Direction des Études et Synthèses Économiques

G 2010 / 14

Le bloc retraites du modèle Destinie 2 : guide de l'utilisateur

Didier BLANCHET
Emmanuelle CRENNER

Document de travail

Institut National de la Statistique et des Études Économiques

INSTITUT NATIONAL DE LA STATISTIQUE ET DES ÉTUDES ÉCONOMIQUES

Série des documents de travail
de la Direction des Études et Synthèses Économiques

G 2010 / 14

Le bloc retraites du modèle Destinie 2 : guide de l'utilisateur

Didier BLANCHET *
Emmanuelle CRENNER **

JUILLET 2010

Le bloc retraite présenté dans ce document s'appuie sur les autres composantes du modèle Destinie mises au point par Sylvie LE MINEZ et Sophie BUFFETEAU. Sa finalisation a beaucoup bénéficié de premiers travaux d'exploitation conduits avec Magali BEFFY, Marion BACHELET et Claire MARBOT.

* Département des Études Économiques d'Ensemble - Timbre G201 - 15, bd Gabriel Péri - BP 100 - 92244 MALAKOFF CEDEX

** Appartenait, lors de ce travail, à la division « Redistribution et Politiques Sociales » du même département.

Département des Études Économiques d'Ensemble - Timbre G201 - 15, bd Gabriel Péri - BP 100 - 92244 MALAKOFF CEDEX - France - Tél. : 33 (1) 41 17 60 68 - Fax : 33 (1) 41 17 60 45 - CEDEX - E-mail : d3e-dg@insee.fr - Site Web INSEE : http://www.insee.fr

Le bloc retraites du modèle Destinie 2 : guide de l'utilisateur

Résumé

Le modèle Destinie est un modèle de microsimulation dynamique dont l'objectif principal est la projection des retraites aux niveaux individuel et agrégé. Ce document présente comment la nouvelle version du modèle permet de réaliser de telles projections. Elles ne découlent pas d'un modèle fermé que l'utilisateur se contenterait de reparamétrer pour la simulation de variantes. Le nouveau modèle Destinie 2 a plutôt été conçu sous forme d'une bibliothèque d'outils mobilisables pour la construction de programmes de projection sur mesure. On présente l'ensemble de ces bibliothèques et divers exemples de mise en œuvre. Le document vise aussi bien les utilisations standard du modèle que les utilisations plus avancées, y compris celles qui nécessiteraient la construction de modules additionnels.

Mots-clés : Microsimulation, retraites

The pension projection package of the Destinie 2 model: user's guide

Abstract

The Destinie model is a dynamic microsimulation model whose main purpose is the long run projection of pensions both at the macro and micro levels. This document presents how such simulations can be built with the new version of the model. They do not rely on a closed ready-made program where variants would be obtained by just changing values for a predetermined set of parameters. The new Destinie 2 model has been rather designed as a toolbox of modules, allowing the construction of ad hoc projection programs that can be tailored to users' demands. We provide a comprehensive view of this toolbox. This document addresses the needs of both basic users and more advanced ones, including for applications that would eventually require the building of additional modules.

Keywords: Microsimulation, pensions.

Classification JEL : C63, H55

Sommaire

I - Introduction

Le modèle Destinie est un modèle de microsimulation dynamique dont l'objectif principal est la projection à long terme des retraites. Sa première version avait été construite à l'Insee dans les années 1990 (Blanchet et Chanut, 1998 ; Division Redistribution et Politiques Sociales, 1999 ; Bardaji *et al.*, 2006). Elle a été utilisée jusqu'en 2008 mais elle s'est progressivement avérée de plus en plus difficile à maintenir. Une réécriture complète a donc débuté à partir de 2005. Les principes généraux de cette réécriture sont présentés dans un document joint (Blanchet, Buffeteau, Crenner et Le Minez, 2010) auquel sont renvoyés les lecteurs uniquement intéressés par une vue d'ensemble du nouveau modèle. Le présent document est un document plus technique, destiné à ses utilisateurs directs ou aux lecteurs souhaitant avoir une connaissance plus approfondie de ses fonctionnalités.

