    ren anais2 anais1
    ren genre2 genre1
    ren ddn2 ddn1
    ren id2 id1
    ren declarant2 declarant1
    ren rank2 rank1
    joinby id_fisc_log_diff using `nsup2' //produit cartésien des individus à
l'intérieur du logement
    gen long diffage=anais2-anais1
    drop if id1==id2
    bys id_fisc_log_diff id1: gen int m=_n
    drop anais2
```

```
    reshape wide diffage id2 declarant2 genre2 ddn2 rank2,i(id_diff
id_fisc_log_diff anais1 genre1 id1 declarant1 ddn1 rank1) j(m) //on reforme la
base pour gagner en horizontalité
    ren genre1 genre
    ren ddn1 ddn
    ren (id1 id2* declarant1 declarant2* genre2* ddn2* rank1 rank2*) (id id*
declarant declarant* genre* ddn* rank rank*)
    foreach v of varlist diffage* {
        replace `v'=. if abs(`v')>120 //on retire les différences d'âge de
plus de 120 ans
    }
    keep if !mi(id_diff) & anais1>=1994 & anais1<. //on ne garde que les
enfants EDP de notre base /!\ il peut y avoir pls foyers par enfant edp
    ren anais1 anais
    save `nsup2',replace
    use `year`i'',clear
    keep if capn==1
    egen id=concat(id_fisc_foy_diff ordrefip type_fisc) //future clé
d'appariement avec la base EDP
    keep if !mi(id_diff) & anais>=1994 & anais<.
    keep id_diff id anais genre ddn declarant id_fisc_log_diff
    append using `nsup2',gen(_nsup2)
    save `year`i'',replace
}
use `year2011',clear
append using `year2012' `year2013' `year2014' `year2015' `year2016' `year2017'
`year2018' `year2019',gen(_wave)
local i=0
foreach v of varlist diffage* { //on considère que 2 individus séparés de 15
ans d'âge ou plus appartiennent à deux générations distinctes
    local i=`i'+1
    gen byte samegen`i'=(abs(`v')<15) if `v'<.
    gen byte ascgen`i'=(`v'<=-15) if `v'<.
    gen byte descgen`i'=(`v'>=15) if `v'<. //n'ayant gardé que les enfants EDP
nés àp. de 1994, ces cas devraient etre rares
}
merge m:1 id_diff using edp_be2019_individu.dta,keep(1 3) nogenerate
keepusing(sexe) //récupération des sexes des enfants EDP
drop genre
gen byte genre=1 if sexe=="M"
replace genre=2 if sexe=="F"
gen byte _app_fi=_wave+1
merge m:1 id_diff _app_fi using "$wd\be2019.dta",keep(1 3) keepusing(ddnm
ddnp) nogenerate //on vient récupérer les dates de naissance des parents pour
pouvoir faire des corrections d'erreurs de sexe
drop _app_fi
local i=0
foreach v of varlist genre* {
    if "`v'"=="genre" continue
```

```
    local i=`i'+1
    *création des variables d'homme et femme de génération supérieure à
l'enfant EDP, avant correction des erreurs de sexe :
    gen byte masca`i'=(`v'==1)*ascgen`i' if !mi(`v',ascgen`i')
    gen byte fasca`i'=(`v'==2)*ascgen`i' if !mi(`v',ascgen`i')
    *correction des erreurs de sexe chez les individus de génération
supérieure à l'enfant EDP :
    replace `v'=1 if `v'==2 & ddn`i'==ddnp & ddnm!=ddnp & !mi(ddn`i',ddnp)
    replace `v'=2 if `v'==1 & ddn`i'==ddnm & ddnm!=ddnp & !mi(ddn`i',ddnm)
    *création des variables d'homme et femme de génération supérieure à
l'enfant EDP, après correction des erreurs de sexe
    gen byte mascb`i'=(`v'==1)*ascgen`i' if !mi(`v',ascgen`i')
    gen byte fascb`i'=(`v'==2)*ascgen`i' if !mi(`v',ascgen`i')
}
egen int nmasca=rowtotal(masca*)
egen int nmascb=rowtotal(mascb*)
egen int nfasca=rowtotal(fasca*)
egen int nfascb=rowtotal(fascb*)
egen int nsamegen=rowtotal(samegen*)
egen int nascgen=rowtotal(ascgen*)
egen int ndescgen=rowtotal(descgen*)
egen int capn=rownonmiss(samegen*)
gen int ndiffgen=capn-nsamegen if capn>0
egen int ndeclarants=rowtotal(declarant*)
gen int ypa=.
gen int opa=.
local i=0
foreach v of varlist diffage* { //on calcule l'âge du + jeune et du + âgé des
partenaires, uniquement dans les logements où il y a 2 personnes de génération
+ ancienne que l'enfant EDP :
    local i=`i'+1
    replace ypa=min(ypa,2011+_wave-anais-`v') if (mascb`i'==1 | fascb`i'==1) &
nmascb+nfascb==2
    replace opa=max(opa,2011+_wave-anais-`v') if (mascb`i'==1 | fascb`i'==1) &
nmascb+nfascb==2
}
gen int mpa=(ypa+opa)/2
drop samegen* ascgen* descgen* masc* fasc* sexe
order _wave,first
la var _wave "Vague fiscale 0=2011 ; 8=2019"
la var id_fisc_log_diff "Identifiant de logement fiscal de l'enfant"
la var anais "Année de naissance de l'enfant"
la var ddn "Date de naissance de l'enfant"
la var id "Identifiant fiscal de l'enfant
(id_fisc_foy_diff+ordrefip+type_fisc)"
la var _nsup2 "Le logement compte au moins 2 individus"
la var id1 "Identifiant fiscal de l'individu 1"
la var diffage1 "Année de naissance d'id1 - année de naissance de l'enfant
EDP"
```

```
la var declarant1 "L'individu 1 est déclarant fiscal"
la var genre1 "Le genre de l'individu 1"
la var id_diff "Identifiant de l'enfant EDP"
la var nsamegen "Nombre d'individus de même génération que l'enfant EDP"
la var nascgen "Nombre d'individus de plus ancienne génération que l'enfant
EDP"
la var ndescgen "Nombre d'individus de plus récente génération que l'enfant
EDP"
la var capn "Nombre d'autres individus dans le logement"
la var ndiffgen "Nombre d'individus de génération différente de l'enfant EDP"
la var declarant "L'enfant EDP est déclarant fiscal"
la var genre "Genre de l'enfant EDP"
la var nmasca "Nombre d'hommes de génération + ancienne que l'enfant EDP -
avant correction"
la var nmascb "Nombre d'hommes de génération + ancienne que l'enfant EDP -
après correction"
la var nfasca "Nombre de femmes de génération + ancienne que l'enfant EDP -
avant correction"
la var nfascb "Nombre de femmes de génération + ancienne que l'enfant EDP -
après correction"
la var ndeclarants "Nombre d'autres individus déclarants dans le logement"
la var ypa "Âge du + jeune des 2 partenaires de génération+"
la var opa "Âge du + âgé des 2 partenaires de génération+"
la var mpa "Âge moyen des 2 partenaires de génération+"
order genre ddnm ddnp ndiffgen nsamegen nascgen nmasca nmascb nfasca nfascb
ndescgen ndeclarants capn ypa opa mpa,before(_nsup2)
compress
save "$wd\diffage.dta",replace
```

Une fois cette base des différences d'âge et de sexe créée et sauvegardée, on l'apparie à notre base EDP en construction :

```
use "${wd}\be2019.dta",clear
ren id idlong
egen id=concat(fi1_id_fisc_foy_diff fi1_ordrefip fi1_type_fisc)
gen byte _wave=_app_fi-1
merge 1:1 id_diff id _wave using "$wd\diffage.dta",keep(1 3) keepusing(_nsup2
nsamegen nascgen ndescgen ndeclarants capn ndiffgen nmasca nmascb nfasca
nfascb ypa opa mpa) gen(_merge_fi1_diffage)
ren (_nsup2 nsamegen nascgen ndescgen ndeclarants capn ndiffgen nmasca nmascb
nfasca nfascb ypa opa mpa) (fi1_nsup2 fi1_nsamegen fi1_nascgen fi1_ndescgen
fi1_ndeclarants fi1_capn fi1_ndiffgen fi1_nmasca fi1_nmascb fi1_nfasca
fi1_nfascb fi1_ypa fi1_opa fi1_mpa)
drop id
egen id=concat(fi2_id_fisc_foy_diff fi2_ordrefip fi2_type_fisc)
merge 1:1 id_diff id _wave using "$wd\diffage.dta",keep(1 3) keepusing(_nsup2
nsamegen nascgen ndescgen ndeclarants capn ndiffgen nmasca nmascb nfasca
nfascb ypa opa mpa) gen(_merge_fi2_diffage)
```

```
ren (_nsup2 nsamegen nascgen ndescgen ndeclarants capn ndiffgen nmasca nmascb
nfasca nfascb ypa opa mpa) (fi2_nsup2 fi2_nsamegen fi2_nascgen fi2_ndescgen
fi2_ndeclarants fi2_capn fi2_ndiffgen fi2_nmasca fi2_nmascb fi2_nfasca
fi2_nfascb fi2_ypa fi2_opa fi2_mpa)
drop id _wave
ren idlong id
order fi1_nsup2 fi1_nsamegen fi1_nascgen fi1_ndescgen fi1_ndeclarants fi1_capn
fi1_ndiffgen fi1_nmasca fi1_nmascb fi1_nfasca fi1_nfascb fi1_ypa fi1_opa
fi1_mpa _merge_fi1_diffage,before(fi1d_id_diff)
order fi2_nsup2 fi2_nsamegen fi2_nascgen fi2_ndescgen fi2_ndeclarants fi2_capn
fi2_ndiffgen fi2_nmasca fi2_nmascb fi2_nfasca fi2_nfascb fi2_ypa fi2_opa
fi2_mpa _merge_fi2_diffage,before(fi2d_id_diff)
la var _merge_fi1_diffage "Appariement de l'identifiant fiscal 1 avec la base
diffage"
la var _merge_fi2_diffage "Appariement de l'identifiant fiscal 2 avec la base
diffage"
save "$wd\be2019.dta",replace
```

On peut par ailleurs compléter les informations manquantes sur l'âge des parents plus jeune et plus âgé (*ypa* et *opa*), en s'appuyant sur les déclarations fiscales :

```
*Complétion des âges des + jeune et âgé des partenaires:
replace fi1_ypa=annee-year(max(fi1d_ddn,fi1p_ddn)) if !mi(fi1d_ddn,fi1p_ddn) &
mi(fi1_ypa)
replace fi2_ypa=annee-year(max(fi2d_ddn,fi2p_ddn)) if !mi(fi2d_ddn,fi2p_ddn) &
mi(fi2_ypa)
replace fi1_opa=annee-year(min(fi1d_ddn,fi1p_ddn)) if !mi(fi1d_ddn,fi1p_ddn) &
mi(fi1_opa)
replace fi2_opa=annee-year(min(fi2d_ddn,fi2p_ddn)) if !mi(fi2d_ddn,fi2p_ddn) &
mi(fi2_opa)
replace fi1_mpa=(fi1_ypa+fi1_opa)/2 if !mi(fi1_ypa,fi1_opa) & mi(fi1_mpa)
replace fi2_mpa=(fi2_ypa+fi2_opa)/2 if !mi(fi2_ypa,fi2_opa) & mi(fi2_mpa)
save "$wd\be2019.dta",replace
```

Situations familiales

Parentalité des co-déclarants fiscaux

Dans cette sous-section, on cherche à déterminer si les co-déclarants fiscaux de l'enfant sont les père et mère de naissance de celui-ci. Pour cela, on va s'appuyer sur les variables construites à la section « Appariement des informations fiscales des parents de naissance », *fim_liveswithchild* et *fif_liveswithchild*. Dans la pratique, il n'est pas tant nécessaire d'utiliser ces variables car elles produisent beaucoup de valeurs manquantes, une simple vérification que la date de naissance des co-déclarants fiscaux soit la même que celles des parents de naissance pourrait suffire.

```
*Construction de la parenté de chaque déclarant fiscal:
cap drop d1 d2 d3 d4
gen byte d1=1 if fim_liveswithchild==1 & fi1d_ddn==ddnm & fi1d_sexe==2 &
!mi(fim_liveswithchild,fi1d_ddn,ddnm,fi1d_sexe)
replace d1=2 if fif_liveswithchild==1 & fi1d_ddn==ddnp & fi1d_sexe==1 &
!mi(fif_liveswithchild,fi1d_ddn,ddnp,fi1d_sexe)
gen byte d2=1 if fim_liveswithchild==1 & fi1p_ddn==ddnm & fi1p_sexe==2 &
!mi(fim_liveswithchild,fi1p_ddn,ddnm,fi1p_sexe)
replace d2=2 if fif_liveswithchild==1 & fi1p_ddn==ddnp & fi1p_sexe==1 &
!mi(fif_liveswithchild,fi1p_ddn,ddnp,fi1p_sexe)
replace d2=. if d2==1 & d1==1 //cas impossibles où les co-déclarants sont la
même personne
replace d2=. if d2==2 & d1==2 //idem
gen byte d3=1 if fim_liveswithchild==1 & fi2d_ddn==ddnm & fi2d_sexe==2 &
!mi(fim_liveswithchild,fi2d_ddn,ddnm,fi2d_sexe)
replace d3=2 if fif_liveswithchild==1 & fi2d_ddn==ddnp & fi2d_sexe==1 &
!mi(fif_liveswithchild,fi2d_ddn,ddnp,fi2d_sexe)
gen byte d4=1 if fim_liveswithchild==1 & fi2p_ddn==ddnm & fi2p_sexe==2 &
!mi(fim_liveswithchild,fi2p_ddn,ddnm,fi2p_sexe)
replace d4=2 if fif_liveswithchild==1 & fi2p_ddn==ddnp & fi2p_sexe==1 &
!mi(fif_liveswithchild,fi2p_ddn,ddnp,fi2p_sexe)
replace d3=. if d3==1 & d4==1 //cas impossibles où les codéclarants sont la
même personne
replace d4=. if d3==2 & d4==2 //idem
replace d1=0 if fi1d_ddn!=ddnp & fi1d_ddn!=ddnm & !mi(fi1d_ddn,ddnm,ddnp)
replace d2=0 if fi1p_ddn!=ddnp & fi1p_ddn!=ddnm & !mi(fi1p_ddn,ddnm,ddnp)
replace d3=0 if fi2d_ddn!=ddnp & fi2d_ddn!=ddnm & !mi(fi2d_ddn,ddnm,ddnp)
replace d4=0 if fi2p_ddn!=ddnp & fi2p_ddn!=ddnm & !mi(fi2p_ddn,ddnm,ddnp)
label define decl 0 "Le déclarant n'est ni le père ni la mère" 1 "La
déclarante est la mère" 2 "Le déclarant est le père",replace
label values d1 d2 d3 d4 decl
la var d1 "Qui est le déclarant 1 du foyer 1?"
la var d2 "Qui est le déclarant 2 du foyer 1?"
la var d3 "Qui est le déclarant 1 du foyer 2?"
la var d4 "Qui est le déclarant 2 du foyer 2?"
```

Parentalité dans les foyers fiscaux

Par la suite, on utilise les variables construites d1, d2, d3 et d4 (parentalité des co-déclarants fiscaux dans les foyers 1 et 2), pour déterminer qui, de l'un l'autre ou les deux parents de naissance, déclare l'enfant dans le foyer 1 ou 2 :

```
cap drop f1 f2
gen byte f1=1 if (d1==1 | d2==1) & !(d1==2 | d2==2)
replace f1=2 if (d1==2 | d2==2) & !(d1==1 | d2==1)
replace f1=3 if ((d1==1 & d2==2) | (d1==2 & d2==1))
replace f1=0 if d1==0 & d2==0
gen byte f2=.m if mi(fi2_id_fisc_foy_diff) & _merge_fi>1
replace f2=1 if (d3==1 | d4==1) & !(d3==2 | d4==2)
replace f2=2 if (d3==2 | d4==2) & !(d3==1 | d4==1)
replace f2=3 if ((d3==1 & d4==2) | (d3==2 & d4==1))
replace f2=0 if d3==0 & d4==0
label define foyer 0 "Ni le père ni la mère" 1 "La mère" 2 "Le père" 3 "La
mère et le père" .m "Il n'y a pas de foyer 2",replace
label values f1 f2 foyer
la var f1 "Qui déclare le foyer 1 ?"
la var f2 "Qui déclare le foyer 2 ?"
```

Parentalité dans les logements fiscaux

On construit ensuite les variables *log1* et *log2* pour vérifier quels parents de naissance habitent le 1er logement fiscal et/ou le 2e, s'il y en a :

```
*Dans quel logement vivent les PDN ?
//la mère:
cap drop log1 log2
gen byte log1=1 if fim_id_fisc_log_diff==fi1_id_fisc_log_diff &
fif_id_fisc_log_diff!=fi1_id_fisc_log_diff & !mi(fi1_id_fisc_log_diff)
replace log1=2 if fif_id_fisc_log_diff==fi1_id_fisc_log_diff &
fim_id_fisc_log_diff!=fi1_id_fisc_log_diff & !mi(fi1_id_fisc_log_diff)
replace log1=3 if fif_id_fisc_log_diff==fi1_id_fisc_log_diff &
fim_id_fisc_log_diff==fi1_id_fisc_log_diff & !mi(fi1_id_fisc_log_diff)
replace log1=0 if fif_id_fisc_log_diff!=fi1_id_fisc_log_diff &
fim_id_fisc_log_diff!=fi1_id_fisc_log_diff &
!mi(fi1_id_fisc_log_diff,fif_id_fisc_log_diff,fim_id_fisc_log_diff)
//le père:
gen byte log2=1 if fim_id_fisc_log_diff==fi2_id_fisc_log_diff &
fif_id_fisc_log_diff!=fi2_id_fisc_log_diff & !mi(fi2_id_fisc_log_diff)
replace log2=2 if fif_id_fisc_log_diff==fi2_id_fisc_log_diff &
fim_id_fisc_log_diff!=fi2_id_fisc_log_diff & !mi(fi2_id_fisc_log_diff)
replace log2=3 if fif_id_fisc_log_diff==fi2_id_fisc_log_diff &
fim_id_fisc_log_diff==fi2_id_fisc_log_diff & !mi(fi2_id_fisc_log_diff)
replace log2=0 if fif_id_fisc_log_diff!=fi2_id_fisc_log_diff &
fim_id_fisc_log_diff!=fi2_id_fisc_log_diff &
!mi(fi2_id_fisc_log_diff,fif_id_fisc_log_diff,fim_id_fisc_log_diff)
replace log2=1 if log2==0 & f2==1
replace log2=2 if log2==0 & f2==2
```

```
replace log2=3 if log2==0 & f2==3
label define log 0 "Ni le père ni la mère" 1 "La mère" 2 "Le père" 3 "Le père
et la mère",replace
label values log1 log2 log
la var log1 "Qui est dans le logement 1 ?"
la var log2 "Qui est dans le logement 2 ?"
```

Construction de la variable *situ*

La variable *situ* sert à identifier les multiples situations familiales rencontrées par les enfants EDP dans notre base.

Avant de la construire, nous créons une variable dichotomique servant à identifier si l'enfant EDP présente, dans la longueur du panel, au moins une fois des informations sur la date de naissance de chacun de ses parents de naissance, à condition que celle-ci n'ait pas été imputée :

```
bys idi: egen byte ddnpnm = count(ddnp)
replace ddnpnm=(ddnpnm>0) if ddnpnm<.
replace ddnpnm=0 if n_i_pere_ind_nai_date==1
bys idi: egen byte ddnmnm = count(ddnm)
replace ddnmnm=(ddnmnm>0) if ddnmnm<.
replace ddnmnm=0 if n_i_mere_ind_nai_date==1
la var ddnpnm "Date de naissance du père de naissance non-manquante"
la var ddnmnm "Date de naissance de la mère de naissance non-manquante"
```

Nous passons ensuite à la construction de la variable *situ* :