Pour introduire ce document, on va rappeler brièvement le principe de la microsimulation et indiquer quelle est l'architecture du nouveau modèle. L'une de ses caractéristiques est de ne plus proposer un programme tout fait de simulation des droits à retraite que l'utilisateur se contenterait de reparamétrer ou compléter en fonction des besoins. Le nouveau modèle se présente plutôt comme un outil logiciel permettant la construction de programmes de simulation sur mesure. L'objectif du présent guide est de donner tous les éléments nécessaires à la construction de tels programmes. On présentera d'entrée de jeu un exemple d'un tel programme ce qui permettra de donner un premier aperçu des questions que va couvrir ce document.

I.1 Objectifs et principales caractéristiques du modèle

Simuler les droits à retraite suppose de connaître l'histoire détaillée des individus tant sur le plan démographique (vie en couple, naissance des enfants, décès du conjoint) que sur le plan professionnel (années en emploi, salaires perçus, années au chômage, périodes d'inactivité…). Pour construire ce type de simulation, le modèle Destinie part d'un échantillon représentatif de la population française et il en projette le devenir au niveau individuel par combinaison de règles d'évolution déterministes ou stochastiques. Les droits à retraite sont calculés sur cette base et le fait d'en disposer au niveau individuel permet de produire une grande variété de résultats : non seulement des agrégats, mais aussi des indicateurs d'inégalité des retraites ou de l'ampleur des effets redistributifs de tel ou tel scénario de réforme.

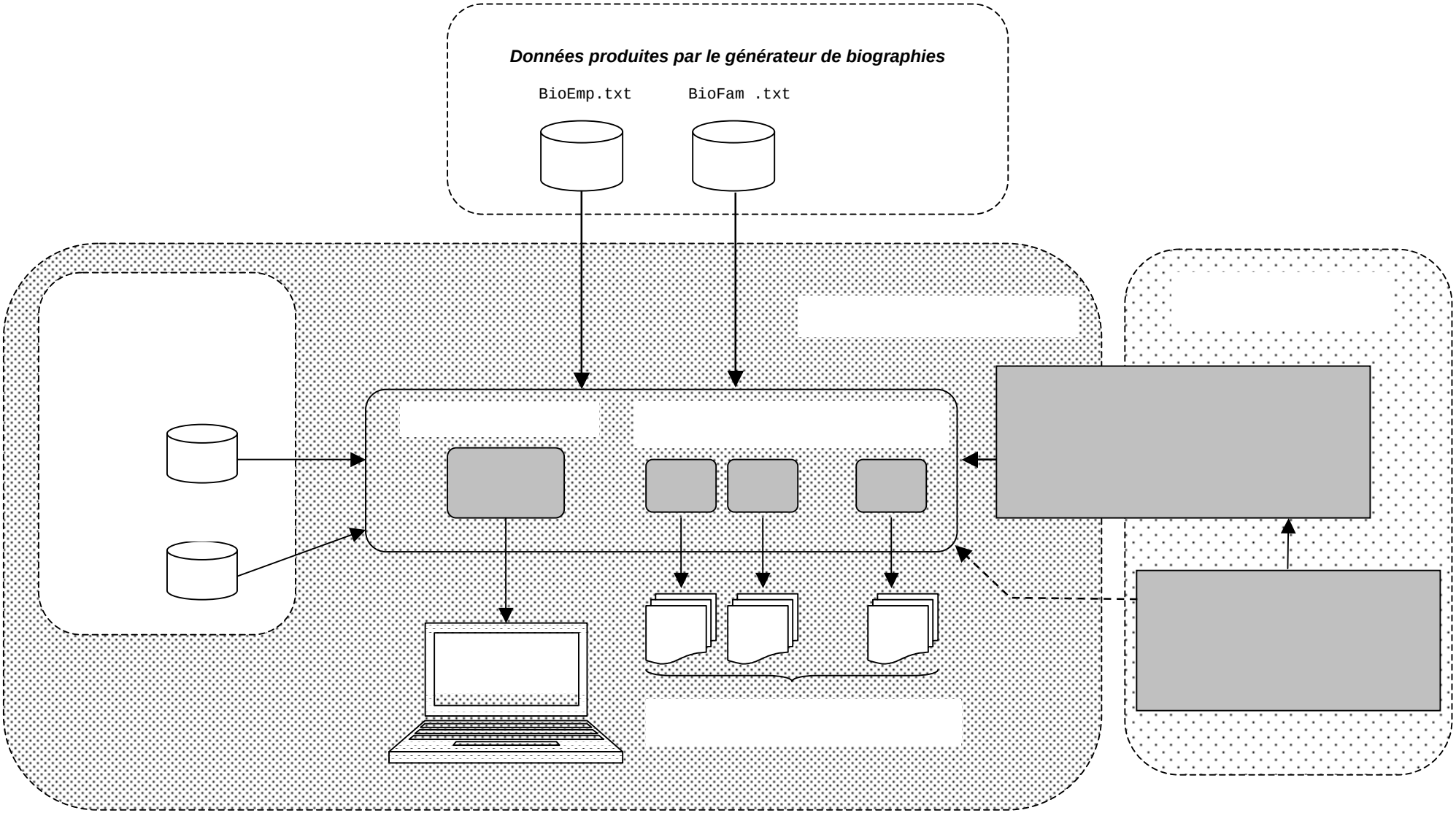
La première version du modèle Destinie avait été écrite en langage Turbo Pascal. La décision de réécriture s'explique entre autres par la difficulté à continuer de faire évoluer le modèle sous ce langage. Après divers essais, on a opté pour une réécriture dans le langage Perl, et cette réécriture a également été l'occasion de revisiter entièrement l'architecture du modèle. Pour la résumer brièvement, on dira que la nouvelle version a cherché à améliorer les performances de l'ancien modèle dans trois domaines principaux: la légèreté, la souplesse, et la traçabilité des résultats des simulations.