```
*Situations manquantes, à exécuter en début de code :
gen int situ=.a if age_t>=18 & age_t<. //l'enfant n'est pas mineur
replace situ=.d if age_t==.d //l'enfant est décédé
replace situ=.u if age_t==.u //l'enfant n'est pas encore né
replace situ=.f if inlist(fi1_type_fisc,"01","1","02","2") & age_t<18
//l'enfant EDP est déclarant fiscal et a moins de 18 ans
replace situ=.m if _merge_fi==1 & situ==. //pas d'appariement aux dsf
replace situ=.l if _merge_fi>1 & mi(fi1_id_fisc_log_diff) &
mi(fi2_id_fisc_log_diff) & situ==. //pas de logement identifié
replace situ=.p if ddnpnm==0 & ddnmnm==1 & _merge_fi>1 & situ==. //pas de ddn
du père
replace situ=.q if ddnpnm==1 & ddnmnm==0 & _merge_fi>1 & situ==. //pas de ddn
de la mère
replace situ=.r if _merge_naissance==1 & _insee_update==0 & _merge_fi>1 &
situ==. //pas de bulletin de naissance

*L'enfant vit avec ses deux parents de naissance, toujours en couple
replace situ=1 if fim_liveswithchild==1 & fif_liveswithchild==1 &
fi1d_cideci=="M" & fi1p_cideci=="M" & mi(fi2_id_fisc_foy_diff) & fr1_typmen==2
& !mi(fi1_id_fisc_log_diff) & situ==. //vit avec les 2 parents de naissance
dans le même foyer, mariés
```

```
replace situ=2 if fim_liveswithchild==1 & fif_liveswithchild==1 &
fi1d_cideci=="O" & fi1p_cideci=="O" & mi(fi2_id_fisc_foy_diff) & fr1_typmen==2
& !mi(fi1_id_fisc_log_diff) & situ==. //vit avec les 2 parents de naissance
dans le même foyer, pacsés
replace situ=3 if fim_liveswithchild==1 & fif_liveswithchild==1 &
!inlist(fi1d_cideci,"M","O") & fi1_nascgen>=2 & fi1_nascgen<. &
mi(fi2_id_fisc_foy_diff) & fr1_typmen==2 & !mi(fi1_id_fisc_log_diff) & situ==.
//vit avec les 2 parents de naissance dans le même logement, en UL

*L'enfant vit en résidence alternée :
replace situ=110 if fim_liveswithchild==1 & fif_liveswithchild==1 &
fi1_t_charge=="H" & fi2_t_charge=="H" & fr1_typmen==1 & fr2_typmen==1 &
fi_nblogs==2 & !mi(fi1_id_fisc_log_diff) & situ==. //RA père et mère seuls
replace situ=120 if fim_liveswithchild==1 & fif_liveswithchild==1 &
fi1_t_charge=="H" & fi2_t_charge=="H" & fr1_typmen==1 & fr2_typmen==2 & f1==2
& f2==1  & fi_nblogs==2 & !mi(fi1_id_fisc_log_diff) & situ==. //père seul en 1
et mère en couple en 2
replace situ=120 if fim_liveswithchild==1 & fif_liveswithchild==1 &
fi1_t_charge=="H" & fi2_t_charge=="H" & fr1_typmen==2 & fr2_typmen==1 & f1==1
& f2==2 & fi_nblogs==2 & !mi(fi1_id_fisc_log_diff) & situ==. //mère en couple
en 1 et père seul en 2
replace situ=130 if fim_liveswithchild==1 & fif_liveswithchild==1 &
fi1_t_charge=="H" & fi2_t_charge=="H" & fr1_typmen==1 & fr2_typmen==2 & f1==1
& f2==2  & fi_nblogs==2 & !mi(fi1_id_fisc_log_diff) & situ==. //mère seule en
1 et père en couple en 2
replace situ=130 if fim_liveswithchild==1 & fif_liveswithchild==1 &
fi1_t_charge=="H" & fi2_t_charge=="H" & fr1_typmen==2 & fr2_typmen==1 & f1==2
& f2==1  & fi_nblogs==2  & !mi(fi1_id_fisc_log_diff) & situ==. //père en
couple en 1 et mère seule en 2
replace situ=140 if fim_liveswithchild==1 & fif_liveswithchild==1 &
fi1_t_charge=="H" & fi2_t_charge=="H" & fr1_typmen==2 & fr2_typmen==2 &
fi_nblogs==2  & !mi(fi1_id_fisc_log_diff) & situ==. //père et mère en couple
séparés
replace situ=150 if fim_liveswithchild==1 & fif_liveswithchild==1 &
fi1_t_charge=="H" & fi2_t_charge!="H" & f1==1 & f2==2  & fi_nblogs==2 &
fr1_typmen==1 & fr2_typmen==1  & !mi(fi1_id_fisc_log_diff) & situ==. //mère
déclare RA en 1 et père non-RA en 2 ; mère seule père seul
replace situ=150 if fim_liveswithchild==1 & fif_liveswithchild==1 &
fi1_t_charge!="H" & fi2_t_charge=="H" & f1==2 & f2==1  & fi_nblogs==2 &
fr1_typmen==1 & fr2_typmen==1  & !mi(fi1_id_fisc_log_diff) & situ==.  //père
déclare non-RA en 1 et mère RA en 2 ; mère seule père seul
replace situ=151 if fim_liveswithchild==1 & fif_liveswithchild==1 &
fi1_t_charge=="H" & fi2_t_charge!="H" & f1==1 & f2==2  & fi_nblogs==2 &
fr1_typmen==2 & fr2_typmen==1 & !mi(fi1_id_fisc_log_diff) & situ==.  //mère
déclare RA en 1 et père non-RA en 2 ; mère en couple père seul
replace situ=151 if fim_liveswithchild==1 & fif_liveswithchild==1 &
fi1_t_charge!="H" & fi2_t_charge=="H" & f1==2 & f2==1  & fi_nblogs==2 &
fr1_typmen==1 & fr2_typmen==2  & !mi(fi1_id_fisc_log_diff) & situ==. //père
déclare non-RA en 1 et mère RA en 2 ; mère en couple père seul
```

```
replace situ=152 if fim_liveswithchild==1 & fif_liveswithchild==1 &
fi1_t_charge=="H" & fi2_t_charge!="H" & f1==1 & f2==2  & fi_nblogs==2 &
fr1_typmen==1 & fr2_typmen==2  & !mi(fi1_id_fisc_log_diff) & situ==. //mère
déclare RA en 1 et père non-RA en 2 ; mère seule père en couple
replace situ=152 if fim_liveswithchild==1 & fif_liveswithchild==1 &
fi1_t_charge!="H" & fi2_t_charge=="H" & f1==2 & f2==1  & fi_nblogs==2 &
fr1_typmen==2 & fr2_typmen==1  & !mi(fi1_id_fisc_log_diff) & situ==. //père
déclare non-RA en 1 et mère RA en 2 ; mère seule père en couple
replace situ=153 if fim_liveswithchild==1 & fif_liveswithchild==1 &
fi1_t_charge=="H" & fi2_t_charge!="H" & f1==1 & f2==2  & fi_nblogs==2 &
fr1_typmen==2 & fr2_typmen==2  & !mi(fi1_id_fisc_log_diff) & situ==. //mère
déclare RA en 1 et père non-RA en 2 ; mère et père en couple
replace situ=153 if fim_liveswithchild==1 & fif_liveswithchild==1 &
fi1_t_charge!="H" & fi2_t_charge=="H" & f1==2 & f2==1  & fi_nblogs==2 &
fr1_typmen==2 & fr2_typmen==2  & !mi(fi1_id_fisc_log_diff) & situ==. //père
déclare non-RA en 1 et mère RA en 2 ; mère et père en couple
replace situ=154 if fim_liveswithchild==1 & fif_liveswithchild==1 &
fi1_t_charge=="H" & fi2_t_charge!="H" & f1==1 & f2==2  & fi_nblogs==2 &
(!inlist(fr1_typmen,1,2) | !inlist(fr2_typmen,1,2))  &
!mi(fi1_id_fisc_log_diff) & situ==. //mère déclare RA en 1 et père non-RA en 2
; situations conjugales des mère et père inconnues
replace situ=154 if fim_liveswithchild==1 & fif_liveswithchild==1 &
fi1_t_charge!="H" & fi2_t_charge=="H" & f1==2 & f2==1  & fi_nblogs==2 &
(!inlist(fr1_typmen,1,2) | !inlist(fr2_typmen,1,2))  &
!mi(fi1_id_fisc_log_diff) & situ==. //père déclare non-RA en 1 et mère RA en 2
; situations conjugales des mère et père inconnues
replace situ=160 if fim_liveswithchild==1 & fif_liveswithchild==1 &
fi1_t_charge=="H" & fi2_t_charge!="H" & f1==2 & f2==1  & fi_nblogs==2 &
fr1_typmen==1 & fr2_typmen==1  & !mi(fi1_id_fisc_log_diff) & situ==. //père
déclare RA en 1 et mère non-RA en 2, mère et père seuls
replace situ=160 if fim_liveswithchild==1 & fif_liveswithchild==1 &
fi1_t_charge!="H" & fi2_t_charge=="H" & f1==1 & f2==2 & fi_nblogs==2 &
fr1_typmen==1 & fr2_typmen==1  & !mi(fi1_id_fisc_log_diff) & situ==. //mère
déclare non-RA en 1 et père RA en 2, mère et père seuls
replace situ=161 if fim_liveswithchild==1 & fif_liveswithchild==1 &
fi1_t_charge=="H" & fi2_t_charge!="H" & f1==2 & f2==1  & fi_nblogs==2 &
fr1_typmen==1 & fr2_typmen==2  & !mi(fi1_id_fisc_log_diff) & situ==. //père
déclare RA en 1 et mère non-RA en 2, mère en couple et père seul
replace situ=161 if fim_liveswithchild==1 & fif_liveswithchild==1 &
fi1_t_charge!="H" & fi2_t_charge=="H" & f1==1 & f2==2 & fi_nblogs==2 &
fr1_typmen==2 & fr2_typmen==1  & !mi(fi1_id_fisc_log_diff) & situ==. //mère
déclare non-RA en 1 et père RA en 2, mère en couple et père seul
replace situ=162 if fim_liveswithchild==1 & fif_liveswithchild==1 &
fi1_t_charge=="H" & fi2_t_charge!="H" & f1==2 & f2==1  & fi_nblogs==2 &
fr1_typmen==2 & fr2_typmen==1  & !mi(fi1_id_fisc_log_diff) & situ==. //père
déclare RA en 1 et mère non-RA en 2, mère seule et père en couple
replace situ=162 if fim_liveswithchild==1 & fif_liveswithchild==1 &
fi1_t_charge!="H" & fi2_t_charge=="H" & f1==1 & f2==2 & fi_nblogs==2 &
```

```
fr1_typmen==1 & fr2_typmen==2  & !mi(fi1_id_fisc_log_diff) & situ==. //mère
déclare non-RA en 1 et père RA en 2, mère seule et père en couple
replace situ=163 if fim_liveswithchild==1 & fif_liveswithchild==1 &
fi1_t_charge=="H" & fi2_t_charge!="H" & f1==2 & f2==1  & fi_nblogs==2 &
fr1_typmen==2 & fr2_typmen==2  & !mi(fi1_id_fisc_log_diff) & situ==. //père
déclare RA en 1 et mère non-RA en 2, mère et père en couple
replace situ=163 if fim_liveswithchild==1 & fif_liveswithchild==1 &
fi1_t_charge!="H" & fi2_t_charge=="H" & f1==1 & f2==2 & fi_nblogs==2 &
fr1_typmen==2 & fr2_typmen==2  & !mi(fi1_id_fisc_log_diff) & situ==. //mère
déclare non-RA en 1 et père RA en 2, mère et père en couple
replace situ=164 if fim_liveswithchild==1 & fif_liveswithchild==1 &
fi1_t_charge=="H" & fi2_t_charge!="H" & f1==2 & f2==1  & fi_nblogs==2 &
(!inlist(fr1_typmen,1,2) | !inlist(fr2_typmen,1,2)) &
!mi(fi1_id_fisc_log_diff) & situ==.  //père déclare RA en 1 et mère non-RA en
2, situations conjugales mère et père inconnues
replace situ=164 if fim_liveswithchild==1 & fif_liveswithchild==1 &
fi1_t_charge!="H" & fi2_t_charge=="H" & f1==1 & f2==2 & fi_nblogs==2 &
(!inlist(fr1_typmen,1,2) | !inlist(fr2_typmen,1,2)) &
!mi(fi1_id_fisc_log_diff) & situ==. //mère déclare non-RA en 1 et père RA en
2, situations conjugales mère et père inconnues
replace situ=170 if fim_liveswithchild==1 & fif_liveswithchild==1 &
fi1_t_charge!="H" & fi2_t_charge!="H" & f1==1 & f2==2 & fi_nblogs==2 &
fr1_typmen==1 & fr2_typmen==1  & !mi(fi1_id_fisc_log_diff) & situ==. //père et
mère déclarent non-RA, mère et père seuls
replace situ=170 if fim_liveswithchild==1 & fif_liveswithchild==1 &
fi1_t_charge!="H" & fi2_t_charge!="H" & f1==2 & f2==1 & fi_nblogs==2 &
fr1_typmen==1 & fr2_typmen==1  & !mi(fi1_id_fisc_log_diff) & situ==. //père et
mère déclarent non-RA, mère et père seuls
replace situ=171 if fim_liveswithchild==1 & fif_liveswithchild==1 &
fi1_t_charge!="H" & fi2_t_charge!="H" & f1==1 & f2==2 & fi_nblogs==2 &
fr1_typmen==2 & fr2_typmen==1 & !mi(fi1_id_fisc_log_diff) & situ==.  //père et
mère déclarent non-RA, mère en couple et père seul
replace situ=171 if fim_liveswithchild==1 & fif_liveswithchild==1 &
fi1_t_charge!="H" & fi2_t_charge!="H" & f1==2 & f2==1 & fi_nblogs==2 &
fr1_typmen==1 & fr2_typmen==2 & !mi(fi1_id_fisc_log_diff) & situ==.  //père et
mère déclarent non-RA, mère en couple et père seul
replace situ=172 if fim_liveswithchild==1 & fif_liveswithchild==1 &
fi1_t_charge!="H" & fi2_t_charge!="H" & f1==1 & f2==2 & fi_nblogs==2 &
fr1_typmen==1 & fr2_typmen==2  & !mi(fi1_id_fisc_log_diff) & situ==. //père et
mère déclarent non-RA, mère seule et père en couple
replace situ=172 if fim_liveswithchild==1 & fif_liveswithchild==1 &
fi1_t_charge!="H" & fi2_t_charge!="H" & f1==2 & f2==1 & fi_nblogs==2 &
fr1_typmen==2 & fr2_typmen==1  & !mi(fi1_id_fisc_log_diff) & situ==. //père et
mère déclarent non-RA, mère seule et père en couple
replace situ=173 if fim_liveswithchild==1 & fif_liveswithchild==1 &
fi1_t_charge!="H" & fi2_t_charge!="H" & f1==1 & f2==2 & fi_nblogs==2 &
fr1_typmen==2 & fr2_typmen==2  & !mi(fi1_id_fisc_log_diff) & situ==. //père et
mère déclarent non-RA, mère et père en couple
```

```
replace situ=173 if fim_liveswithchild==1 & fif_liveswithchild==1 &
fi1_t_charge!="H" & fi2_t_charge!="H" & f1==2 & f2==1 & fi_nblogs==2 &
fr1_typmen==2 & fr2_typmen==2  & !mi(fi1_id_fisc_log_diff) & situ==. //père et
mère déclarent non-RA, mère et père en couple
replace situ=174 if fim_liveswithchild==1 & fif_liveswithchild==1 &
fi1_t_charge!="H" & fi2_t_charge!="H" & f1==1 & f2==2 & fi_nblogs==2 &
(!inlist(fr1_typmen,1,2) | !inlist(fr2_typmen,1,2)) &
!mi(fi1_id_fisc_log_diff) & situ==.  //père et mère déclarent non-RA,
situations conjugales mère et père inconnues
replace situ=174 if fim_liveswithchild==1 & fif_liveswithchild==1 &
fi1_t_charge!="H" & fi2_t_charge!="H" & f1==2 & f2==1 & fi_nblogs==2 &
(!inlist(fr1_typmen,1,2) | !inlist(fr2_typmen,1,2))  &
!mi(fi1_id_fisc_log_diff) & situ==. //père et mère déclarent non-RA,
situations conjugales mère et père inconnues
replace situ=180 if fim_liveswithchild==1 & fif_liveswithchild==0 &
((fi1_t_charge=="H" & f1==1 & fr1_typmen==1)|(fi2_t_charge=="H" & f2==1 &
fr2_typmen==1))  & !mi(fi1_id_fisc_log_diff) & situ==. //mère déclare en RA,
père ne le déclare pas, mère seule
replace situ=181 if fim_liveswithchild==1 & fif_liveswithchild==0 &
((fi1_t_charge=="H" & f1==1 & fr1_typmen==2)|(fi2_t_charge=="H" & f2==1 &
fr2_typmen==2))  & !mi(fi1_id_fisc_log_diff) & situ==. //mère déclare en RA,
père ne le déclare pas, mère en couple
replace situ=182 if fim_liveswithchild==1 & fif_liveswithchild==0 &
((fi1_t_charge=="H" & f1==1 & !inlist(fr1_typmen,1,2))|(fi2_t_charge=="H" &
f2==1 & !inlist(fr2_typmen,1,2)))  & !mi(fi1_id_fisc_log_diff) & situ==.
//mère déclare en RA, père ne le déclare pas, mère en situation conjugale
inconnue
replace situ=190 if fim_liveswithchild==0 & fif_liveswithchild==1 &
((fi1_t_charge=="H" & f1==2 & fr1_typmen==1)|(fi2_t_charge=="H" & f2==2 &
fr2_typmen==1))  & !mi(fi1_id_fisc_log_diff) & situ==. //père déclare en RA,
mère ne le déclare pas, père seul
replace situ=191 if fim_liveswithchild==0 & fif_liveswithchild==1 &
((fi1_t_charge=="H" & f1==2 & fr1_typmen==2)|(fi2_t_charge=="H" & f2==2 &
fr2_typmen==2))  & !mi(fi1_id_fisc_log_diff) & situ==. //père déclare en RA,
mère ne le déclare pas, père en couple
replace situ=192 if fim_liveswithchild==0 & fif_liveswithchild==1 &
((fi1_t_charge=="H" & f1==2 & !inlist(fr1_typmen,1,2))|(fi2_t_charge=="H" &
f2==2 & inlist(fr2_typmen,1,2)))  & !mi(fi1_id_fisc_log_diff) & situ==. //père
déclare en RA, mère ne le déclare pas, père en situation conjugale inconnue
replace situ=195 if fim_liveswithchild==1 & fif_liveswithchild>=. & ddnpnm==0
& (fi1_t_charge=="H" | fi2_t_charge=="H") & (_merge_naissance>1 |
_insee_update==1) & age_t<18 //mère déclare en RA, père absent du bulletin de
naissance
replace situ=196 if fim_liveswithchild>=. & fif_liveswithchild==1 & ddnmnm==0
& (fi1_t_charge=="H" | fi2_t_charge=="H") & (_merge_naissance>1 |
_insee_update==1) & age_t<18 //père déclare en RA, mère absente du bulletin de
naissance
```

```
replace situ=200 if (fi1_t_charge=="H" | fi2_t_charge=="H") &
_merge_naissance==1 & _insee_update==0 & fi_nblogs==2  &
!mi(fi1_id_fisc_log_diff)  & age_t<18 //déclaré en RA mais pas de bulletin de
naissance
replace situ=210 if (fi1_t_charge=="H" | fi2_t_charge=="H") &
mi(fi2_id_fisc_log_diff) & (_merge_naissance>1 | _insee_update==1) &
fi_nblogs==1  & !mi(fi1_id_fisc_log_diff) & situ==. //déclaré en RA mais un
seul logement identifié
replace situ=220 if (fi1_t_charge=="H" | fi2_t_charge=="H") &
(_merge_naissance>1 | _insee_update==1) & !mi(fi2_id_fisc_log_diff) & situ==.
& !mi(fi1_id_fisc_log_diff) //Autres situations de RA


*L'enfant vit avec l'un de ses parents de naissance :
replace situ=310 if fim_liveswithchild==1 & fif_liveswithchild==0 &
fr1_typmen==1  & fi_nblogs==1 & situ==.  & !mi(fi1_id_fisc_log_diff) //mère
seulement, seule
replace situ=320 if fim_liveswithchild==1 & fif_liveswithchild==0 &
fr1_typmen==2 & fi_nblogs==1 & situ==.  & !mi(fi1_id_fisc_log_diff) //mère
seulement, en couple
replace situ=330 if fim_liveswithchild==1 & fif_liveswithchild>=. & ddnpnm==0
& fr1_typmen==1 & fi_nblogs==1 & !mi(fi1_id_fisc_log_diff) & age_t<18 //mère
seulement, seule, père absent du bulletin de naissance
replace situ=340 if fim_liveswithchild==1 & fif_liveswithchild>=. & ddnpnm==0
& fr1_typmen==2 & fi_nblogs==1 & !mi(fi1_id_fisc_log_diff) & age_t<18 //mère
seulement, en couple, père absent du bulletin de naissance
replace situ=350 if fim_liveswithchild==0 & fif_liveswithchild==1 &
fr1_typmen==1 & fi_nblogs==1 & situ==.  & !mi(fi1_id_fisc_log_diff) //père
seulement, seul
replace situ=360 if fim_liveswithchild==0 & fif_liveswithchild==1 &
fr1_typmen==2 & fi_nblogs==1 & situ==.  & !mi(fi1_id_fisc_log_diff) //père
seulement, en couple


*Autres situations:
replace situ=400 if fim_liveswithchild==0 & fif_liveswithchild==0 & situ==. &
!mi(fi1_id_fisc_log_diff) //ni père ni mère
replace situ=510 if situ==.r & fr1_typmen==1 & fi_nblogs==1 & fi1d_sexe==2 &
!mi(fi1_id_fisc_log_diff) & age_t<18 //pas de bulletin de naissance, femme
monoparentale
replace situ=511 if situ==.r & fr1_typmen==1 & fi_nblogs==1 & fi1d_sexe==1 &
!mi(fi1_id_fisc_log_diff) & age_t<18 //pas de bulletin de naissance, homme
monoparental
replace situ=520 if situ==.r & fr1_typmen==2 & fi_nblogs==1 &
!mi(fi1_id_fisc_log_diff) & age_t<18 //pas de bulletin de naissance, couple
avec enfant
la var situ "Situations familiales - ajusté des DDN des PDN Non-Manquantes"
```

On précise les labels de la variable *situ* :

```
label define situ /*
*/ .a "L'enfant n'est pas mineur" /*
*/ .f "L'enfant est déclarant fiscal" /*
*/ .m "Les DSF de l'enfant sont manquantes" /*
*/ .l "Le logement de l'enfant est manquant" /*
*/ .p "La date de naissance du père est manquante" /*
*/ .q "La date de naissance de la mère est manquante" /*
*/ .r "Pas de bulletin de naissance" /*
*/ 1 "Vit avec les 2 parents de naissance mariés dans le même foyer" /*
*/ 2 "Vit avec les 2 parents de naissance pacsés dans le même foyer" /*
*/ 3 "Vit avec les 2 parents de naissance en UL dans le même logement" /*
*/ 110 "Déclaré en RA par les 2 PDN séparés seuls dans 2 foyers distincts" /*
*/ 120 "Déclaré en RA par les 2 PDN séparés dans 2 foyers distincts, père seul
et mère en couple" /*
*/ 130 "Déclaré en RA par les 2 PDN séparés dans 2 foyers distincts, père en
couple et mère seule" /*
*/ 140 "Déclaré en RA par les 2 PDN séparés en couple dans 2 foyers distincts"
/*
*/ 150 "Déclaré en RA par la mère dans un foyer, en charge principale par le
père dans l'autre, mère et père seuls" /*
*/ 151 "Déclaré en RA par la mère dans un foyer, en charge principale par le
père dans l'autre, mère en couple et père seul" /*
*/ 152 "Déclaré en RA par la mère dans un foyer, en charge principale par le
père dans l'autre, mère seule et père en couple" /*
*/ 153 "Déclaré en RA par la mère dans un foyer, en charge principale par le
père dans l'autre, mère en couple et père en couple" /*
*/ 154 "Déclaré en RA par la mère dans un foyer, en charge principale par le
père dans l'autre, situations conjugales des PDN inconnues" /*
*/ 160 "Déclaré en RA par le père dans un foyer, en charge principale par la
mère dans l'autre, mère et père seuls" /*
*/ 161 "Déclaré en RA par le père dans un foyer, en charge principale par la
mère dans l'autre, mère en couple et père seul" /*
*/ 162 "Déclaré en RA par le père dans un foyer, en charge principale par la
mère dans l'autre, mère seule et père en couple" /*
*/ 163 "Déclaré en RA par le père dans un foyer, en charge principale par la
mère dans l'autre, mère en couple et père en couple" /*
*/ 164 "Déclaré en RA par le père dans un foyer, en charge principale par la
mère dans l'autre, situations conjugales des PDN inconnues" /*
*/ 170 "Vit avec ses 2 PDN dans 2 logements distincts, non-déclaré en RA, mère
et père seuls" /*
*/ 171 "Vit avec ses 2 PDN dans 2 logements distincts, non-déclaré en RA, mère
en couple et père seul" /*
*/ 172 "Vit avec ses 2 PDN dans 2 logements distincts, non-déclaré en RA, mère
seule et père en couple" /*
*/ 173 "Vit avec ses 2 PDN dans 2 logements distincts, non-déclaré en RA, mère
en couple et père en couple" /*
```

```
*/ 174 "Vit avec ses 2 PDN dans 2 logements distincts, non-déclaré en RA,
situations conjugales des PDN inconnues" /*
*/ 180 "La mère déclare en RA et le père ne déclare pas l'enfant, mère seule"
/*
*/ 181 "La mère déclare en RA et le père ne déclare pas l'enfant, mère en
couple" /*
*/ 182 "La mère déclare en RA et le père ne déclare pas l'enfant, mère en
situation conjugale inconnue" /*
*/ 190 "Le père déclare en RA et la mère ne déclare pas l'enfant, père seul"
/*
*/ 191 "Le père déclare en RA et la mère ne déclare pas l'enfant, père en
couple" /*
*/ 192 "Le père déclare en RA et la mère ne déclare pas l'enfant, père en
situation conjugale inconnue" /*
*/ 195 "La mère déclare en RA et le père est absent du bulletin de
naissance"/*
*/ 196 "Le père déclare en RA et la mère est absente du bulletin de
naissance"/*
*/ 200 "L'enfant est déclaré en RA mais n'a pas de bulletin de naissance" /*
*/ 210 "L'enfant est déclaré en RA mais un seul logement identifié" /*
*/ 220 "Autres situations de RA" /*
*/ 310 "Vit avec sa mère seulement, seule" /*
*/ 320 "Vit avec sa mère seulement, en couple" /*
*/ 330 "Vit avec sa mère seulement, seule, père absent du bulletin de
naissance" /*
*/ 340 "Vit avec sa mère seulement, en couple, père absent du bulletin de
naissance" /*
*/ 350 "Vit avec son père seulement, seul" /*
*/ 360 "Vit avec son père seulement, en couple" /*
*/ 400 "Ne vit ni avec son père ni avec sa mère" /*
*/ 510 "Pas de bulletin de naissance, femme monoparentale" /*
*/ 511 "Pas de bulletin de naissance, homme monoparental" /*
*/ 520 "Pas de bulletin de naissance, couple avec enfant(s)" /*
*/ ,replace
label values situ situ
la var situ "Situation familiale"
```

Construction des types d'union

Dans cette section, nous nous intéressons au type d'union qu'il y a entre les adultes cohabitant avec l'enfant EDP. Etant donné que l'enfant peut être déclaré dans deux foyers différents, qui peuvent ou non se situer dans le même logement, que l'enfant peut ne pas être déclaré par l'un des adultes exerçant une parenté sur lui, les différents types d'union peuvent être nombreux. On distinguera donc les couples mariés/pacsés (MP) des unions libres (UL) au sein des couples. On distinguera également les situations homoparentales au sein de cette variable, soit des couples de femmes (FF) ou d'hommes (HH), soit des couples de sexe différent (FH).

Dans les cas d'unions libres, on utilisera les variables *nmascb* et *nfascb*, respectivement le nombre d'hommes et de femmes de génération plus ancienne (>=15 ans), après correction des erreurs de sexe, vivant dans le même logement que l'enfant. On se contraint à n'étudier les unions libres que dans les logements présentant deux adultes de génération plus ancienne que l'enfant, pas davantage.

La première partie de la construction cherche à identifier les types d'union dans les situations où l'enfant n'est déclaré que dans un seul foyer, ou lorsqu'il n'y a qu'un seul parent (de naissance ou non) dans l'autre foyer :

```
gen int union=1 if fi1d_sexe!=fi1p_sexe & !mi(fi1d_sexe,fi1p_sexe) &
fi_nbfoys==1 //f1 est MP FH, f2 absent
replace union=2 if fi1_nmascb==1 & fi1_nfascb==1 & fi_nbfoys==1 &
fr1_typmen==2 & !inlist(fi1d_cideci,"M","O") //f1 est UL FH, f2 absent
replace union=3 if fi1d_sexe==fi1p_sexe & fi1d_sexe==2 & fi1d_ddn!=fi1p_ddn &
!mi(fi1d_ddn,fi1p_ddn) & fi_nbfoys==1 //f1 est MP FF, f2 absent
replace union=4 if fi1_nmascb==0 & fi1_nfascb==2 & fi_nbfoys==1 &
fr1_typmen==2 & !inlist(fi1d_cideci,"M","O") //f1 est UL FF, f2 absent
replace union=5 if fi1d_sexe==fi1p_sexe & fi1d_sexe==1 & fi1d_ddn!=fi1p_ddn &
!mi(fi1d_ddn,fi1p_ddn) & fi_nbfoys==1 //f1 est MP HH, f2 absent
replace union=6 if fi1_nmascb==2 & fi1_nfascb==0 & fi_nbfoys==1 &
fr1_typmen==2 & !inlist(fi1d_cideci,"M","O") //f1 est UL HH, f2 absent

replace union=101 if fi1d_sexe!=fi1p_sexe & !mi(fi1d_sexe,fi1p_sexe) &
fr2_typmen==1 //f1 est MP FH, f2 seul
replace union=102 if fi1_nmascb==1 & fi1_nfascb==1 & fr2_typmen==1 &
fr1_typmen==2 & !inlist(fi1d_cideci,"M","O") //f1 est UL FH, f2 seul
replace union=103 if fi1d_sexe==fi1p_sexe & fi1d_sexe==2 & fi1d_ddn!=fi1p_ddn
& !mi(fi1d_ddn,fi1p_ddn) & fr2_typmen==1 //f1 est MP FF, f2 seul
replace union=104 if fi1_nmascb==0 & fi1_nfascb==2 & fr2_typmen==1 &
fr1_typmen==2 & !inlist(fi1d_cideci,"M","O") //f1 est UL FF, f2 seul
replace union=105 if fi1d_sexe==fi1p_sexe & fi1d_sexe==1 & fi1d_ddn!=fi1p_ddn
& !mi(fi1d_ddn,fi1p_ddn) & fr2_typmen==1 //f1 est MP HH, f2 seul
replace union=106 if fi1_nmascb==2 & fi1_nfascb==0 & fr2_typmen==1 &
fr1_typmen==2 & !inlist(fi1d_cideci,"M","O") //f1 est UL HH, f2 seul
replace union=201 if fr1_typmen==1 & fi2d_sexe!=fi2p_sexe &
!mi(fi2d_sexe,fi2p_sexe)  //f1 est seul, f2 est MP FH
replace union=202 if fr1_typmen==1 & fi2_nmascb==1 & fi2_nfascb==1 &
fr2_typmen==2 & !inlist(fi2d_cideci,"M","O") //f1 est seul, f2 est UL FH
replace union=203 if fr1_typmen==1 & fi2d_sexe==fi2p_sexe & fi2d_sexe==2 &
fi2d_ddn!=fi2p_ddn & !mi(fi2d_ddn,fi2p_ddn) //f1 est seul, f2 MP FF
replace union=204 if fr1_typmen==1 & fi2_nmascb==0 & fi2_nfascb==2 &
fr2_typmen==2 & !inlist(fi2d_cideci,"M","O") //f1 est seul, f2 UL FF
replace union=205 if fr1_typmen==1 & fi2d_sexe==fi2p_sexe & fi2d_sexe==1 &
fi2d_ddn!=fi2p_ddn & !mi(fi2d_ddn,fi2p_ddn) //f1 est seul, f2 MP HH
replace union=206 if fr1_typmen==1 & fi2_nmascb==2 & fi2_nfascb==0 &
fr2_typmen==2 & !inlist(fi2d_cideci,"M","O") //f1 est seul, f2 UL HH

label define union /*
```

```
*/ 1 "f1 est MPFH ; pas de f2" /*
*/ 2 "f1 est ULFH ; pas de f2" /*
*/ 3 "f1 est MPFF ; pas de f2" /*
*/ 4 "f1 est ULFF ; pas de f2" /*
*/ 5 "f1 est MPHH ; pas de f2" /*
*/ 6 "f1 est ULHH ; pas de f2" /*
*/ 101 "f1 est MPFH ; f2 est seul" /*
*/ 102 "f1 est ULFH ; f2 est seul" /*
*/ 103 "f1 est MPFF ; f2 est seul" /*
*/ 104 "f1 est ULFF ; f2 est seul" /*
*/ 105 "f1 est MPHH ; f2 est seul" /*
*/ 106 "f1 est ULHH ; f2 est seul" /*
*/ 201 "f1 est seul ; f2 est MPFH" /*
*/ 202 "f1 est seul ; f2 est ULFH" /*
*/ 203 "f1 est seul ; f2 est MPFF" /*
*/ 204 "f1 est seul ; f2 est ULFF" /*
*/ 205 "f1 est seul ; f2 est MPHH" /*
*/ 206 "f1 est seul ; f2 est ULHH" /*
*/,replace
```

Dans la deuxième partie du code, on cherche à déterminer toutes les combinaisons de type d'union dans l'un *et* l'autre des deux foyers fiscaux où est déclaré l'enfant :

```
local i=0
foreach m in MPFH ULFH MPFF ULFF MPHH ULHH {
    if "`m'"=="MPFH" local f1 "fi1d_sexe!=fi1p_sexe &
!mi(fi1d_sexe,fi1p_sexe)"
    else if "`m'"=="ULFH" local f1 "fi1_nmascb==1 & fi1_nfascb==1 &
fr1_typmen==2 & !inlist(fi1d_cideci,"M","O")"
    else if "`m'"=="MPFF" local f1 "fi1d_sexe==fi1p_sexe & fi1d_sexe==2 &
fi1d_ddn!=fi1p_ddn & !mi(fi1d_ddn,fi1p_ddn)"
    else if "`m'"=="ULFF" local f1 "fi1_nmascb==0 & fi1_nfascb==2 &
fr1_typmen==2 & !inlist(fi1d_cideci,"M","O")"
    else if "`m'"=="MPHH" local f1 "fi1d_sexe==fi1p_sexe & fi1d_sexe==1 &
fi1d_ddn!=fi1p_ddn & !mi(fi1d_ddn,fi1p_ddn)"
    else if "`m'"=="ULHH" local f1 "fi1_nmascb==2 & fi1_nfascb==0 &
fr1_typmen==2 & !inlist(fi1d_cideci,"M","O")"
    foreach n in MPFH ULFH MPFF ULFF MPHH ULHH {
        if "`n'"=="MPFH" local f2 "fi2d_sexe!=fi2p_sexe &
!mi(fi2d_sexe,fi2p_sexe)"
        else if "`n'"=="ULFH" local f2 "fi2_nmascb==1 & fi2_nfascb==1 &
fr2_typmen==2 & !inlist(fi2d_cideci,"M","O")"
        else if "`n'"=="MPFF" local f2 "fi2d_sexe==fi2p_sexe & fi2d_sexe==2 &
fi2d_ddn!=fi2p_ddn & !mi(fi2d_ddn,fi2p_ddn)"
        else if "`n'"=="ULFF" local f2 "fi2_nmascb==0 & fi2_nfascb==2 &
fr2_typmen==2 & !inlist(fi2d_cideci,"M","O")"
        else if "`n'"=="MPHH" local f2 "fi2d_sexe==fi2p_sexe & fi2d_sexe==1 &
fi2d_ddn!=fi2p_ddn & !mi(fi2d_ddn,fi2p_ddn)"
```

```
        else if "`n'"=="ULHH" local f2 "fi2_nmascb==2 & fi2_nfascb==0 &
fr2_typmen==2 & !inlist(fi2d_cideci,"M","O")"
        local i=`i'+1
        local j=300+`i'
        replace union=`j' if `f1' & `f2'
        label define union `j' "f1 est `m' ; f2 est `n'",add
    }
}
label values union union
la var union "Type d'union"
```

On compare également les types d'union selon qu'ils soient mariés/pacsés, ou en unions libres, dans l'un ou l'autre des foyers :

```
*Mariages / Pacs / UL :
recode union (1 3 5 101 103 105 201 203 205 301 303 305 313 315 317 325 327
329=0 "MP")(2 4 6 102 104 106 202 204 206 308 310 312 320 322 332 336=1
"UL")(302 304 306 307 309 311 314 319 321 326 331 335=2 "MP/UL"),gen(ul)
recode union (201/206=0) (1 3 5 101 103 105 301/306 313/317 325/329= 1)(2 4 6
102 104 106 307/312 319/322 331/336=3) ,gen(tuf1)
replace tuf1=2 if fi1d_cideci=="O" & tuf1==1
recode union (1/6=.m)(101/106=0)(201 203 205 301 303 305 307 309 311 313 315
317 319 321 325 327 329 331 335=1)(202 204 206 302 304 306 308 310 312 314 320
322 326 332 336=3),gen(tuf2)
replace tuf2=2 if fi2d_cideci=="O" & tuf2==1
label define tuf .m "Pas de foyer 2" 0 "Seul-e" 1 "Mariés" 2 "Pacsés" 3 "Union
libre",replace
label values tuf1 tuf2 tuf
la var ul "Unions libres ou autre"
la var tuf1 "Type d'union - foyer 1"
la var tuf2 "Type d'union - foyer 2"
```

Dans le code suivant, on utilise les valeurs créées par la variable *union*, pour reconstituer des situations de familles « traditionnelles » (situ=1,2,3). Il s'agit de situations où les deux foyers fiscaux sont habités par des couples (mariés/pacsés ou en union libre), mais dont le logement est unique (et non-manquant) :

```
*Imputation des situations familiales où les foyer 1 et 2 sont les mêmes à
l'ordrefip près:
replace situ=1 if situ==. & union==301 & fi1d_cideci=="M" & fi_nblogs==1 &
!mi(fi1_id_fisc_log_diff)
replace situ=2 if situ==. & union==301 & fi1d_cideci=="O" & fi_nblogs==1 &
!mi(fi1_id_fisc_log_diff)
replace situ=3 if situ==. & union==308 & fi_nblogs==1 &
!mi(fi1_id_fisc_log_diff)
```

Construction des familles homoparentales

Après avoir construit la variable *union*, on peut la recoder plus simplement pour catégoriser les enfants selon qu'il vive au sein d'au moins une famille homoparentale, ou non :