Ceci a notamment conduit à adopter une architecture à deux étages (figure 1).

- Le premier étage est qualifié de *générateur de biographies*. Il simule des trajectoires démographiques et professionnelles en faisant abstraction des départs en retraite, et donc jusqu'à des âges aussi élevés que nécessaire pour pouvoir simuler des scénarios de remontée éventuellement importante des âges de liquidation. Les biographies qui sont produites par ce générateur sont stockées dans des fichiers intermédiaires.

- Le second étage concerne la simulation des retraites. Les opérations de ce second étage consistent à relire les données biographiques fournies par le générateur, et à y rajouter une simulation des comportements de départ en retraite et le calcul des droits associés.

Figure 1 : architecture simplifiée du bloc retraite du modèle Destinie 2



Cette organisation a facilité les travaux de développement en permettant un travail autonome sur chacun des deux étages. En utilisation courante, son intérêt est qu'elle évite d'avoir à relancer l'ensemble de la simulation des trajectoires démographiques et professionnelles pour chaque variante de retraite, ce qui allège les travaux de simulation. Un intérêt additionnel est que, par construction, les variantes retraites tourneront sur une population d'individus aux caractéristiques fixes et entièrement contrôlées, ce qui facilite l'analyse des effets des réformes.

La construction de ces variantes se présente par ailleurs d'une manière très différente de ce qu'elle était dans le modèle Destinie 1. Les projections ne découlent plus d'un programme de référence unique qu'on modifie d'un scénario à l'autre. Compte tenu de la grande diversité des exercices de projection qui peuvent être demandés au modèle, on a opté pour un fonctionnement beaucoup plus ouvert et modulaire. Ce qui est proposé n'est plus un programme, mais des bibliothèques d'utilitaires servant à construire des programmes de projection « à la carte ». Ceci rend l'instrument beaucoup plus flexible, et améliore la traçabilité de ses résultats car le principe est désormais d'avoir autant de programmes différents que d'exercices de simulation. Avec un tel système, toutes les informations sur les hypothèses qui ont conduit à une simulation peuvent être automatiquement retrouvées dans le programme qui l'a généré, sans qu'il y ait besoin d'autre forme d'archivage.

La figure 1 présente l'organisation générale des fichiers et des programmes. Les programmes de simulation des retraites sont des programmes source Perl de type `.pl` représentés au centre du diagramme qui lisent les fichiers fournis par le générateur de biographies et qui permettent de produire des fichiers de résultats qui sont des fichiers Excel ou des fichiers texte.