```
recode union (1 2 101 102 201 202 301 302 307 308 = 0 "FH")(3 4 103 104 203
204 303 304 309 310 313/316 318/322 =1 "FF")(5 6 105 106 205 206 305 306 311
312 325 326 328/336 =2 "HH") (317 327 = 3 "FF/HH"),gen(fhp)
la var fhp "Familles homoparentales"
```

Quatre catégories sont créées : les enfants vivant avec des parents de sexe différent, quelque soit le logement où ils habitent (FH) ; les enfants vivant avec un couple de parents de même sexe (FF et HH) ; et les enfants vivant avec un couple de femmes, et un couple d'hommes, dans deux logements différents (FF/HH).

Décès d'un parent de naissance

Nous créons pour chaque parent de naissance une variable qui vaut 1 lorsque le parent décède, et 0 sinon. Pour cela, nous ne pouvons nous appuyer sur la variable *i_fisc*_deces car elle n'est disponible qu'à partir de 2016. De même, la variable *dec_date* ne nous est utile que pour les parents EDP. Nous nous appuyons donc sur la base exhaustive des décès de l'INSEE. Celle-ci ne permet pas d'identifier le logement de l'enfant, mais nous pouvons retrouver les caractéristiques du parent de naissance en nous fondant sur son genre, sa date et son lieu de naissance, ainsi que sa ville de domicile l'année de son décès. Par soucis de légèreté, et parce que cette variable n'a pour but que de distinguer les séparations de couple des veuvages, nous ne nous intéresserons qu'aux décès survenus sur la période d'observation fiscale 2011-2019. Ainsi, si un parent est décédé par exemple en 2005, la variable ne vaudra jamais 1 car on n'a pas pu observer le décès, mais vaudra 0. Similairement, si un parent décède en 2015, la valeur sera de 1 cette année-là, et nulle les autres années.

On doit noter que les variables *zoxyzd* et *dacoed* permettent de savoir si le conjoint de la personne est décédé et de dater cet évènement. Toutefois, cela n'est utile que pour les couples mariés ou pacsés. Ainsi, la procédure que nous utilisons ici sera utile également pour les couples en union libre.

On commence par récupérer la clé d'appariement qui nous manque : le code commune (et département) de résidence des parents. Cette clé aurait pu être récupérée dès l'étape de la section « Appariement des informations fiscales des parents de naissance », page 42, mais nous choisissons plutôt de montrer ici un exemple de code nous permettant de récupérer cette clé en aval de cette section :

```
*Collecte des décès:
//Récupération de la clé d'appariement : département & commune de résidence
des parents:
set more off
preserve
tempfile fi
forvalues i=2011/2019 {
    tempfile fi`i'
    use edp_be2019_fisc_individu_`i'.dta,clear
    gen int annee=`i'
    egen fim_idfoy=concat(id_fisc_foy_diff ordrefip)
    clonevar fif_idfoy=fim_idfoy
```

```
    egen fim_comdom=concat(csdep cne1)
    clonevar fif_comdom=fim_comdom
    drop if mi(fim_idfoy,fim_comdom)
    duplicates drop fim_idfoy fim_comdom,force
    keep annee fim_* fif_*
    save `fi`i'',replace
}
use `fi2011',clear
append using `fi2012' `fi2013' `fi2014' `fi2015' `fi2016' `fi2017' `fi2018'
`fi2019'
save `fi',replace
restore
merge m:1 fim_idfoy annee using `fi',keep(1 3) keepusing(fim_comdom)
nogenerate
merge m:1 fif_idfoy annee using `fi',keep(1 3) keepusing(fif_comdom)
nogenerate
```

Ensuite, nous récupérons les informations concernant les décès exhaustifs, que nous apparions ensuite avec la base EDP grâce à la clé d'appariement formée notamment par *fim_comdom* et *fif_comdom* :

```
//On fixe les clés d'appariement pour toutes les années:
sort id_diff fim_comdom
by id_diff: gen fim_comdomm=fim_comdom[_N]
sort id_diff fim_codnais
by id_diff: gen fim_codnaism=fim_codnais[_N]
bys id_diff: egen ddnmm=max(ddnm)
sort id_diff fif_comdom
by id_diff: gen fif_comdomm=fif_comdom[_N]
sort id_diff fif_codnais
by id_diff: gen fif_codnaism=fif_codnais[_N]
bys id_diff: egen ddnpm=max(ddnp)
//Récupération des informations sur les décès:
set more off
preserve
tempfile deces
forvalues i=2011/2019 { //mort des mères
    tempfile d`i'
    use deces`i'.dta,clear
    gen int annee=`i'
    keep if sexe=="2"
    gen ddnmm=mdy( real(mnais), real(jnais), real(anais))
    gen dddm=mdy( real(mdec), real(jdec), real(adec))
    format ddnmm dddm %td
    gen byte fim_deces=1
    replace comnais=pnais if mi(comnais) & !mi(pnais) //pour les mères nées à
l'étranger
    ren comdom fim_comdomm
```

```
    ren comnais fim_codnaism
    keep if !mi(ddnmm,fim_codnaism,fim_comdomm,dddm)
    duplicates drop ddnmm fim_codnaism fim_comdomm,force
    keep annee ddnmm dddm fim_codnaism fim_comdomm fim_deces
    save `d`i'',replace
}
use `d2011',clear
append using `d2012' `d2013' `d2014' `d2015' `d2016' `d2017' `d2018' `d2019'
save `deces',replace
restore
merge m:1 annee ddnmm fim_codnaism fim_comdomm using `deces',keep(1 3)
keepusing(fim_deces dddm) nogenerate
replace fim_deces=0 if fim_deces==. & !mi(fim_type_fisc)
drop ddnmm fim_codnaism fim_comdomm
preserve
tempfile deces
forvalues i=2011/2019 { //mort des pères
    tempfile d`i'
    use deces`i'.dta,clear
    gen int annee=`i'
    keep if sexe=="1"
    gen ddnpm=mdy( real(mnais), real(jnais), real(anais))
    gen dddp=mdy( real(mdec), real(jdec), real(adec))
    format ddnpm dddp %td
    gen byte fif_deces=1
    replace comnais=pnais if mi(comnais) & !mi(pnais) //pour les pères nés à
l'étranger
    ren comdom fif_comdomm
    ren comnais fif_codnaism
    keep if !mi(ddnpm,fif_codnaism,fif_comdomm,dddp)
    duplicates drop ddnpm fif_codnaism fif_comdomm,force
    keep annee ddnpm dddp fif_codnaism fif_comdomm fif_deces
    save `d`i'',replace
}
use `d2011',clear
append using `d2012' `d2013' `d2014' `d2015' `d2016' `d2017' `d2018' `d2019'
save `deces',replace
restore
merge m:1 annee ddnpm fif_codnaism fif_comdomm using `deces',keep(1 3)
keepusing(fif_deces dddp) nogenerate
replace fif_deces=0 if fif_deces==. & !mi(fif_type_fisc)
drop ddnpm fif_codnaism fif_comdomm
la var fim_deces "Evenement décès de la mère"
la var fif_deces "Evenement décès du père"
la var dddm "Date de décès de la mère"
la var dddp "Date de décès du père"
la var fim_comdom "Commune de résidence de la mère"
la var fif_comdom "Commune de résidence du père"
compress
```

```
save "$wd\be2019.dta",replace
```

Effectifs et comparaisons

On peut dresser les effectifs de la variable *situ*, et plus particulièrement observer le contraste entre certaines proportions de situations familiales, pour les enfants nés avant juillet 2010, et ceux après :

```
1. gen byte post2010=(anaisi+mnaisi/12>2010.5) if !mi(anaisi,mnaisi)
2. la var post2010 "Né à partir de juillet 2010"
3. ta situ post2010,m
```

L'exécution de la ligne 3 fait apparaître le tableau suivant :

Tableau 13 - Effectifs et proportions de la variable situ, après correction de l'Ordrefip

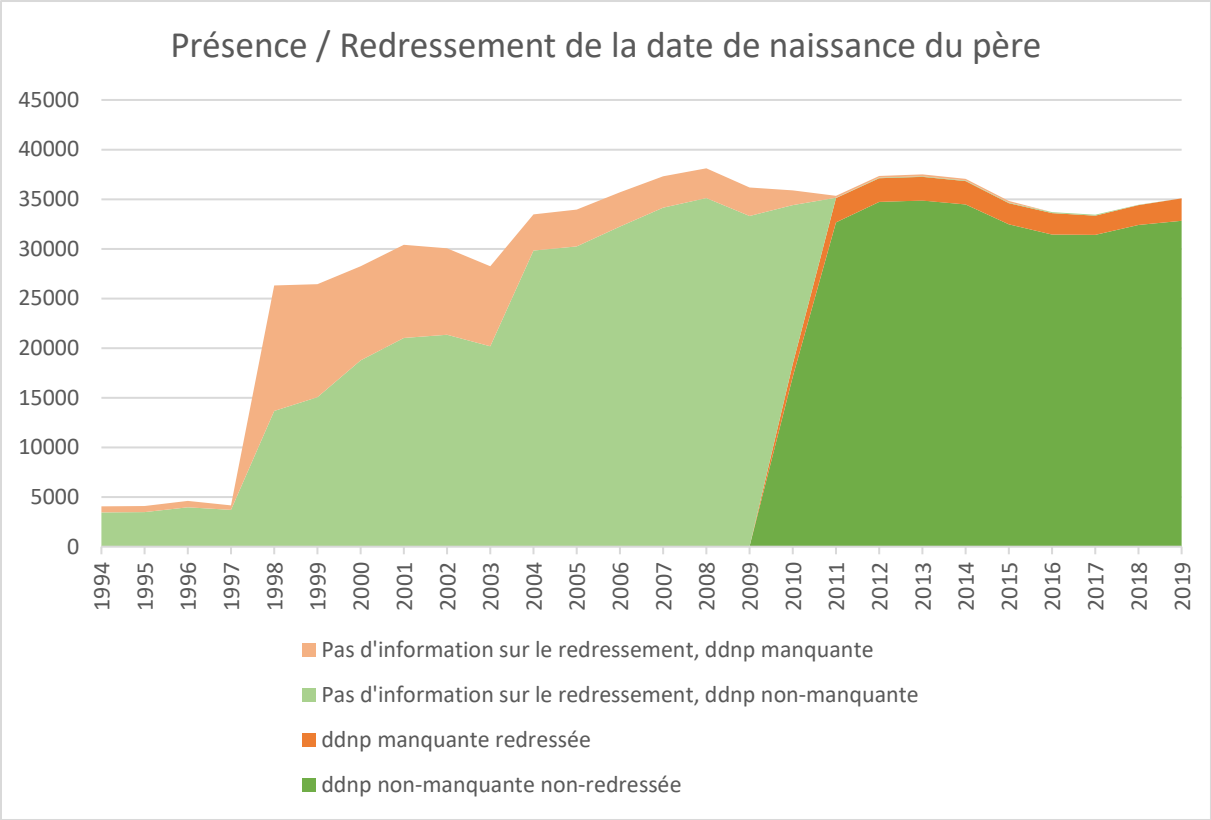
	Naissance de l'enfant EDP						
	Avant Juillet 2010			A partir de Juillet 2010			
Situ	Effectifs	Proportions valides	Proportions	Effectifs	Proportions valides	Proportions	Total
. Manquant	152 234		2,73%	91 340		2,98%	243 574
.a L'enfant n'est pas mineur	1 369 475		24,56%	0		0,00%	1 369 475
.d L'enfant est décédé	15 418		0,28%	5 789		0,19%	21 207
.u L'enfant n'est pas encore né	0		0,00%	1 260 116		41,07%	1 260 116
.f L'enfant est déclarant fiscal	550		0,01%	0		0,00%	550
.m Les DSF de l'enfant sont manquantes	1 330 368		23,86%	498 341		16,24%	1 828 709
.l Le logement de l'enfant est manquant	62 537		1,12%	49 967		1,63%	112 504
.p La date de naissance du père est manquante	23 133		0,41%	52 894		1,72%	76 027
.q La date de naissance de la mère est manquante	66 674		1,20%	809		0,03%	67 483
.r Pas de bulletin de naissance	12 548		0,23%	369		0,01%	12 917
1 Vit avec les 2 parents de naissance mariés dans le même foyer	1 514 867	54,24%	27,17%	547 104	47,50%	17,83%	2 061 971
2 Vit avec les 2 parents de naissance pacsés dans le même foyer	131 427	4,71%	2,36%	175 099	15,20%	5,71%	306 526
3 Vit avec les 2 parents de naissance en UL dans le même logement	255 053	9,13%	4,57%	226 486	19,67%	7,38%	481 539
110 Déclaré en RA par les 2 PDN séparés seuls dans 2 foyers distincts	7 924	0,28%	0,14%	5 232	0,45%	0,17%	13 156
120 Déclaré en RA par les 2 PDN séparés dans 2 foyers distincts, père seul et mère en couple	3 268	0,12%	0,06%	1 190	0,10%	0,04%	4 458
130 Déclaré en RA par les 2 PDN séparés dans 2 foyers distincts, père en couple et mère seule	2 140	0,08%	0,04%	1 133	0,10%	0,04%	3 273
140 Déclaré en RA par les 2 PDN séparés en couple dans 2 foyers distincts	2 361	0,08%	0,04%	561	0,05%	0,02%	2 922
150 Déclaré en RA par la mère dans un foyer, en charge principale par le père dans l'autre, mère et père seuls	643	0,02%	0,01%	552	0,05%	0,02%	1 195
151 Déclaré en RA par la mère dans un foyer, en charge principale par le père dans l'autre, mère en couple et père seul	345	0,01%	0,01%	141	0,01%	0,00%	486
152 Déclaré en RA par la mère dans un foyer, en charge principale par le père dans l'autre, mère seule et père en couple	225	0,01%	0,00%	112	0,01%	0,00%	337
153 Déclaré en RA par la mère dans un foyer, en charge principale par le père dans l'autre, mère en couple et père en couple	238	0,01%	0,00%	49	0,00%	0,00%	287
154 Déclaré en RA par la mère dans un foyer, en charge principale par le père dans l'autre, situations conjugales des PDN inconnues	184	0,01%	0,00%	271	0,02%	0,01%	455
160 Déclaré en RA par le père dans un foyer, en charge principale par la mère dans l'autre, mère et père seuls	1 368	0,05%	0,02%	1 751	0,15%	0,06%	3 119
161 Déclaré en RA par le père dans un foyer, en charge principale par la mère dans l'autre, mère en couple et père seul	773	0,03%	0,01%	315	0,03%	0,01%	1 088
162 Déclaré en RA par le père dans un foyer, en charge principale par la mère dans l'autre, mère seule et père en couple	501	0,02%	0,01%	585	0,05%	0,02%	1 086
163 Déclaré en RA par le père dans un foyer, en charge principale par la mère dans l'autre, mère en couple et père en couple	795	0,03%	0,01%	210	0,02%	0,01%	1 005
164 Déclaré en RA par le père dans un foyer, en charge principale par la mère dans l'autre, situations conjugales des PDN inconnues	682	0,02%	0,01%	1 025	0,09%	0,03%	1 707
170 Vit avec ses 2 PDN dans 2 logements distincts, non-déclaré en RA, mère et père seuls	586	0,02%	0,01%	3 855	0,33%	0,13%	4 441

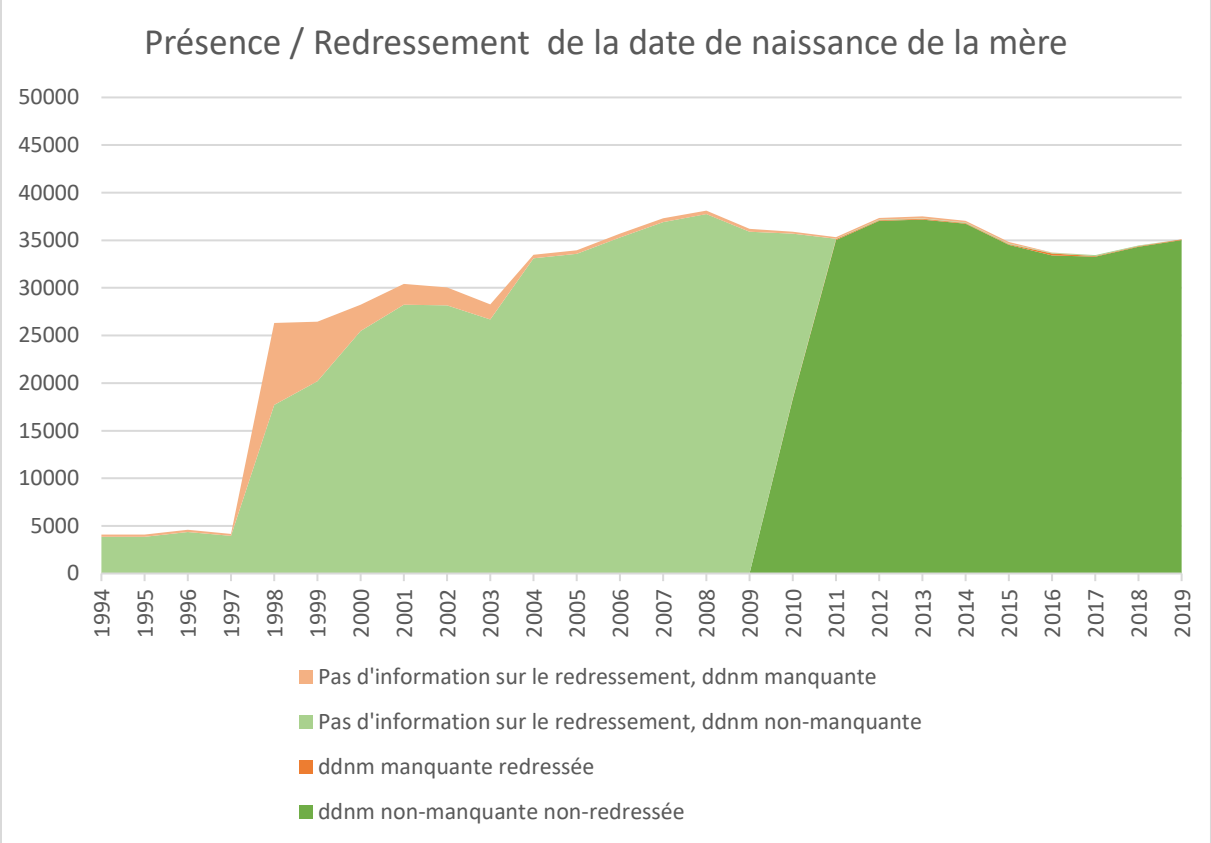
171 Vit avec ses 2 PDN dans 2 logements distincts, non-déclaré en RA, mère en couple et père seul	205	0,01%	0,00%	452	0,04%	0,01%	657
172 Vit avec ses 2 PDN dans 2 logements distincts, non-déclaré en RA, mère seule et père en couple	270	0,01%	0,00%	760	0,07%	0,02%	1 030
173 Vit avec ses 2 PDN dans 2 logements distincts, non-déclaré en RA, mère en couple et père en couple	129	0,00%	0,00%	179	0,02%	0,01%	308
174 Vit avec ses 2 PDN dans 2 logements distincts, non-déclaré en RA, situations conjugales des PDN inconnues	307	0,01%	0,01%	4 300	0,37%	0,14%	4 607
180 La mère déclare en RA et le père ne déclare pas l'enfant, mère seule	1 513	0,05%	0,03%	1 240	0,11%	0,04%	2 753
181 La mère déclare en RA et le père ne déclare pas l'enfant, mère en couple	1 083	0,04%	0,02%	284	0,02%	0,01%	1 367
182 La mère déclare en RA et le père ne déclare pas l'enfant, mère en situation conjugale inconnue	152	0,01%	0,00%	147	0,01%	0,00%	299
190 Le père déclare en RA et la mère ne déclare pas l'enfant, père seul	24 123	0,86%	0,43%	1 653	0,14%	0,05%	25 776
191 Le père déclare en RA et la mère ne déclare pas l'enfant, père en couple	11 551	0,41%	0,21%	443	0,04%	0,01%	11 994
192 Le père déclare en RA et la mère ne déclare pas l'enfant, père en situation conjugale inconnue	2 774	0,10%	0,05%	428	0,04%	0,01%	3 202
195 La mère déclare en RA et le père est absent du bulletin de naissance	549	0,02%	0,01%	0	0,00%	0,00%	549
196 Le père déclare en RA et la mère est absente du bulletin de naissance	18	0,00%	0,00%	0	0,00%	0,00%	18
200 L'enfant est déclaré en RA mais n'a pas de bulletin de naissance	420	0,02%	0,01%	0	0,00%	0,00%	420
210 L'enfant est déclaré en RA mais un seul logement identifié	9 580	0,34%	0,17%	155	0,01%	0,01%	9 735
220 Autres situations de RA	2 949	0,11%	0,05%	2 250	0,20%	0,07%	5 199
310 Vit avec sa mère seulement, seule	59 781	2,14%	1,07%	79 521	6,90%	2,59%	139 302
320 Vit avec sa mère seulement, en couple	51 865	1,86%	0,93%	17 205	1,49%	0,56%	69 070
330 Vit avec sa mère seulement, seule, père absent du bulletin de naissance	79 146	2,83%	1,42%	4	0,00%	0,00%	79 150
340 Vit avec sa mère seulement, en couple, père absent du bulletin de naissance	84 766	3,04%	1,52%	47	0,00%	0,00%	84 813
350 Vit avec son père seulement, seul	40 667	1,46%	0,73%	17 592	1,53%	0,57%	58 259
360 Vit avec son père seulement, en couple	31 601	1,13%	0,57%	11 279	0,98%	0,37%	42 880
400 Ne vit ni avec son père ni avec sa mère	397	0,01%	0,01%	92	0,01%	0,00%	489
510 Pas de bulletin de naissance, femme monoparentale	17 282	0,62%	0,31%	135	0,01%	0,00%	17 417
511 Pas de bulletin de naissance, homme monoparental	8 920	0,32%	0,16%	159	0,01%	0,01%	9 079
520 Pas de bulletin de naissance, couple avec enfant(s)	190 161	6,81%	3,41%	4 740	0,41%	0,15%	194 901
Total non-manquant	2 543 589	91,08%	45,61%	1 108 736	96,27%	36,13%	3 652 325
Total manquant	3 032 937		54,39%	1 959 625		63,87%	4 992 562
Total	5 576 526		100,00%	3 068 361		100,00%	8 644 887

On note l'absence d'enfants majeurs (situ=.a) pour les enfants nés à partir de Juillet 2010. En effet, les années fiscales d'observation étant de 2011 à 2019, les enfants ont au plus 9 ans sur la fenêtre d'observation. Similairement, on n'observe aucun enfant « pas encore né » (situ=.u) chez les enfants nés avant Juillet 2010 : la fenêtre d'observation commençant en 2011, tous les enfants nés avant ont déjà au moins quelques mois d'âge. A l'inverse, les enfants nés, par exemple, en 2012, ne sont pas encore nés lorsqu'ils sont « observés » fiscalement en 2011. L'observation fiscale, pour les enfants pas encore nés, peut être utile par exemple pour étudier les situations des (futurs) parents qui sont EDP, dans les années précédant la naissance de l'enfant EDP.

On note par ailleurs qu'à partir de Juillet 2010, la date de naissance des parents est toujours bien davantage disponible (situ=330, 340) qu'avant, de même que le bulletin de naissance en lui-même (situ=.r, 510, 511, 520). Ceci demeure difficile à expliquer compte tenu du fait que nous avons neutralisé le redressement des informations de naissance sur les bulletins à partir de Juillet 2010.

Sur ce sujet, on vérifie la variable de redressement des dates de naissance du père et de la mère, par année de naissance :





On dresse par la suite un tableau des effectifs de type d'union :

Tableau 14 - Effectifs et proportions de type d'union

	Effectifs	Proportions valides
1 f1 est MPFH ; pas de f2	3 304 883	80,86807%
2 f1 est ULFH ; pas de f2	705 026	17,25147%
3 f1 est MPFF ; pas de f2	1 512	0,03700%
4 f1 est ULFF ; pas de f2	1 345	0,03291%
5 f1 est MPHH ; pas de f2	456	0,01116%
6 f1 est ULHH ; pas de f2	301	0,00737%
101 f1 est MPFH ; f2 est seul	3 691	0,09032%
102 f1 est ULFH ; f2 est seul	4 826	0,11809%
103 f1 est MPFF ; f2 est seul	36	0,00088%
104 f1 est ULFF ; f2 est seul	51	0,00125%
105 f1 est MPHH ; f2 est seul	19	0,00046%
106 f1 est ULHH ; f2 est seul	47	0,00115%
201 f1 est seul ; f2 est MPFH	4 550	0,11134%
202 f1 est seul ; f2 est ULFH	4 581	0,11209%
203 f1 est seul ; f2 est MPFF	36	0,00088%
204 f1 est seul ; f2 est ULFF	47	0,00115%
205 f1 est seul ; f2 est MPHH	15	0,00037%
206 f1 est seul ; f2 est ULHH	24	0,00059%
301 f1 est MPFH ; f2 est MPFH	28 363	0,69402%
302 f1 est MPFH ; f2 est ULFH	1 333	0,03262%
303 f1 est MPFH ; f2 est MPFF	11	0,00027%
304 f1 est MPFH ; f2 est ULFF	8	0,00020%
305 f1 est MPFH ; f2 est MPHH	9	0,00022%
306 f1 est MPFH ; f2 est ULHH	2	0,00005%
307 f1 est ULFH ; f2 est MPFH	6 360	0,15562%
308 f1 est ULFH ; f2 est ULFH	19 055	0,46626%
309 f1 est ULFH ; f2 est MPFF	7	0,00017%
310 f1 est ULFH ; f2 est ULFF	15	0,00037%
311 f1 est ULFH ; f2 est MPHH	4	0,00010%
312 f1 est ULFH ; f2 est ULHH	7	0,00017%
313 f1 est MPFF ; f2 est MPFH	11	0,00027%
314 f1 est MPFF ; f2 est ULFH	5	0,00012%
315 f1 est MPFF ; f2 est MPFF	20	0,00049%
316 f1 est MPFF ; f2 est ULFF	0	0,00000%
317 f1 est MPFF ; f2 est MPHH	5	0,00012%
318 f1 est MPFF ; f2 est ULHH	0	0,00000%
319 f1 est ULFF ; f2 est MPFH	11	0,00027%
320 f1 est ULFF ; f2 est ULFH	8	0,00020%
321 f1 est ULFF ; f2 est MPFF	3	0,00007%
322 f1 est ULFF ; f2 est ULFF	23	0,00056%
323 f1 est ULFF ; f2 est MPHH	0	0,00000%
324 f1 est ULFF ; f2 est ULHH	0	0,00000%
325 f1 est MPHH ; f2 est MPFH	4	0,00010%

326 f1 est MPHH ; f2 est ULFH	7	0,00017%
327 f1 est MPHH ; f2 est MPFF	4	0,00010%
328 f1 est MPHH ; f2 est ULFF	0	0,00000%
329 f1 est MPHH ; f2 est MPHH	6	0,00015%
330 f1 est MPHH ; f2 est ULHH	0	0,00000%
331 f1 est ULHH ; f2 est MPFH	1	0,00002%
332 f1 est ULHH ; f2 est ULFH	4	0,00010%
333 f1 est ULHH ; f2 est MPFF	0	0,00000%
334 f1 est ULHH ; f2 est ULFF	0	0,00000%
335 f1 est ULHH ; f2 est MPHH	2	0,00005%
336 f1 est ULHH ; f2 est ULHH	25	0,00061%
Total non-manquant	4 086 759	100,00000%
Manquants	4 558 128	
Total	8 644 887	

A partir de la variable *fhp* (familles homoparentales), on retrouve les effectifs et proportions suivantes :

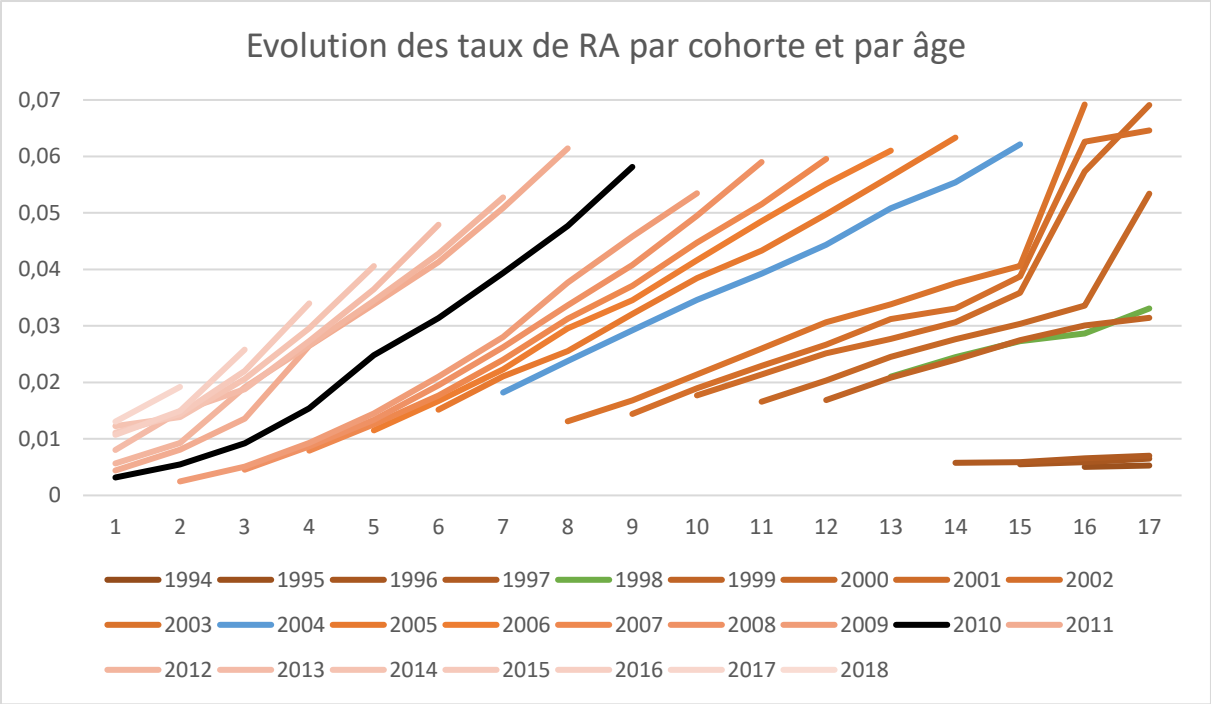
Tableau 15 - Effectifs et proportions de familles homoparentales

	Effectifs	**Proportions valides**
0 FH	4 082 668	99,89990%
1 FF	3 149	0,07705%
2 HH	933	0,02283%
3 FF/HH	9	0,00022%
Total non-manquants	4 086 759	100,00000%
Manquants	4 558 128	
Total	8 644 887	

On peut recoder la variable *situ* pour définir une indicatrice de résidence alternée :

```
1. recode situ (1 2 3 300/max=0 "Non-RA")(100/220=1 "RA"),gen(ra)
2. la var ra "Résidence alternée"
3. table age_t anaisi if age_t<18,contents(mean ra)
4. save "$wd\be2019.dta", replace
```

L'exécution de la ligne 3 fait apparaître des taux de résidence alternée par âge de l'enfant et par cohorte de naissance. Nous en dressons la figure :



Nous observons trois grandes irrégularités :

- Les cohortes 1994-1997 ont des taux exceptionnellement bas de résidence alternée. On rappelle que ces cohortes se distinguent des suivantes par le nombre de jours EDP permettant l'inclusion du bulletin de naissance, dont nous avons besoin pour construire la variable : 2 jours EDP avant 1998, 4 jours EDP à partir de 1998.
- Les cohortes 1998-2003 ont des taux également bas, comparés aux cohortes plus récentes. Elles se distinguent des cohortes précédentes là aussi par le nombre de jours EDP couverts : 2 jours EDP avant 1998, 4 jours EDP de 1998 à 2003, 16 jours EDP à partir de 2004
- Les cohortes 2004-2010 et 2010-2018 se distinguent les unes des autres par des taux plus élevés dans ces dernières. Il est possible que l'imputation des informations de naissance des enfants EDP à partir de Juillet 2010 permette une meilleure remontée des informations concernant les résidences alternées, bien que le mécanisme exact ne soit pas clair.

Concernant les décès de parents de naissance, les variables construites *fim_deces* et *fif_deces* valent 1 l'année où un appariement de la mère ou du père avec la table des décès est possible (sur la clé d'appariement : année fiscale, date de naissance du parent, commune de naissance du parent, commune de résidence du parent). On dénombre ainsi les évènements de décès suivants (dans les couples de parents) :

Année fiscale	Mères identifiées fiscalement	Décès observés de mères cette année à l'INSEE	Pères identifiés fiscalement	Décès observés de pères cette année à l'INSEE
2011	257030	102	257030	232
2012	275774	93	275774	242
2013	294703	102	294703	263
2014	312352	117	312351	273
2015	334041	119	334039	292
2016	340278	130	340276	298

2017	343029	141	343031	301
2018	345802	137	345803	317
2019	346988	128	346985	260

Les variables *fim_zoxyzd fim_dacoed* et *fif_zoxyzd fif_dacoed* donnent respectivement le code et la date du dernier évènement de la mère et du père. On extrait, pour la mère et le père, les années du dernier évènement lorsque celui-ci était le décès d'un conjoint, et lorsque la mère et le père étaient en couple l'année d'avant :

	Mères identifiées fiscalement	Conjoints de mères décédés	Pères identifiés fiscalement	Conjointes de pères décédées
2011	246810	140	247492	97
2012	264947	137	265394	77
2013	283184	123	284429	71
2014	301081	71	301573	92
2015	322113	74	322031	111
2016	323428	202	322422	112
2017	329962	260	328869	132
2018	322048	244	330555	118

Les variables *fim_cideci* et *fif_cideci* peuvent valoir « V » lorsque la mère ou le père est veuf. En vérifiant chez une mère ou un père veuf qu'ils étaient en couple et n'étaient pas veufs l'année d'avant, on peut créer une identificatrice de décès du conjoint, dont on distribue par année les effectifs :

	Mères identifiées fiscalement	Conjoints de mères décédés	Pères identifiés fiscalement	Conjointes de pères décédées
2012	257030	134	257030	87
2013	275774	133	275774	68
2014	294703	120	294703	65
2015	312352	76	312352	89
2016	334047	68	334047	103
2017	340284	200	340284	105
2018	343039	260	343039	130
2019	345812	237	345812	108

Séparations

Construction de la variable *sep*

Ici, nous construisons la variable de séparation, qui permet d'identifier un évènement une année *t* où les parents de naissance de l'enfant EDP se séparent. Plus formellement, elle est définie de façon suivante :

Elle vaut 1 :

- Lorsque l'enfant EDP mineur ne vit pas avec ses deux parents de naissance dans le même logement en t, alors qu'il vivait avec ses deux parents de naissance en t-1

Elle vaut 0 :

- Lorsque l'enfant EDP vit avec ses deux parents de naissance dans le même logement
- Lorsque l'enfant EDP a connu une séparation de ses parents de naissance en t, puis une remise en couple en t+n (n>0) de ses parents de naissance entre eux (on considère que la séparation était une erreur)
- Lorsque l'enfant EDP ne vit plus avec ses deux parents de naissance dans le même logement en t, alors qu'il vivait avec ses deux parents de naissance en t-1, mais qu'il ne vivait pas avec ses deux parents de naissance en t-n (n>1).
- Lorsque le père ou la mère de l'enfant EDP est décédé en t, t-1 ou t-2

Elle est manquante :

- Si l'enfant ne vit pas avec ses parents de naissance en t, mais qu'il ne vivait pas non plus avec eux en t-1

```
*Création de la variable de séparation
cap drop sep
gen byte sep=(situ>100 & L.situ<100) if situ<. & L.situ<. & age_t<18
//variable de séparation
//on laisse manquant si les PDN n'étaient pas ensemble en t-1 :
replace sep=. if sep==0 & situ>100 & situ<.

//on considère qu'il n'y a pas séparation si remise en couple ultérieure des
PDN :
replace sep=0 if sep==1 & F.situ<100 replace sep=0 if sep==1 & F2.situ<100
replace sep=0 if sep==1 & F3.situ<100
replace sep=0 if sep==1 & F4.situ<100
replace sep=0 if sep==1 & F5.situ<100
replace sep=0 if sep==1 & F6.situ<100
replace sep=0 if sep==1 & F7.situ<100
//on considère qu'il n'y a pas séparation si PDN pas en couple en t-n :
replace sep=0 if sep==1 & L2.situ>100 & L2.situ<.
replace sep=0 if sep==1 & L3.situ>100 & L3.situ<.
replace sep=0 if sep==1 & L4.situ>100 & L4.situ<.
replace sep=0 if sep==1 & L5.situ>100 & L5.situ<.
replace sep=0 if sep==1 & L6.situ>100 & L6.situ<.
replace sep=0 if sep==1 & L7.situ>100 & L7.situ<.
```

```
replace sep=0 if sep==1 & L8.situ>100 & L8.situ<.
replace sep=0 if sep==1 & fim_zoxyzd=="Z" &
real(substr(fim_dacoed,5,4))>=annee-2 & !mi(fim_dacoed) //on considère qu'il
n'y a pas séparation si le conjoint marié/pacsé du parent est décédé
replace sep=0 if sep==1 & fif_zoxyzd=="Z" &
real(substr(fif_dacoed,5,4))>=annee-2 & !mi(fif_dacoed)
replace sep=0 if sep==1 & (L.fim_deces==1 | L.fif_deces==1 | L2.fim_deces==1 |
L2.fif_deces==1 | fim_deces==1 | fif_deces==1) //même principe pour les
parents décédés
label define sep 1 "Séparation des PDN" 0 "Pas de séparation des PDN",replace
label values sep sep
la var sep "Séparation, si celle-ci a eu lieu pdt la minorité"
```

Construction de la distance à la séparation *deltasep*

Une fois la variable dichotomique *sep* créée, on s'appuie dessus pour créer une variable de distance, en années, à la séparation (celle-ci étant marquée par sep=1).

```
*On construit une variable de distance à la séparation:
cap drop deltasep
gen int deltasep=.
replace deltasep=-8 if F8.sep==1 & F8.age_t<18
replace deltasep=-7 if F7.sep==1 & F7.age_t<18
replace deltasep=-6 if F6.sep==1 & F6.age_t<18
replace deltasep=-5 if F5.sep==1 & F5.age_t<18
replace deltasep=-4 if F4.sep==1 & F4.age_t<18
replace deltasep=-3 if F3.sep==1 & F3.age_t<18
replace deltasep=-2 if F2.sep==1 & F2.age_t<18
replace deltasep=-1 if F.sep==1 & F.age_t<18
replace deltasep=0 if sep==1 & age_t<18
replace deltasep=1 if L.sep==1 & L.age_t<18
replace deltasep=2 if L2.sep==1 & L2.age_t<18
replace deltasep=3 if L3.sep==1 & L3.age_t<18
replace deltasep=4 if L4.sep==1 & L4.age_t<18
replace deltasep=5 if L5.sep==1 & L5.age_t<18
replace deltasep=6 if L6.sep==1 & L6.age_t<18
replace deltasep=7 if L7.sep==1 & L7.age_t<18
la var deltasep "Distance à la séparation, si celle-ci a eu lieu pdt la
minorité"
```

Situation familiale à la séparation *situsep*

On construit par la suite une variable *situ_parent* permettant d'identifier la situation familiale avant et après séparation. En cas de séparation, celle-ci vaudra 2 s'il y a une résidence alternée, 3 si l'enfant se retrouve avec sa mère de naissance uniquement, 4 si l'enfant se retrouve avec son père de naissance uniquement, 5 dans les autres cas ; et elle vaudra 1 si les parents de naissance de l'enfant ne se sont pas encore séparés.

On fixe cette variable le long du panel à sa valeur l'année de la séparation. La variable qui en résulte s'appelle *situsep*.