I.2 Un exemple introductif

L'objectif de ce guide est de donner tous les éléments nécessaires à la construction de tels programmes. Pour qu'on ait tout de suite une idée de leur contenu, la figure 2 donne un exemple simple pour une simulation élémentaire. On y reconnaît deux boucles emboitées sur la date et l'ensemble des individus simulés, caractéristiques d'un modèle de microsimulation dynamique. À l'intérieur de ces boucles, on utilise des fonctions prédéfinies telles que `Relec` (relecture des données biographiques en input), `UseLeg` (choix de la législation) ou `SimDir` (simulation de la liquidation des droits directs). La boucle temporelle contient également des appels à des fonctions de calcul de résultats agrégés telles que `Mvar`, `Svar` ou `Count` dont les résultats sont sauvegardés en fin de programme par appel à d'autres fonctions prédéfinies (`ClassOut`, `SavSer` et `CloseOut`).

Cet exemple montre que le bloc retraites du nouveau modèle Destinie fonctionne bien comme un mini-logiciel de calcul des droits à retraite, dont les éléments sont des variables prédéfinies et des programmes fournis par les bibliothèques indiquées sur la partie droite de la figure 1. Ces éléments se divisent en deux groupes :

- Le premier regroupe ce qui est indispensable aux utilisations les plus courantes du modèle. Il s'agit de variables ou programmes définis dans quatre bibliothèques : la librairie qui déclare l'ensemble des variables Destinie (`DefVarRetr.pm`), celle qui fournit les principaux utilitaires de calcul des retraites (`OutilsRetr.pm`) et des bibliothèques de programmes de tabulation et de sortie sous Excel (`OutilsTab.pm` et `OutilsExcel.pm`).
- La connaissance du second groupe n'est utile que pour des utilisations plus avancées. Il s'agit d'une part de certaines des composantes des bibliothèques mentionnées à l'instant qui n'ont pas besoin d'être connues pour les utilisations usuelles. Il s'agit par ailleurs de l'intégralité du contenu de quatre autres bibliothèques qui, en utilisation courante, ne sont mobilisées qu'indirectement, mais dans lesquelles on peut aller puiser pour des calculs plus élaborés. Cet ensemble contient notamment deux bibliothèques sur lesquelles s'appuie `OutilsrRetr.pm` : la bibliothèque contenant les détails des calculs de droits à retraite (`OutilsDroits.pm`), et la bibliothèque simulant les différentes options de comportement de départ en retraite (`OutilsComp.pm`). Il existe aussi une bibliothèque d'outils généraux

partagés par l'ensemble des autres bibliothèques (`OutilsBase.pm`), et une bibliothèque permettant divers types de calculs au niveau des ménages (`OutilsMen.pm`).

Figure 2 : un exemple simple de programme de simulation

```
# Ce programme effectue une projection des retraites sous l'hypothèse départ
# au taux plein avec la législation de 2003. Il calcule trois variables
# d'output :
# - le niveau de la retraite moyenne,
# - le ratio retraites/masse salariale
# - Le ratio retraités/actifs occupés
# Ces résultats sont sauvegardés dans le fichier Excel Exemple.xls
```

1: Préambule

```
use strict;

use lib "..\/..\/Outils";
use DefVarRetr;
use OutilsRetr;
use OutilsTab;
use OutilsExcel;
my ($i,$t,@PENSION_MOYENNE,@RATIO_DEM,@RATIO_FIN,$resultats);
UseBios("Repertoire Bios",2003,1/9246);
UseOpt("TP") ;
```

2: Boucles temporelle et individuelle

```
# Boucle temporelle
for $t (103..150)
{
   # Simulation
   Relec($t);
   for $i (1..IMax)
   {
      UseLegRetroMax(2003);
      UseLeg($t,$anaiss[$i]);
      SimDir($i);
   }
```

3 : Comptages

```
   # Comptages année courante

   $PENSION_MOYENNE[$t] = MVar(@pension, sub{$pension[$_]>0   });
   $RATIO_FINANCIER[$t] = SVar(@pension)/Svar(@salaire);
   $RATIO_DEMOGR[$t]    = Ratio(sub{$pension[$_]>0},sub{$salaire[$_]>0});

};  # Fin boucle temporelle
```

4 : Sortie des résultats

```
$resultats = ClassOut("Exemple.xls");
SavSer($resultats,"Resultats","Principaux agregats",
                  "Pension moyenne    ",@PENSION_MOYENNE,
                  "Ratio financier    ",@RATIO_FIN,
                  "Ratio démographique",@RATIO_DEM) ;
CloseOut($resultats);
```

Pour revenir aux utilisations courantes, la droite de la figure 1 fait état d'un dernier instrument, le programme prédéfini `Explore.pl`. Ce programme s'appuie sur la même boite à outils que les programmes `.pl` sur mesure mais son but n'est pas de construire des simulations complètes. Il sert à explorer interactivement le contenu détaillé des fichiers d'input et de tester différentes hypothèses de départ en retraite, individu par individu. Cet outil est principalement destiné au débogage des programmes ou à l'examen de résultats au niveau fin, par exemple lorsqu'une simulation conduit à des résultats inattendus pour un certain nombre d'individus. Les résultats de cet explorateur peuvent éventuellement être dirigés vers des fichiers `.xls` ou `.txt`, mais sont le plus souvent consultés à l'écran, dans une fenêtre d'exécution MS-DOS.

I.3 Plan du document

Ayant présenté cette organisation générale du programme, on peut préciser les objectifs et le plan du présent document. Il n'abordera pas le fonctionnement du générateur de biographies. Le lecteur ayant besoin de le connaître est renvoyé à Buffeteau et Le Minez (à paraître). Il s'adresse à l'utilisateur qui part de fichiers de biographies déjà constitués et qui souhaite construire des programmes de projection des retraites. Il présente l'ensemble de ce qui doit être connu pour construire ces projections.

La section II est principalement consacrée à la présentation du langage Perl et de son environnement de programmation. Elle peut-être sautée par l'utilisateur qui connaît déjà ce langage, hormis sa dernière section consacrée aux principes d'organisation physique des fichiers Destinie.

Les sections III à V sont ensuite consacrées aux utilisations courantes. La section III permet de découvrir l'ensemble des variables que Destinie met à disposition de l'utilisateur et indique comment procéder lorsqu'il doit en créer de nouvelles. La section IV présente en détail les fonctions d'usage courant et notamment celles qui figuraient dans l'exemple de la figure 2. La section V présente l'explorateur et les autres possibilités qui existent pour inspecter les résultats d'une simulation au niveau individuel.

La section VI est destinée à l'utilisateur avancé. Elle contient la description complète de l'ensemble des procédures des différents modules, hormis celles qui auront déjà été présentées aux sections IV et V.

La section VII s'adresse pour finir à l'utilisateur qui souhaiterait développer des modules additionnels. Par exemple, la version I du modèle Destinie avait servi à des projections de dépendance aux grands âges (Duée et Rebillard, 2005) et cet exercice devrait être prochainement renouvelé. Un module simulant la consommation de soins de santé est également en cours de développement (Albouy, Davezies et Debrand, 2010). Un module épargne/patrimoine est également à l'étude. On donnera quelques principes simples concernant la façon dont peuvent être développés de tels modules.

II - L'environnement de programmation

Cette première partie est consacrée à l'environnement général de programmation de Destinie 2. L'essentiel est consacré à présenter les rudiments du langage Perl (section II.1) et un exemple d'outil de développement intégré dans ce langage qui est celui qui a servi à la mise au point de ce modèle Destinie 2 (section II.2). Ces deux sections peuvent être sautées par l'utilisateur qui connaît déjà ce langage. La dernière section présente la structure physique d'organisation des fichiers de travail : sa lecture est donc indispensable (section II.3).

II.1 Introduction au langage Perl

Plusieurs possibilités de langage ont été explorées pour la réécriture du modèle Destinie. On aurait pu envisager de s'appuyer sur un logiciel statistique tels que SAS. L'intérêt aurait été de bénéficier de procédures de calcul statistique ou de mise en forme des résultats toutes faites. Mais ces logiciels sont surtout conçus pour des traitements de données statistiques en bloc, alors que la microsimulation oblige à alterner traitement en blocs et travail différencié individu par individu. De manière plus générale, le recours à un logiciel évolué crée le risque de se laisser enfermer dans des solutions de programmation restrictives interdisant de faire évoluer le modèle dans des directions non prévues au départ.