```
*Situation familiale à la séparation:
gen byte situ_parent=1 if inlist(situ,1,2,3) | deltasep<0
replace situ_parent=2 if inrange(situ,110,220) & deltasep>=0
replace situ_parent=3 if inrange(situ,310,340) & deltasep>=0
replace situ_parent=4 if inlist(situ,350,360) & deltasep>=0
replace situ_parent=5 if inlist(situ,400,510,511,520,.r) & deltasep>=0
label define situ_parentf 1 "2 parents de naissance" 2 "RA" 3 "mère" 4 "père"
5 "autre, pas def.", replace
la var situ_parent "Mode de résidence de l'enfant"
gen byte situ_ap=sep*situ_parent
bys idi: egen byte situsep=max(situ_ap)
drop situ_ap
la var situsep "Mode de résidence de l'enfant à la séparation"
label values situ_parent situsep situ_parentf
compress
save "$wd\be2019.dta",replace
```

Veuvage

On considère à présent la construction de variables de veuvage et de distance au veuvage sur le même modèle que *sep* et *deltasep*. On s'appuie pour cela sur plusieurs variables, déjà utilisées dans la construction de *sep* pour exclure les veuvages du champ des séparations : *fim_zoxyzd, fif_zoxyzd, fim_deces* et *fif_deces*. On considèrera qu'un veuvage ne survient que lors d'une « séparation » des parents de naissance (PDN), et lorsque :

- le code du dernier évènement (*zoxzd*) dans les 2 années conduisant à la « séparation » est le décès du conjoint du père ou de la mère ; ou lorsque :
- on observe un décès de la mère ou du père dans la base exhaustive des décès, l'année *t* de la « séparation », l'année *t-1* ou l'année *t-2*

```
*On construit une variable de veuvage:
cap drop veuvage
gen byte veuvage=(situ>100 & L.situ<100) if age_t<18
replace veuvage=. if veuvage==0 & situ>100 //on laisse manquant si les PDN
n'étaient pas ensemble en t-1
replace veuvage=0 if veuvage==1 & (fim_zoxyzd!="Z" |
real(substr(fim_dacoed,5,4))<annee-2) & (fif_zoxyzd!="Z" |
real(substr(fif_dacoed,5,4))<annee-2) & fim_deces!=1 & L.fim_deces!=1 &
L2.fim_deces!=1 & fif_deces!=1 & L.fif_deces!=1 & L2.fif_deces!=1
replace veuvage=0 if veuvage==1 & L.veuvage==1 //on restreint l'occurrence
multiple de veuvages
replace veuvage=0 if veuvage==1 & L2.veuvage==1
replace veuvage=0 if veuvage==1 & L3.veuvage==1
replace veuvage=0 if veuvage==1 & L4.veuvage==1
replace veuvage=0 if veuvage==1 & L5.veuvage==1
replace veuvage=0 if veuvage==1 & L6.veuvage==1
replace veuvage=0 if veuvage==1 & L7.veuvage==1
replace veuvage=0 if veuvage==1 & L8.veuvage==1
```

```
replace veuvage=0 if veuvage==1 & F.situ<100 //on considère qu'il n'y a pas de
veuvage si remise en couple ultérieure des PDN
replace veuvage=0 if veuvage==1 & F2.situ<100
replace veuvage=0 if veuvage==1 & F3.situ<100
replace veuvage=0 if veuvage==1 & F4.situ<100
replace veuvage=0 if veuvage==1 & F5.situ<100
replace veuvage=0 if veuvage==1 & F6.situ<100
replace veuvage=0 if veuvage==1 & F7.situ<100
la var veuvage "Veuvage"

*On construit une variable de distance au veuvage:
cap drop deltaveuvage
gen int deltaveuvage=.
replace deltaveuvage=-8 if F8.veuvage==1 & F8.age_t<18
replace deltaveuvage=-7 if F7.veuvage==1 & F7.age_t<18
replace deltaveuvage=-6 if F6.veuvage==1 & F6.age_t<18
replace deltaveuvage=-5 if F5.veuvage==1 & F5.age_t<18
replace deltaveuvage=-4 if F4.veuvage==1 & F4.age_t<18
replace deltaveuvage=-3 if F3.veuvage==1 & F3.age_t<18
replace deltaveuvage=-2 if F2.veuvage==1 & F2.age_t<18
replace deltaveuvage=-1 if F.veuvage==1 & F.age_t<18
replace deltaveuvage=0 if veuvage==1 & age_t<18
replace deltaveuvage=1 if L.veuvage==1 & L.age_t<18
replace deltaveuvage=2 if L2.veuvage==1 & L2.age_t<18
replace deltaveuvage=3 if L3.veuvage==1 & L3.age_t<18
replace deltaveuvage=4 if L4.veuvage==1 & L4.age_t<18
replace deltaveuvage=5 if L5.veuvage==1 & L5.age_t<18
replace deltaveuvage=6 if L6.veuvage==1 & L6.age_t<18
replace deltaveuvage=7 if L7.veuvage==1 & L7.age_t<18
la var deltaveuvage "Distance au veuvage"
```

Tous les décès de parents collectés dans la base exhaustive INSEE ne correspondent pas nécessairement à des veuvages de parents, caractérisés par une rupture de l'union familiale.

```
bys id_diff: egen byte mveuvage=max(veuvage)
bys id_diff: egen byte mfim_deces=max(fim_deces)
bys id_diff: egen byte mfif_deces=max(fif_deces)
unique id_diff if mveuvage==1 & (mfim_deces==1 | mfif_deces==1)
unique id_diff if mveuvage==0 & (mfim_deces==1 | mfif_deces==1)
```

Pondération

Pondération fiscale

La pondération fiscale de l'enfant est calculée à partir de la variable *poids_fideli*. D'après la documentation de l'EDP, cette pondération est « calée sur les marges de Fidéli de l'année fiscale considérée pour les individus EDP dont TYPE_PRES = 1 ». Or, par construction, le 2e foyer fiscal de l'enfant a une valeur pour *type_pres* jamais égale à 1 (cf. ligne 4, sous-section « Reformation du panel », page 13). La variable *fi2_poids_fideli* vaut donc toujours 0 lorsque l'enfant est déclaré dans un 2e foyer fiscal, et est manquante sinon. Nous nous appuyons donc systématiquement sur la variable *fi1_poids_fideli* pour obtenir le poids de l'enfant :

```
*Pondération:
gen long poids=round(fi1_poids_fideli)
la var poids "Pondération Fideli"
save "$wd\be2019.dta",replace
```

La pondération fiscale des parents de naissance a été importée dans la base depuis la section « Appariement des informations fiscales des parents de naissance » et se présente sous les variables *fim_poids_fideli* et *fif_poids_fideli*, pour respectivement, le poids de la mère et le poids du père.

La pondération fiscale des déclarants fiscaux (qu'il s'agisse des parents de naissance ou non), a été importée depuis la sous-section « Informations fiscales relatives aux déclarants de l'enfant » et se présente sous les variables *fi1d_poids_fideli*, *fi1p_poids_fideli*, *fi2d_poids_fideli* et *fi2p_poids_fideli*, pour respectivement, les poids du déclarant fiscal du 1er foyer et son/sa partenaire, et du déclarant fiscal du 2e foyer et son/sa partenaire.

Pondération de recensement

La pondération des personnes de référence du ménage a été importée dans la base depuis la sous-section « Injection de variables d'EAR (2011-2019) dans la base ». Il s'agit, pour la personne de référence du ménage, des variables *ear1_poids_ea, ear1_poids_ea_cale, ear1_poids_ea_cale_oct, ear1_poids_panel_5,* et *ear1_poids_panel_10*. Pour le conjoint de la personne de référence du ménage, il s'agit des variables *ear2_poids_ea, ear2_poids_ea_cale, ear2_poids_ea_cale_oct, ear2_poids_panel_5,* et *ear2_poids_panel_10*.

Variables économiques

Indices des prix

Afin de pouvoir déflater les niveaux économiques, on commence par introduire pour chaque année, l'indice des prix de l'année considérée, telle que reportée par Eurostat en € de 2019 :

```
gen double idprix=0.9166 if annee==2011
replace idprix=0.9369 if annee==2012
replace idprix=0.9463 if annee==2013
replace idprix=0.9520 if annee==2014
replace idprix=0.9528 if annee==2015
replace idprix=0.9558 if annee==2016
replace idprix=0.9668 if annee==2017
replace idprix=0.9871 if annee==2018
replace idprix=1.0000 if annee==2019
la var idprix "Indice des prix à la consommation base 2019 (Eurostat)"
```

Indices de pauvreté

On crée l'indice de pauvreté de l'enfant à 60% de la médiane, sous condition à nouveau d'appartenance au champ de diffusion Filosofi :

```
egen idp60=rowmean(fr1_i_pauvre60m fr2_i_pauvre60m) if (fr1_i_champm==1 |
fr2_i_champm==1)
la var idp60 "Indice de pauvreté moyen de l'enfant à 60%"
```

Niveaux de vie

On calcule par la suite le niveau de vie de l'enfant, déflaté, et moyenné sur l'ensemble de ses foyers fiscaux, sous condition qu'il appartienne au champ de diffusion *filosofi* (variable *i_champm*) :

```
gen double fr1_nivviemcst=fr1_nivviem/idprix
order fr1_nivviemcst,after(fr1_nivviem)
la var fr1_nivviemcst "Ndv du foyer 1 en € constants"
gen double fr2_nivviemcst=fr2_nivviem/idprix
order fr2_nivviemcst,after(fr2_nivviem)
la var fr2_nivviemcst "Ndv du foyer 2 en € constants"

egen double ndv=rowmean(fr1_nivviemcst fr2_nivviemcst) if (fr1_i_champm==1 |
fr2_i_champm==1)
la var ndv "Niveau de vie moyen de l'enfant (€ 2019)"
```

Lorsque les parents de naissance sont séparés, il peut être intéressant de ne suivre que le niveau de vie de l'enfant relatif au parent de naissance qui a conservé la garde, ou de faire une moyenne des niveaux de vie des deux foyers où l'enfant est déclaré en résidence alternée. On note *ndvsep* une telle variable :

```
//on identifie le niveau de vie comme la moyenne de celles des 2 logements où
l'enfant est en RA
```

```
egen double ndvsep=rowmean(fr1_nivviemcst fr2_nivviemcst) if situ_parent==2 &
(fr1_i_champm==1 | fr2_i_champm==1)
//on identifie le ndv dans le logement où l'enfant est en garde principale
replace ndvsep=fr1_nivviemcst if log1==1 & situ_parent==3 & fr1_i_champm==1
replace ndvsep=fr2_nivviemcst if log2==1 & situ_parent==3 & fr2_i_champm==1
replace ndvsep=fr1_nivviemcst if log1==2 & situ_parent==4 & fr1_i_champm==1
replace ndvsep=fr2_nivviemcst if log2==2 & situ_parent==4 & fr2_i_champm==1
//on identifie le niveau de vie dans le logement principal qd les parents ne
sont pas séparés:
replace ndvsep=ndv if log1==3 & situ_parent==1 & (fr1_i_champm==1 |
fr2_i_champm==1)
la var ndvsep "Niveau de vie de l'enfant dont les parents se séparent (en
€2019)"
```

Nombre d'unités de consommation

On s'intéresse à capturer le nombre d'unités de consommation dans le ménage où l'enfant vit avec son parent de naissance après séparation. On note *nbucsep* cette variable :

```
*Pour voir l'évolution du nb d'uc autour de la séparation :
egen double nbucsep=rowmean(fr1_nb_uc fr2_nb_uc) if situ_parent==2
replace nbucsep=fr1_nb_uc if log1==1 & situ_parent==3
replace nbucsep=fr2_nb_uc if log2==1 & situ_parent==3
replace nbucsep=fr1_nb_uc if log1==2 & situ_parent==4
replace nbucsep=fr2_nb_uc if log2==2 & situ_parent==4
replace nbucsep=fr1_nb_uc if log1==3 & situ_parent==1
la var nbucsep "Nombre d'unités de consommation, dans les ménages où les
parents se séparent"
```

Revenus

On déflate les revenus du ménage et les revenus détaillés individuels :

```
*Revenus: Prise en compte de l'inflation:
foreach v of varlist fr1_revdecm fr2_revdecm fr1_produitfin fr2_produitfin
fr1_psocm fr2_psocm fr1_m_ppem fr2_m_ppem fr1_zimpom fr2_zimpom fr1_zthabm
fr2_zthabm fr1_zimpvalm fr2_zimpvalm fr1_csgim fr2_csgim fr1_crdsm fr2_crdsm
fr1_csgpatm fr2_csgpatm fr1_csgvalm fr2_csgvalm fr1_csgimpm fr2_csgimpm
fr1_ztsam fr2_ztsam fr1_zperm fr2_zperm fr1_zragm fr2_zragm fr1_zricm
fr2_zricm fr1_zrncm fr2_zrncm fr1_zfonm fr2_zfonm fr1_zvamm fr2_zvamm
fr1_zvalm fr2_zvalm fr1_zracm fr2_zracm fr1_zetrm fr2_zetrm fr1_zalvm
fr2_zalvm fr1_pfamm fr2_pfamm fr1_minim fr2_minim fr1_logtm fr2_logtm
fr1_csgisalm fr2_csgisalm fr1_zalrm fr2_zalrm fr1_zsalm fr2_zsalm fr1_zchom
fr2_zchom { //Revenus du ménage
    gen double `v'cst=`v'/idprix
}
foreach v of varlist fi1drd_ysali fi2drd_ysali fi1prd_ysali fi2prd_ysali
fi1drd_ychoi fi2drd_ychoi fi1prd_ychoi fi2prd_ychoi  fi1drd_yalri fi2drd_yalri
fi1prd_yalri fi2prd_yalri { //Revenus détaillés individuels
    gen double `v'cst=`v'/idprix
```

```
}
```

On s'intéresse plus particulièrement à tracer les revenus des enfants chez leur père ou mère de naissance, autour de la séparation :

```
*Traçage des revenus selon le mode de garde:
set more off
foreach v in revdecm produitfin psocm m_ppem zimpom zthabm zimpvalm csgim
crdsm csgpatm csgvalm csgimpm ztsam zperm zragm zricm zrncm zfonm zvamm zvalm
zracm zetrm zalvm pfamm minim logtm csgisalm zalrm zsalm zchom {
    egen double `v'_sep=rowmean(fr1_`v'cst fr2_`v'cst) if situ_parent==2 &
(fr1_i_champm==1 | fr2_i_champm==1)
    replace `v'_sep=fr1_`v'cst if log1==1 & situ_parent==3 & fr1_i_champm==1
    replace `v'_sep=fr2_`v'cst if log2==1 & situ_parent==3 & fr2_i_champm==1
    replace `v'_sep=fr1_`v'cst if log1==2 & situ_parent==4 & fr1_i_champm==1
    replace `v'_sep=fr2_`v'cst if log2==2 & situ_parent==4 & fr2_i_champm==1
    replace `v'_sep=fr1_`v'cst if log1==3 & situ_parent==1 & fr1_i_champm==1
}
foreach v in ysali ychoi yalri {
    egen double `v'd_sep=rowmean(fi1drd_`v'cst fi2drd_`v'cst) if
situ_parent==2 & (fr1_i_champm==1 | fr2_i_champm==1)
    replace `v'd_sep=fi1drd_`v'cst if log1==1 & inlist(tuf1,1,2) &
situ_parent==3 & fr1_i_champm==1
    replace `v'd_sep=fi2drd_`v'cst if log2==1 & inlist(tuf2,1,2) &
situ_parent==3 & fr2_i_champm==1
    replace `v'd_sep=fi1drd_`v'cst if log1==2 & inlist(tuf1,1,2) &
situ_parent==4 & fr1_i_champm==1
    replace `v'd_sep=fi2drd_`v'cst if log2==2 & inlist(tuf2,1,2) &
situ_parent==4 & fr2_i_champm==1
    replace `v'd_sep=fi1drd_`v'cst if log1==3 & inlist(tuf1,1,2) &
situ_parent==1 & fr1_i_champm==1
    egen double `v'p_sep=rowmean(fi1prd_`v'cst fi2prd_`v'cst) if
situ_parent==2 & (fr1_i_champm==1 | fr2_i_champm==1)
    replace `v'p_sep=fi1prd_`v'cst if log1==1 & inlist(tuf1,1,2) &
situ_parent==3 & fr1_i_champm==1
    replace `v'p_sep=fi2prd_`v'cst if log2==1 & inlist(tuf2,1,2) &
situ_parent==3 & fr2_i_champm==1
    replace `v'p_sep=fi1prd_`v'cst if log1==2 & inlist(tuf1,1,2) &
situ_parent==4 & fr1_i_champm==1
    replace `v'p_sep=fi2prd_`v'cst if log2==2 & inlist(tuf2,1,2) &
situ_parent==4 & fr2_i_champm==1
    replace `v'p_sep=fi1prd_`v'cst if log1==3 & inlist(tuf1,1,2) &
situ_parent==1 & fr1_i_champm==1
}
```

Toujours pour les ménages dont les parents de naissance se séparent, on calcule les revenus primaires, revenus d'activité, transferts publics, et pensions alimentaires nettes relatives à l'enfant :

$$
\begin{aligned}
Revenus\ primaires &= revdecm + produitfin + m\_ppem - zalrm + zalvm \\
&= (ztsam + zperm + zragm + zricm + zrncm + zfonm + zvamm \\
&+ zvalm + zracm + zetrm - zalvm) + produitfin + m\_ppem - zalrm \\
&+ zalvm \\
&= (zsalm + zchom) + (zretm + zrtom) + zragm + zricm + zrncm \\
&+ zfonm + zvamm + zvalm + zracm + zetrm - zalvm + produitfin \\
&+ m\_ppem - zalrm + zalvm \\
&= zsalm + zchom + zrstm \sout{+ zalrm} + zrtom + zragm + zricm \\
&+ zrncm + zfonm + zvamm + zvalm + zracm + zetrm \sout{- zalvm} \\
&+ produitfin + m\_ppem \sout{- zalrm} \sout{+ zalvm} \\
&= Salaire + Chômage + Retraites + Rentes\ viagères \\
&+ Revenus\ agricoles + Revenus\ industriels\ et\ commerciaux \\
&+ Revenus\ non\ commerciaux + Revenus\ fonciers \\
&+ Revenus\ des\ valeurs\ mobilières + Revenus\ accessoires \\
&+ Revenus\ de\ l'étranger + Revenus\ des\ produits\ financiers \\
&+ Prime\ pour\ l'emploi
\end{aligned}
$$

$$
\begin{aligned}
Revenus\ d'activité &= ztsam + \max(0, zragm + zricm + zrncm) \\
&= Salaire + Chômage \\
&+ \max(0, Revenus\ agricoles + Revenus\ industriels\ et\ commerciaux \\
&+ Revenus\ non\ commerciaux)
\end{aligned}
$$

```
*Revenus primaires = Revenus de travail + chômage + retraites + revenus des
capitaux - pensions alimentaires nettes:
gen double revpri_sep=revdecm_sep+produitfin_sep+m_ppem_sep -
zalrm_sep+zalvm_sep if annee<2017
replace revpri_sep=revdecm_sep+produitfin_sep - zalrm_sep+zalvm_sep if
annee>=2017
la var revpri_sep "Revenus primaires, dans les ménages où les parents se
séparent"

*Revenus d'activité:
gen double revact_sep=ztsam_sep+max(0,zragm_sep+zricm_sep+zrncm_sep)
la var revact_sep "Revenus d'activité, dans les ménages où les parents se
séparent"
```

$$
\begin{aligned}
&Transferts\ publics\ nets = Prestations\ sociales - impôts \\
&= psocm - zimpom - zthabm - zimpvalm - csgim - crdsm - csgpatm - csgvalm \\
&- csgimpm \\
&= (pfamm + minim + logtm) - zimpom - zthabm - zimpvalm - csgim - crdsm - csgpatm \\
&- csgvalm - csgimpm \\
&= Prestations\ familiales + Minima\ sociaux + Prestations\ logement - Impôt\ sur\ le\ revenu \\
&- Taxe\ d'habitation - Prélèvement\ sur\ valeurs\ mobilières - CSG - CRDS \\
&- CSG\&CRDS\ sur\ revenus\ du\ patrimoine - CSG\&CRDS\ sur\ valeurs\ mobilières \\
&- CSG\&CRDS\ sur\ produits\ financiers
\end{aligned}
$$

```
*Transferts publics nets = Prestations familiales - prélevement obligatoires:
cap drop  transferts_sep
gen double transferts_sep=psocm_sep-zimpom_sep - zthabm_sep -zimpvalm_sep -
csgim_sep -crdsm_sep - csgpatm_sep - csgvalm_sep - csgimpm_sep
la var transferts_sep "Transferts publics nets"
```

$$Pensions\ alimentaires\ nettes = zalrm - zalvm$$
$$= Pensions\ alimentaires\ reçues - Pensions\ alimentaires\ versées$$

```
*Pensions alimentaires nettes (reçues - versées) :
gen double pensionsali_sep=zalrm_sep-zalvm_sep
la var pensionsali_sep "Pensions alimentaires nettes"
```

Revenus détaillés des parents de naissance et beaux-parents

On a vu dans la sous-section « Revenus détaillés par déclarant fiscal », page 19, comment apparier aux déclarants fiscaux leurs revenus personnels détaillés. Ici, on cherchera à distinguer qui, des parents de naissance ou d'éventuels beaux-parents (individus mariés, pacsés, ou en union libre avec un parent de naissance, n'apparaissant pas sur le bulletin de naissance de l'enfant), perçoivent effectivement ces revenus détaillés. On utilisera pour cela les informations collectées aux sous-sections « Parentalité des co-déclarants fiscaux », page 52 et « Différences d'âge et de sexe entre l'enfant et tous les autres habitants du logement », page 46.

Pour commencer, on crée des variables de revenus détaillés associés aux parents de naissance de l'enfant. Elles s'appuient sur les revenus détaillés des déclarants fiscaux. Elles sont préfixées par *fim_* et *fif_* :