Il vaut donc, mieux, en général, s'appuyer sur un langage élémentaire, ce qui avait déjà été le cas pour Destinie 1. Un assez grand nombre de modèles existants recourent au langage C++ et à la programmation orientée objet, par exemple la famille de modèles développés à partir du logiciel ModGen (Statistique Canada, 2010)[1]. Quelques tentatives ont été faites de reprogrammation en C++ d'une partie du programme Destinie 1 mais les pousser à terme aurait nécessité un investissement et des compétences supérieures aux moyens disponibles sur le projet : il s'agissait d'un changement de logique de programmation trop important par rapport à l'expérience acquise sur le modèle Destinie 1.

On a donc préféré se rabattre sur un langage permettant de conserver l'esprit de la programmation de Destinie 1, mais sans les limites auxquelles avait fini par se heurter le langage Turbo Pascal. Le langage Perl est apparu constituer une réponse assez pratique à nos besoins. Ce langage n'est pas sans défaut, mais c'est un langage extrêmement flexible, qui n'impose aucun style de programmation *a priori*. Il présentait un certain nombre d'avantages additionnels pour la microsimulation, tels que l'allocation dynamique des tableaux, une grande souplesse dans la spécification des sous-programmes, l'existence d'une fonction de tri incorporée au langage ou l'existence de bibliothèques d'interface avec Excel. D'autres fonctionnalités se sont avérées utiles *a posteriori* : le fait qu'il s'agisse d'un langage interprété plutôt que compilé a notamment permis la construction de fonctions de tabulation très flexibles et a rendu extrêmement facile l'ajout d'un explorateur interactif à la batterie d'instruments qui avait été prévue au départ.

Les quelques sections qui suivent ne peuvent évidemment se substituer à la lecture de présentations plus complètes du langage, mais elles fournissent le minimum indispensable à la compréhension du reste du manuel et devraient suffire à la programmation d'applications Destinie élémentaires. On notera qu'on n'y aborde pas la question des entrées-sorties, car il existe dans Destinie 2 des fonctions qui permettent d'éviter le recours aux entrées sorties de base de Perl. Celles-ci peuvent néanmoins s'avérer utiles dans certains cas, mais les exemples correspondants seront donnés au fur et à mesure du reste du document, notamment dans les sections IV.6 et V.2.

[1] Parmi les autres produits disponibles pour le développement de modèles de microsimulation dynamique figure le logiciel LIAM, mais dont le développement est intervenu après le lancement de la réécriture de Destinie 2 (voir O'Donoghue, 2008).

II.1.1 Éléments généraux de syntaxe

Perl différencie majuscules et minuscules. Les instructions sont séparées par des points virgule. Elles peuvent s'étendre sur plusieurs lignes si nécessaire.

La syntaxe de Perl est assez libre -souvent même jugée trop libre. C'est à l'utilisateur de se créer ses propres règles de présentation et d'essayer de s'y tenir. Toutes les librairies et exemples de programme du modèle Destinie 2 ont essayé de suivre les règles suivantes dont on a commencé à avoir l'illustration dans l'exemple de l'introduction :

- Une indentation systématique pour les boucles et instructions conditionnelles ainsi que pour le contenu des sous-programmes.

- Une notation en minuscules pour les données individuelles et les compteurs de boucle

- Une combinaison majuscules/minuscules pour les noms de sous-programmes et les paramètres législatifs de calcul des retraites

- Une écriture exclusivement en majuscules pour les variables de niveau macro

Un autre élément de sécurisation consiste à activer certaines directives de compilation. Par exemple, comme on le reprécisera plus loin, l'instruction `use strict` rend la prédéclaration des variables obligatoires et son usage est donc fortement recommandé. L'instruction `use diagnostics` conduit par ailleurs à l'affichage de messages d'erreurs plus nombreux et plus complets. Son usage n'est pas nécessairement recommandé en phase d'exploitation, car elle ralentit l'exécution. En revanche, elle peut être utile en phase de mise au point.