```
set more off
foreach r in rev_princ ysali ychoi yrsti yalri yragi ybici ybnci { //parents
de naissance mariés/pacsés
    gen fim_`r'= fi1drd_`r' if d1==1
    replace fim_`r'=fi1prd_`r' if d2==1 & mi(fim_`r')
    replace fim_`r'=fi2drd_`r' if d3==1 & mi(fim_`r')
    replace fim_`r'=fi2prd_`r' if d4==1 & mi(fim_`r')
    if "`r'"!="rev_princ" replace fim_`r'=fim_`r'/idprix
    gen fif_`r'= fi1drd_`r' if d1==2
    replace fif_`r'=fi1prd_`r' if d2==2 & mi(fif_`r')
    replace fif_`r'=fi2drd_`r' if d3==2 & mi(fif_`r')
    replace fif_`r'=fi2prd_`r' if d4==2 & mi(fif_`r')
    if "`r'"!="rev_princ" replace fif_`r'=fif_`r'/idprix

}
```

Ensuite, on crée des variables de revenus détaillés associés aux beaux-parents, lorsqu'ils sont mariés ou pacsés aux parents de naissance de l'enfant EDP. Elles s'appuient elles aussi sur les variables des revenus détaillés des déclarants fiscaux. Et sont préfixées par *fimcj_* (conjoint de la mère) et *fifcj_* (conjoint du père) :

```
set more off
```

```
foreach r in rev_princ ysali ychoi yrsti yalri yragi ybici ybnci { //beaux
parents mariés/pacsés
    gen fimcj_`r'=fi1prd_`r' if d1==1 & inlist(d2,0,.) & !mi(fi1prd_`r')
    replace fimcj_`r'=fi1drd_`r' if d2==1 & inlist(d1,0,.) & !mi(fi1drd_`r') &
mi(fimcj_`r')
    replace fimcj_`r'=fi2prd_`r' if d3==1 & inlist(d4,0,.) & !mi(fi2prd_`r') &
mi(fimcj_`r')
    replace fimcj_`r'=fi2drd_`r' if d4==1 & inlist(d3,0,.) & !mi(fi2drd_`r') &
mi(fimcj_`r')
    if "`r'"!="rev_princ" replace fimcj_`r'=fimcj_`r'/idprix
    gen fifcj_`r'=fi1prd_`r' if d1==2 & inlist(d2,0,.) & !mi(fi1prd_`r')
    replace fifcj_`r'=fi1drd_`r' if d2==2 & inlist(d1,0,.) & !mi(fi1drd_`r') &
mi(fifcj_`r')
    replace fifcj_`r'=fi2prd_`r' if d3==2 & inlist(d4,0,.) & !mi(fi2prd_`r') &
mi(fifcj_`r')
    replace fifcj_`r'=fi2drd_`r' if d4==2 & inlist(d3,0,.) & !mi(fi2drd_`r') &
mi(fifcj_`r')
    if "`r'"!="rev_princ" replace fifcj_`r'=fifcj_`r'/idprix
}
```

Ensuite, on complète ces dernières variables par les revenus détaillés des beaux-parents lorsqu'ils sont en unions libres avec les parents de naissance. Etant donné que ces beaux-parents ne sont pas les déclarants fiscaux de l'enfant, on ne peut pas s'appuyer sur les revenus détaillés des déclarants fiscaux. Au lieu de cela, on va devoir retrouver dans la base de Différences d'âge et de sexe, constituée auparavant, les identifiants fiscaux de tous les membres du logement de l'enfant (au maximum, 32, dans le millésime 2019), et parmi ceux-ci, identifier ceux qui sont déclarants fiscaux de leur foyer, ne sont ni le père ni la mère de naissance de l'enfant, et vivent dans un logement où il y a, soit : exactement 2 déclarants fiscaux ; soit plus de 2 déclarants fiscaux, mais où il y a exactement 2 individus de génération plus ancienne (>=15 ans) que l'enfant EDP. Enfin, on élimine les situations où l'enfant EDP ne vit pas dans un ménage avec couple (typmen). On considère alors ces déclarants comme les conjoints cohabitants des parents de naissance, on obtient leurs identifiants fiscaux, que l'on conservera sous les noms de variable *fimcj_id* et *fifcj_id*, et on importe leurs revenus détaillés (*fimcj_* et *fifcj_*) :

```
*beaux-parents en union libre:
ren id idlong //renommage temporaire
egen id=concat(fi1_id_fisc_foy_diff fi1_ordrefip fi1_type_fisc)
gen byte _wave=_app_fi-1
merge 1:1 id_diff id _wave using "$wd\diffage.dta",keep(1 3)
keepusing(declarant? ddn? genre? id? diffage? declarant?? ddn?? genre?? id??
diffage??) nogenerate
ren (ddnm ddnp idi)(ddnm__ ddnp__ idi__) //renommage temporaire
ren (declarant? ddn? genre? id? diffage?)(fi1_declarant? fi1_ddn? fi1_genre?
fi1_id? fi1_diffage?)
ren (declarant?? ddn?? genre?? id?? diffage??)(fi1_declarant?? fi1_ddn??
fi1_genre?? fi1_id?? fi1_diffage??)
drop id
egen id=concat(fi2_id_fisc_foy_diff fi2_ordrefip fi2_type_fisc)
```

```
merge 1:1 id_diff id _wave using "$wd\diffage.dta",keep(1 3)
keepusing(declarant? ddn? genre? id? diffage? declarant?? ddn?? genre?? id??
diffage??) nogenerate
ren (declarant? ddn? genre? id? diffage?)(fi2_declarant? fi2_ddn? fi2_genre?
fi2_id? fi2_diffage?)
ren (declarant?? ddn?? genre?? id?? diffage??)(fi2_declarant?? fi2_ddn??
fi2_genre?? fi2_id?? fi2_diffage??)
ren (ddnm__ ddnp__ idi__)(ddnm ddnp idi)
drop id _wave
ren idlong id
set more off
gen fimcj_id=""
gen fifcj_id=""
//Pour chacun des 32 autres cohabitants de l'enfant dans le logement, on
vérifie s'il s'agit d'un adulte déclarant fiscal cohabitant en couple avec la
mère, ou le père, et on récupère son identifiant si c'est le cas:
forvalues i=1/32 { //conjoint cohabitant de la mère
    replace fimcj_id=fi1_id`i' if fr1_typmen==2 & fi1_ndeclarants==2 & d1==1 &
inlist(d2,0,.) & fi1_diffage`i'<=15 & fi1_declarant`i'==1 & fi1_ddn`i'!=ddnm
    replace fimcj_id=fi2_id`i' if fr2_typmen==2 & fi2_ndeclarants==2 & d3==1 &
inlist(d4,0,.) & fi2_diffage`i'<=15 & fi2_declarant`i'==1 & fi2_ddn`i'!=ddnm &
mi(fimcj_id)
    replace fimcj_id=fi1_id`i' if fr1_typmen==2 & fi1_ndeclarants>=2 &
fi1_nascgen==2 & d1==1 & inlist(d2,0,.) & fi1_diffage`i'<=15 &
fi1_declarant`i'==1 & fi1_ddn`i'!=ddnm
    replace fimcj_id=fi2_id`i' if fr2_typmen==2 & fi2_ndeclarants>=2 &
fi2_nascgen==2 & d3==1 & inlist(d4,0,.) & fi2_diffage`i'<=15 &
fi2_declarant`i'==1 & fi2_ddn`i'!=ddnm & mi(fimcj_id)
}
set more off
forvalues i=1/32 { //conjoint cohabitant du père
    replace fifcj_id=fi1_id`i' if fr1_typmen==2 & fi1_ndeclarants==2 & d1==2 &
inlist(d2,0,.) & fi1_diffage`i'<=15 & fi1_declarant`i'==1 & fi1_ddn`i'!=ddnp
    replace fifcj_id=fi2_id`i' if fr2_typmen==2 & fi2_ndeclarants==2 & d3==2 &
inlist(d4,0,.) & fi2_diffage`i'<=15 & fi2_declarant`i'==1 & fi2_ddn`i'!=ddnp &
mi(fifcj_id)
    replace fifcj_id=fi1_id`i' if fr1_typmen==2 & fi1_ndeclarants>=2 &
fi1_nascgen==2 & d1==2 & inlist(d2,0,.) & fi1_diffage`i'<=15 &
fi1_declarant`i'==1 & fi1_ddn`i'!=ddnp
    replace fifcj_id=fi2_id`i' if fr2_typmen==2 & fi2_ndeclarants>=2 &
fi2_nascgen==2 & d3==2 & inlist(d4,0,.) & fi2_diffage`i'<=15 &
fi2_declarant`i'==1 & fi2_ddn`i'!=ddnp & mi(fifcj_id)
}
//On utilise l'identifiant ainsi obtenu pour récupérer ses revenus détaillés:
set more off
preserve
tempfile revdet
forvalues y=2011/2019 {
    tempfile revdet`y'
```

```
    use edp_be2019_fisc_revdet_`y'.dta,clear
    gen annee=`y'
    egen fimcj_id=concat(id_fisc_foy_diff ordrefip type_fisc)
    clonevar fifcj_id=fimcj_id
    save `revdet`y'',replace
}
use `revdet2011',clear
append using `revdet2012' `revdet2013' `revdet2014' `revdet2015' `revdet2016'
`revdet2017' `revdet2018' `revdet2019'
save `revdet',replace
restore
merge m:1 fimcj_id annee using `revdet', keep(1 3) keepusing(rev_princ ysali
ychoi yrsti yalri yragi ybici ybnci) generate(_merge_fimcjrd)
destring rev_princ fimcj_rev_princ,replace
foreach v of varlist rev_princ ysali ychoi yrsti yalri yragi ybici ybnci {
    if "`v'"=="rev_princ" replace fimcj_`v'=`v' if mi(fimcj_`v')
    else replace fimcj_`v'=`v'/idprix if mi(fimcj_`v')
    drop `v'
}
merge m:1 fifcj_id annee using `revdet', keep(1 3) keepusing(rev_princ ysali
ychoi yrsti yalri yragi ybici ybnci) generate(_merge_fifcjrd)
destring rev_princ fifcj_rev_princ,replace
foreach v of varlist rev_princ ysali ychoi yrsti yalri yragi ybici ybnci {
    if "`v'"=="rev_princ" replace fifcj_`v'=`v' if mi(fifcj_`v')
    else replace fifcj_`v'=`v'/idprix if mi(fifcj_`v')
    drop `v'
}
ren (rev_princ ysali ychoi yrsti yalri yragi ybici ybnci)(fifcj_rev_princ
fifcj_ysali fifcj_ychoi fifcj_yrsti fifcj_yalri fifcj_yragi fifcj_ybici
fifcj_ybnci)
la var fimcj_id "Identifiant fiscal du conjoint de la mère et beau-parent de
l'enfant EDP"
la var fifcj_id "Identifiant fiscal du conjoint du père et beau-parent de
l'enfant EDP"
la var _merge_fimcjrd "Appariement des revenus détaillés du conjoint de la
mère (en union libre)"
la var _merge_fifcjrd "Appariement des revenus détaillés du conjoint du père
(en union libre)"
*On supprime les centaines de variables importées de diffage.dta:
drop fi?_declarant? fi?_declarant?? fi?_ddn? fi?_ddn?? fi?_genre? fi?_genre??
fi?_id? fi?_id?? fi?_diffage? fi?_diffage??
```

Enfin, on cherche les revenus détaillés des parents de naissance lorsque ceux-ci vivent en union libre l'un avec l'autre. Etant donné que la clé d'identification fiscale des parents est connue, l'import des revenus détaillés est relativement aisée :

```
*Parents de naissance en union libre:
```

```
//on a déjà les identifiants fiscaux des parents de naissance, il s'agit des clés fim_idfoy+fim_type_fisc pour la mère, et fif_iddfoy+fif_type_fisc pour le père:
set more off
preserve
tempfile revdet
forvalues i=2011/2019 {
    tempfile revdet`y'
    use edp_be2019_fisc_revdet_`y'.dta,clear
    gen annee=`y'
    egen fim_idfoy=concat(id_fisc_foy_diff ordrefip)
    clonevar fif_idfoy=fim_idfoy
    ren type_fisc fim_type_fisc
    clonevar fif_type_fisc=fim_type_fisc
    save `revdet`y'',replace
}
use `revdet2011',clear
append using `revdet2012' `revdet2013' `revdet2014' `revdet2015' `revdet2016' `revdet2017' `revdet2018' `revdet2019'
save `revdet',replace
restore
merge m:1 fim_idfoy fim_type_fisc annee using `revdet',keep(1 3)
keepusing(rev_princ ysali ychoi yrsti yalri yragi ybici ybnci)
generate(_merge_fimrd)
destring rev_princ fim_rev_princ,replace
foreach v of varlist rev_princ ysali ychoi yrsti yalri yragi ybici ybnci {
    if "`v'"=="rev_princ" replace fim_`v'=`v' if mi(fim_`v')
    else replace fim_`v'=`v'/idprix if mi(fim_`v')
    drop `v'
}
merge m:1 fif_idfoy fif_type_fisc annee using `revdet',keep(1 3)
keepusing(rev_princ ysali ychoi yrsti yalri yragi ybici ybnci)
generate(_merge_fifrd)
destring rev_princ fif_rev_princ,replace
foreach v of varlist rev_princ ysali ychoi yrsti yalri yragi ybici ybnci {
    if "`v'"=="rev_princ" replace fif_`v'=`v' if mi(fif_`v')
    else replace fif_`v'=`v'/idprix if mi(fif_`v')
    drop `v'
}
la var _merge_fimrd "Appariement des revenus détaillés de la mère (en UL)"
la var _merge_fifrd "Appariement des revenus détaillés du père (en UL)"
compress
save "$wd\be2019.dta",replace
```

Catégories sociales et niveaux d'éducation des parents de naissance

Grâce aux variables de l'EAR récupérées à la sous-section « Récupération des informations les plus fraîches des EAR », on peut facilement obtenir la catégorie sociale (*cs*) et le niveau d'éducation (*dipl*) des parents de naissance :

```
*Catégorie sociale et niveau d'éducation:
gen fim_cs=ear1_cs_r if ear1_ddn_r==ddnm & ear1_sexe_r==2 & !mi(ddnm)
replace fim_cs=ear2_cs_r if ear2_ddn_r==ddnm & ear2_sexe_r==2 & !mi(ddnm)
gen fif_cs=ear1_cs_r if ear1_ddn_r==ddnp & ear1_sexe_r==1 & !mi(ddnp)
replace fif_cs=ear2_cs_r if ear2_ddn_r==ddnp & ear2_sexe_r==1 & !mi(ddnp)
gen fim_dipl=ear1_dipl_r if ear1_ddn_r==ddnm & ear1_sexe_r==2 & !mi(ddnm)
replace fim_dipl=ear2_dipl_r if ear2_ddn_r==ddnm & ear2_sexe_r==2 & !mi(ddnm)
gen fif_dipl=ear1_dipl_r if ear1_ddn_r==ddnp & ear1_sexe_r==1 & !mi(ddnp)
replace fif_dipl=ear2_dipl_r if ear2_ddn_r==ddnp & ear2_sexe_r==1 & !mi(ddnp)
gen fimcj_cs=ear1_cs_r if ear2_ddn_r==ddnm & ear2_sexe_r==2 & !mi(ddnm)
replace fimcj_cs=ear2_cs_r if ear1_ddn_r==ddnm & ear1_sexe_r==2 & !mi(ddnm)
gen fifcj_cs=ear1_cs_r if ear2_ddn_r==ddnp & ear2_sexe_r==1 & !mi(ddnp)
replace fifcj_cs=ear2_cs_r if ear1_ddn_r==ddnp & ear1_sexe_r==1 & !mi(ddnp)
gen fimcj_dipl=ear1_dipl_r if ear2_ddn_r==ddnm & ear2_sexe_r==2 & !mi(ddnm)
replace fimcj_dipl=ear2_dipl_r if ear1_ddn_r==ddnm & ear1_sexe_r==2 &
!mi(ddnm)
gen fifcj_dipl=ear1_dipl_r if ear2_ddn_r==ddnp & ear2_sexe_r==1 & !mi(ddnp)
replace fifcj_dipl=ear2_dipl_r if ear1_ddn_r==ddnp & ear1_sexe_r==1 &
!mi(ddnp)
la var fim_cs "Catégorie sociale de la mère"
la var fif_cs "Catégorie sociale du père"
la var fim_dipl "Niveau d'éducation de la mère"
la var fif_dipl "Niveau d'éducation du père"
la var fimcj_cs "Catégorie sociale du conjoint de la mère"
la var fifcj_cs "Catégorie sociale du conjoint du père"
la var fimcj_dipl "Niveau d'éducation du conjoint de la mère"
la var fifcj_dipl "Niveau d'éducation du conjoint du père"
```

Champ retenu

On définit la variable *champ_niv* telle que l'enfant a moins de 18 ans, a un identifiant de logement fiscal, et dont les déclarants ont un niveau de vie ni négatif ni manquant :

```
*Champ:
gen byte champ_niv=(age_t<18 & !inlist(situ,.m,.l) & ndv>0 & ndv!=.)
la var champ_niv "Champ d'étude du niveau de vie"
compress
save "$wd\be2019.dta",replace
```

Fratrie

Composition de la fratrie

Dans cette section, on propose de récupérer les années de naissance des 10 plus jeunes personnes vivant dans le même logement que l'enfant EDP, et ayant au maximum 14 ans de plus que lui. On s'assure que ces personnes ne sont pas les parents de naissance de l'enfant EDP (cas des mères ou pères âgés de moins de 15 ans au moment de la naissance de l'enfant EDP). On considérera que l'ensemble de ces personnes forment la fratrie de l'enfant EDP, et on distribuera leurs années de naissance en fonction de leur rang dans la fratrie (rang 1 : personne la plus jeune du logement ; rang 2 : 2e personne la plus jeune du logement ; etc. jusqu'au rang 10). On nommera, pour le 1er logement fiscal, les variables *f1cohab1_anais*, *f1cohab2_anais*, … *f1cohab10_anais* pour les années de naissance des individus de rang 1 à 10 ; et pour le 2e logement fiscal, on nommera les variables *f2cohab1_anais*, *f2cohab2_anais*, … *f2cohab10_anais* pour les années de naissance des individus de rang 1 à 10. Pour l'enfant EDP, son rang sera renseigné dans la variable *rank*, et son année de naissance sera reproduite dans la variables relative à son rang dans la fratrie.

Lors de l'apparition d'un nouveau-né une année fiscale donnée, le rang des personnes glissera de telle sorte que l'individu de rang 1 devienne de rang 2, l'individu de rang 2 devienne de rang 3, etc. et que le nouveau-né apparaisse comme de rang 1. Un exemple est donné dans le tableau suivant, pour le 1er logement fiscal d'un enfant EDP :

Tableau 16 - Exemple de composition de fratrie

Id_diff	anaisi	annee	rank	f1cohab1_anais	f1cohab2_anais	f1cohab3_anais	f1cohab4_anais	situ
0000069162	2008	2011	1	2008	.	.	.	Vit avec les 2 parents de naissance pacsés dans le même foyer
0000069162	2008	2012	1	2008	.	.	.	Vit avec les 2 parents de naissance pacsés dans le même foyer
0000069162	2008	2013	1	2008	.	.	.	Vit avec les 2 parents de naissance pacsés dans le même foyer
0000069162	2008	2014	1	2008	.	.	.	Vit avec les 2 parents de naissance pacsés dans le même foyer
0000069162	2008	2015	1	2008	.	.	.	Vit avec les 2 parents de naissance pacsés dans le même foyer
0000069162	2008	2016	2	2015	2008	.	.	Vit avec les 2 parents de naissance pacsés dans le même foyer
0000069162	2008	2017	2	2015	2008	.	.	Vit avec les 2 parents de naissance pacsés dans le même foyer
0000069162	2008	2018	3	2015	2012	2008	.	Le père déclare en RA et la mère ne déclare pas l'enfant, père en couple
0000069162	2008	2019	3	2015	2012	2008	.	Le père déclare en RA et la mère ne déclare pas l'enfant, père en couple

Dans cet exemple, l'enfant EDP né en 2008 est l'individu le plus jeune du logement, habité également par ses deux parents de naissance, pacsés. En 2015, un 2e enfant est né de cette union, il sera reporté comme le nouvel enfant de rang 1 l'année suivante (année de déclaration fiscale sur l'année précédente), tandis que l'enfant EDP passera au rang 2. En 2018, les parents se séparent et l'enfant EDP n'est plus déclaré que par son père, qui s'est remis en couple avec une autre personne, et déclare l'enfant de celle-ci, né en 2012. Le 2e enfant du couple précédent reste de rang 1, l'enfant de la nouvelle union arrive avec le 2e rang, tandis que l'enfant EDP glisse au rang 3.

Le code pour obtenir ces différentes variables est le suivant :

```
*On récupére les années de naissance et rangs des 10 membres du logement les + jeunes:
use "$wd\be2019.dta",clear
ren idi idi__ //renommages temporaires
ren id idlong
egen id=concat(fi1_id_fisc_foy_diff fi1_ordrefip fi1_type_fisc)
```

```
gen byte _wave=_app_fi-1
merge 1:1 id_diff id _wave using "$wd\diffage.dta",keep(1 3) keepusing(rank
id? diffage? rank? id?? diffage?? rank??) nogenerate
ren (id? diffage? rank?)(fi1_id? fi1_diffage? fi1_rank?)
ren (id?? diffage?? rank??)(fi1_id?? fi1_diffage?? fi1_rank??)
drop id
egen id=concat(fi2_id_fisc_foy_diff fi2_ordrefip fi2_type_fisc)
merge 1:1 id_diff id _wave using "$wd\diffage.dta",keep(1 3) keepusing(id?
diffage? rank? id?? diffage?? rank??) nogenerate
ren (id? diffage? rank?)(fi2_id? fi2_diffage? fi2_rank?)
ren (id?? diffage?? rank??)(fi2_id?? fi2_diffage?? fi2_rank??)
drop id _wave
ren idlong id
ren idi__ idi
//on récupère les anais des 10 + jeunes membres du logement, de la même
génération ou + jeune que l'enfant EDP, du plus jeune (1) au moins jeune (10)
:
set more off
forvalues i=1/10 {
    di "i=`i'"
    gen f1cohab`i'_anais=.
    gen f2cohab`i'_anais=.
    forvalues j=1/32 {
        replace f1cohab`i'_anais=anaisi+fi1_diffage`j' if fi1_diffage`j'>-15 &
fi1_rank`j'==`i'
        replace f2cohab`i'_anais=anaisi+fi2_diffage`j' if fi2_diffage`j'>-15 &
fi2_rank`j'==`i'
    }
}
forvalues i=1/10 { //on exclut des 10 + jeunes les cas de mères/pères
adolescents (<15 ans d'écart avec l'enfant EDP):
    replace f1cohab`i'_anais=. if (f1cohab`i'_anais==anaism |
f1cohab`i'_anais==anaisp) & !mi(f1cohab`i'_anais)
    replace f2cohab`i'_anais=. if (f2cohab`i'_anais==anaism |
f2cohab`i'_anais==anaisp) & !mi(f2cohab`i'_anais)
}
forvalues i=1/10 { //on rajoute l'année de naissance de l'enfant EDP
    replace f1cohab`i'_anais=anaisi if rank==`i' & _merge_fi1_diffage==3
    replace f2cohab`i'_anais=anaisi if rank==`i' & _merge_fi2_diffage==3
}
drop fi?_rank? fi?_id? fi?_diffage? fi?_rank?? fi?_id?? fi?_diffage??
la var rank "Rang de l'enfant EDP (1 : le plus jeune du logement; 2 : le 2e +
jeune ; etc.)"
egen rownm = rownonmiss(f1cohab?_anais)
gen rank_inv=rownm-rank+1 if !mi(rank,rownm)
replace rank_inv=.n if rank_inv<1
la var rank_inv "Rang de l'enfant inversé (1: le plus âgé du logement; 2: le
2e + âgé, etc."
drop rownm
```

```
compress
save "$wd\be2019.dta",replace
```

Jumeaux

Il y a plusieurs moyens de savoir si un enfant EDP possède un jumeau.

La première méthode consiste à étudier la variable *n_jumeaux*, construite à partir de *n_ctx_catacc* du bulletin de naissance, indiquant le type d'accouchement. En 2019, cette variable donne les effectifs suivants :

Tableau 17 - Effectifs de la variable n_jumeaux

	Effectifs	Proportions	Proportions valides
0 Naissance simple	279 768	29.13%	96.81%
1 Jumeau	9 226	0.96%	3.19%
.	671 035	69.86%	
.d Jumeau décédé à la naissance	317	0.03%	
.t Tri-/Quadruplets	197	0.02%	
Total	960 543	100.00%	

Le taux d'accouchements doubles en France en 2019 était de 1.61%. La proportion retrouvée ici de 3.19%, près de deux fois ce taux, témoigne du fait que chaque jumeau EDP fait l'objet d'une entrée dans la base. On déplore toutefois un fort taux de valeurs manquantes pour cette variable, qui ne s'explique que partiellement par des absences de bulletin de naissance. On propose donc de construire une autre variable d'identification des gemellités, cette fois en comptant les doublons d'observations d'individus partageant des caractéristiques similaires : dates et lieux de naissance de l'individu et de sa mère.

```
*Jumeaux:
duplicates tag annee ddn nai_lieu depcomm ddnm if !mi(ddn ,nai_lieu,depcomm
,ddnm ),gen(jumeaux)
label define jumeaux 0 "Naissance simple" 1 "Jumeaux" 2 "Triplets" 3
"Quadruplets",replace
label values jumeaux jumeaux
la var jumeaux "Gemellités - variable construite à partir de duplicates"
compress
save "$wd\be2019.dta",replace
```

On obtient les effectifs suivants pour 2019 :

Tableau 18 - Effectifs de la variable jumeaux

	Effectifs	Proportions	Proportions valides
0 Naissance simple	702 253	73.11%	96.83%
1 Jumeaux	22 554	2.35%	3.11%
2 Triplets	432	0.04%	0.06%
3 Quadruplets	16	0.00%	0.00%
.	235 288	24.50%	

Total	960 543	100.00%	

L'analyse croisée de *n_jumeaux* et *jumeaux* fait apparaître, sur l'ensemble des années 2011-2019, les effectifs suivants :

Tableau 19 - n_jumeaux vs. jumeaux

		Variable *jumeaux*		
		Naissances simples	Jumeaux	Total
Variable *n_jumeaux*	Naissances simples	2 513 673	765	2 514 438
	Jumeaux	477	82 458	82 935
	Total	2 514 150	83 223	2 597 373

Scinder la base

La taille de la base atteinte fait que son usage en mémoire tend à saturer les ressources collectives. Il vaut mieux par la suite scinder la base en deux sous-bases plus petites. La base *be2019.dta* contiendra toutes les observations, ainsi que toutes les variables à l'exception des variables de revenu. La base *be2019_revenus.dta* contiendra toutes les observations, ainsi que toutes les variables de revenus, à l'exception des variables de revenus constants, définitivement abandonnées.

```
*On supprime les variables de revenus constants et on isole les variables de
revenu dans un fichier séparé:
use "$wd\be2019.dta",clear

keep id_diff annee fr1_an_revenu fr1_reg fr1_dep fr1_depcom fr1_tu10
fr1_nbfoym fr1_nbpersm fr1_inf14m fr1_sup14m fr1_nberam fr1_nb_uc fr1_agerf
fr1_sexerf fr1_occtyp fr1_typmen9 fr1_i_champm fr1_i_pauvre50m fr1_i_pauvre60m
fr1_nb_alloc_cnaf fr1_nb_alloc_cnav fr1_nb_alloc_msaf fr1_nb_alloc_msav
fr1_nivviem fr1_nivviemcst fr1_centile fr1_revdispm fr1_revperm fr1_revinim
fr1_revdecm fr1_ztsam fr1_zsalm fr1_zchom fr1_zperm fr1_zretm fr1_zrstm
fr1_zalrm fr1_zrtom fr1_zragm fr1_zricm fr1_zrncm fr1_zfonm fr1_zvamm0
fr1_zvamm fr1_zvalm0 fr1_zvalm fr1_zracm fr1_zetrm fr1_zalvm fr1_psocm
fr1_pfamm fr1_m_pnam fr1_m_pajem fr1_m_clcam fr1_m_colcam fr1_m_afm fr1_m_cfm
fr1_m_arsm fr1_m_aeehm fr1_m_asfm fr1_minim fr1_m_minvm fr1_m_rsam
fr1_m_rsa_actm fr1_m_rsa_socm fr1_m_rsa_pfam fr1_m_aah_caahm fr1_logtm
fr1_m_alfm fr1_m_aplm fr1_m_alsm fr1_m_alsetm fr1_m_ppem fr1_zimpom
fr1_zimpvalm fr1_zthabm fr1_csgim fr1_csgisalm fr1_crdsm fr1_crdssalm
fr1_csgpatm fr1_csgvalm fr1_csgimpm fr1_csgdm fr1_csgdsalm fr1_produitfin
fr1_zppem fr1_zquom fr1_zdivm fr1_zglom fr1_zavfm fr1_zimpform fr1_gardem
fr1_m_ppam fr1_m_preparem fr1_m_prepare_mm fr1_nbpac fr1_servdomm fr1_typmenr
fr1_typmen fr1_zactm fr1_zavfm_decl fr1_zimpom_anc fr1_zimpom_anc_decl
fr1_zimpom_decl fr1_zthabm_decl _merge_fr1 fr2_an_revenu fr2_reg fr2_dep
fr2_depcom fr2_tu10 fr2_nbfoym fr2_nbpersm fr2_inf14m fr2_sup14m fr2_nberam
fr2_nb_uc fr2_agerf fr2_sexerf fr2_occtyp fr2_typmen9 fr2_i_champm
fr2_i_pauvre50m fr2_i_pauvre60m fr2_nb_alloc_cnaf fr2_nb_alloc_cnav
fr2_nb_alloc_msaf fr2_nb_alloc_msav fr2_nivviem fr2_nivviemcst fr2_centile
fr2_revdispm fr2_revperm fr2_revinim fr2_revdecm fr2_ztsam fr2_zsalm fr2_zchom
fr2_zperm fr2_zretm fr2_zrstm fr2_zalrm fr2_zrtom fr2_zragm fr2_zricm
fr2_zrncm fr2_zfonm fr2_zvamm0 fr2_zvamm fr2_zvalm0 fr2_zvalm fr2_zracm
fr2_zetrm fr2_zalvm fr2_psocm fr2_pfamm fr2_m_pnam fr2_m_pajem fr2_m_clcam
fr2_m_colcam fr2_m_afm fr2_m_cfm fr2_m_arsm fr2_m_aeehm fr2_m_asfm fr2_minim
fr2_m_minvm fr2_m_rsam fr2_m_rsa_actm fr2_m_rsa_socm fr2_m_rsa_pfam
fr2_m_aah_caahm fr2_logtm fr2_m_alfm fr2_m_aplm fr2_m_alsm fr2_m_alsetm
fr2_m_ppem fr2_zimpom fr2_zimpvalm fr2_zthabm fr2_csgim fr2_csgisalm fr2_crdsm
fr2_crdssalm fr2_csgpatm fr2_csgvalm fr2_csgimpm fr2_csgdm fr2_csgdsalm
fr2_produitfin fr2_zppem fr2_zquom fr2_zdivm fr2_zglom fr2_zavfm fr2_zimpform
fr2_gardem fr2_m_ppam fr2_m_preparem fr2_m_prepare_mm fr2_nbpac fr2_servdomm
fr2_typmenr fr2_typmen fr2_zactm fr2_zavfm_decl fr2_zimpom_anc
fr2_zimpom_anc_decl fr2_zimpom_decl fr2_zthabm_decl _merge_fr2 revdecm_sep
produitfin_sep psocm_sep m_ppem_sep zimpom_sep zthabm_sep zimpvalm_sep
```

```
csgim_sep crdsm_sep csgpatm_sep csgvalm_sep csgimpm_sep ztsam_sep zperm_sep
zragm_sep zricm_sep zrncm_sep zfonm_sep zvamm_sep zvalm_sep zracm_sep
zetrm_sep zalvm_sep pfamm_sep minim_sep logtm_sep csgisalm_sep zalrm_sep
zsalm_sep zchom_sep revpri_sep revact_sep transferts_sep pensionsali_sep
ysalid_sep ysalip_sep ychoid_sep ychoip_sep yalrid_sep yalrip_sep

save "$wd\be2019_revenus.dta",replace
use "$wd\be2019.dta",clear

drop fr1_revdecmcst fr1_produitfincst fr1_psocmcst fr1_m_ppemcst fr1_zimpomcst
fr1_zthabmcst fr1_zimpvalmcst fr1_csgimcst fr1_crdsmcst fr1_csgpatmcst
fr1_csgvalmcst fr1_csgimpmcst fr1_ztsamcst fr1_zpermcst fr1_zragmcst
fr1_zricmcst fr1_zrncmcst fr1_zfonmcst fr1_zvammcst fr1_zvalmcst fr1_zracmcst
fr1_zetrmcst fr1_zalvmcst fr1_pfammcst fr1_minimcst fr1_logtmcst
fr1_csgisalmcst fr1_zalrmcst fr1_zsalmcst fr1_zchomcst fi1drd_ysalicst
fi1prd_ysalicst fi1drd_ychoicst fi1prd_ychoicst fi1drd_yalricst
fi1prd_yalricst fr2_revdecmcst fr2_produitfincst fr2_psocmcst fr2_m_ppemcst
fr2_zimpomcst fr2_zthabmcst fr2_zimpvalmcst fr2_csgimcst fr2_crdsmcst
fr2_csgpatmcst fr2_csgvalmcst fr2_csgimpmcst fr2_ztsamcst fr2_zpermcst
fr2_zragmcst fr2_zricmcst fr2_zrncmcst fr2_zfonmcst fr2_zvammcst fr2_zvalmcst
fr2_zracmcst fr2_zetrmcst fr2_zalvmcst fr2_pfammcst fr2_minimcst fr2_logtmcst
fr2_csgisalmcst fr2_zalrmcst fr2_zsalmcst fr2_zchomcst fi2drd_ysalicst
fi2prd_ysalicst fi2drd_ychoicst fi2prd_ychoicst fi2drd_yalricst
fi2prd_yalricst

drop fr1_an_revenu fr1_reg fr1_dep fr1_depcom fr1_tu10 fr1_nbfoym fr1_nbpersm
fr1_inf14m fr1_sup14m fr1_nberam fr1_nb_uc fr1_agerf fr1_sexerf fr1_occtyp
fr1_typmen9 fr1_i_champm fr1_i_pauvre50m fr1_i_pauvre60m fr1_nb_alloc_cnaf
fr1_nb_alloc_cnav fr1_nb_alloc_msaf fr1_nb_alloc_msav fr1_nivviem
fr1_nivviemcst fr1_centile fr1_revdispm fr1_revperm fr1_revinim fr1_revdecm
fr1_ztsam fr1_zsalm fr1_zchom fr1_zperm fr1_zretm fr1_zrstm fr1_zalrm
fr1_zrtom fr1_zragm fr1_zricm fr1_zrncm fr1_zfonm fr1_zvamm0 fr1_zvamm
fr1_zvalm0 fr1_zvalm fr1_zracm fr1_zetrm fr1_zalvm fr1_psocm fr1_pfamm
fr1_m_pnam fr1_m_pajem fr1_m_clcam fr1_m_colcam fr1_m_afm fr1_m_cfm fr1_m_arsm
fr1_m_aeehm fr1_m_asfm fr1_minim fr1_m_minvm fr1_m_rsam fr1_m_rsa_actm
fr1_m_rsa_socm fr1_m_rsa_pfam fr1_m_aah_caahm fr1_logtm fr1_m_alfm fr1_m_aplm
fr1_m_alsm fr1_m_alsetm fr1_m_ppem fr1_zimpom fr1_zimpvalm fr1_zthabm
fr1_csgim fr1_csgisalm fr1_crdsm fr1_crdssalm fr1_csgpatm fr1_csgvalm
fr1_csgimpm fr1_csgdm fr1_csgdsalm fr1_produitfin fr1_zppem fr1_zquom
fr1_zdivm fr1_zglom fr1_zavfm fr1_zimpform fr1_gardem fr1_m_ppam
fr1_m_preparem fr1_m_prepare_mm fr1_nbpac fr1_servdomm fr1_typmenr fr1_typmen
fr1_zactm fr1_zavfm_decl fr1_zimpom_anc fr1_zimpom_anc_decl fr1_zimpom_decl
fr1_zthabm_decl _merge_fr1 fr2_an_revenu fr2_reg fr2_dep fr2_depcom fr2_tu10
fr2_nbfoym fr2_nbpersm fr2_inf14m fr2_sup14m fr2_nberam fr2_nb_uc fr2_agerf
fr2_sexerf fr2_occtyp fr2_typmen9 fr2_i_champm fr2_i_pauvre50m fr2_i_pauvre60m
fr2_nb_alloc_cnaf fr2_nb_alloc_cnav fr2_nb_alloc_msaf fr2_nb_alloc_msav
fr2_nivviem fr2_nivviemcst fr2_centile fr2_revdispm fr2_revperm fr2_revinim
fr2_revdecm fr2_ztsam fr2_zsalm fr2_zchom fr2_zperm fr2_zretm fr2_zrstm
fr2_zalrm fr2_zrtom fr2_zragm fr2_zricm fr2_zrncm fr2_zfonm fr2_zvamm0
```

```
fr2_zvamm fr2_zvalm0 fr2_zvalm fr2_zracm fr2_zetrm fr2_zalvm fr2_psocm
fr2_pfamm fr2_m_pnam fr2_m_pajem fr2_m_clcam fr2_m_colcam fr2_m_afm fr2_m_cfm
fr2_m_arsm fr2_m_aeehm fr2_m_asfm fr2_minim fr2_m_minvm fr2_m_rsam
fr2_m_rsa_actm fr2_m_rsa_socm fr2_m_rsa_pfam fr2_m_aah_caahm fr2_logtm
fr2_m_alfm fr2_m_aplm fr2_m_alsm fr2_m_alsetm fr2_m_ppem fr2_zimpom
fr2_zimpvalm fr2_zthabm fr2_csgim fr2_csgisalm fr2_crdsm fr2_crdssalm
fr2_csgpatm fr2_csgvalm fr2_csgimpm fr2_csgdm fr2_csgdsalm fr2_produitfin
fr2_zppem fr2_zquom fr2_zdivm fr2_zglom fr2_zavfm fr2_zimpform fr2_gardem
fr2_m_ppam fr2_m_preparem fr2_m_prepare_mm fr2_nbpac fr2_servdomm fr2_typmenr
fr2_typmen fr2_zactm fr2_zavfm_decl fr2_zimpom_anc fr2_zimpom_anc_decl
fr2_zimpom_decl fr2_zthabm_decl _merge_fr2 revdecm_sep produitfin_sep
psocm_sep m_ppem_sep zimpom_sep zthabm_sep zimpvalm_sep csgim_sep crdsm_sep
csgpatm_sep csgvalm_sep csgimpm_sep ztsam_sep zperm_sep zragm_sep zricm_sep
zrncm_sep zfonm_sep zvamm_sep zvalm_sep zracm_sep zetrm_sep zalvm_sep
pfamm_sep minim_sep logtm_sep csgisalm_sep zalrm_sep zsalm_sep zchom_sep
revpri_sep revact_sep transferts_sep pensionsali_sep ysalid_sep ysalip_sep
ychoid_sep ychoip_sep yalrid_sep yalrip_sep

save "$wd\be2019.dta",replace
```

On illustre ici comment recréer la base complète :

```
use "$wd\be2019.dta",clear
merge 1:1 id_diff annee using "$wd\be2019_revenus.dta",nogenerate keep(1 3)
```

A partir de la base complète, on extrait un échantillon aléatoire constitué d'1/16^{e} des observations, et toutes les variables (y compris de revenus). L'échantillonnage se fait sur l'extraction de tous les enfants nés un 1er octobre (jour EDP). L'échantillon résultant s'appellera *be2019_1eroct.dta* :

```
*Prélever 1/16e de la base (1er octobre):
keep if mnaisi==10 & jnaisi==1
save "$wd\be2019_1eroct.dta",replace
```