II.1.2 Types de données

Perl est un langage très peu typé. Une variable n'a pas à être prédéfinie comme étant de type caractère, entier ou réel. En fait, une variable peut prendre tour à tour les trois formes selon le contexte et les affectations. De la même manière, un tableau peut mélanger des éléments des trois types.

En revanche, un critère de typage fondamental oppose scalaires, tableaux ordinaires (dits encore listes) et tableaux associatifs dits encore « hachages ».

Le nom d'un objet scalaire commence toujours par le symbole $ (comme *scalar*) qu'il s'agisse d'un nombre ou d'une chaîne de caractère. Par exemple

```
$age  = 10 ;
$sexe = "Hom" ;
```

Il existe un nom de scalaire réservé, qui est `$_`. Le contenu de ce scalaire dépend du contexte où il est appelé : dans les applications qui nous intéresseront, il s'agira de l'indice de boucle par défaut (cf. infra).

Un tableau est une liste de scalaires donc l'indiçage commence à la valeur 0. Il faut distinguer l'objet tableau, dont le nom commence toujours par @ (comme *array*), et les éléments de ce tableau qui sont des scalaires et dont le nom commence par $.

Ainsi,

```
@statut
```

sera un tableau contenant les statuts des différents individus d'une base, mais

```
$statut[$i]
```

sera le statut de l'individu `$i`.

La gestion des tableaux sous Perl est entièrement dynamique, ce qui veut dire qu'il n'y a pas besoin de prédéfinir leur longueur, ce qui est un avantage pour des programmes de simulation travaillant sur des populations dont la taille évolue au cours du temps et est donc indéterminée *a priori*.

Comme pour les scalaires, il existe une notion de tableau par défaut, nommé `@_`. Savoir le manipuler n'est pas indispensable pour les utilisations standard des modules retraite. Il est par contre nécessaire à connaître pour les utilisations avancées, notamment pour l'utilisateur qui veut construire ses propres procédures : à l'intérieur d'un sous programme, le tableau `@_` sert en effet à stocker les valeurs des arguments transmis à ce sous programme lors de son appel. On y reviendra dans la partie VII.

On notera par ailleurs que le même nom peut être utilisé pour un scalaire pur et un tableau : `@tableau`, `$tableau[$i]` et `$tableau` renverront donc à trois entités différentes.

Un hachage est enfin un tableau dont les éléments sont indicés par des chaînes de caractère et non pas par des entiers ordonnés : on parle d'ailleurs de clés plutôt que d'indices. Le nom d'un hachage commence par %, mais, comme pour les tableaux, ses éléments sont des scalaires et sont donc nommés avec le préfixe $, avec une clé passée entre accolades plutôt qu'entre crochets. Par exemple, si on définit le hachage

```
%effectif
```

indicé par les clés `"Hommes"` et `"Femmes"`, l'objet

```
$effectif{"Hommes"}
```

pourra désigner l'effectif de la population masculine[2].

On peut construire des tableaux à double indice comme tableaux de tableaux, ou encore des hachages de hachages, des hachages de tableaux… Par exemple

```
$qmor{"Hom"}[60]
```

pourra désigner la mortalité masculine à 60 ans. Dans ce cas particulier, si on veut accéder au tableau global de ces quotients de mortalité masculins par âge, il faudra noter

```
@{$qmor{"Hom"}}
```

Cette notation un peu lourde se comprend comme suit : `$qmor{"Hom"}`, qui est un scalaire, est en fait l'adresse en mémoire du tableau qui nous intéresse. C'est pour accéder au tableau entier qu'on rajoute l'appel `@{…}` qui veut dire « retourner l'intégralité du tableau dont l'adresse est passée en argument ».

Un tableau à double entrée n'est pas nécessairement rectangulaire. Ce sera souvent le cas dans Destinie. Par exemple, le modèle gère un tableau à double indice `@statut_` qui donne les séquences par âge des statuts individuels vis-à-vis du marché du travail. Ce tableau a une forme irrégulière puisque les durées de vie des individus sont différentes.

II.1.3 Boucles et instructions conditionnelles

Les instructions conditionnelles s'écrivent de la manière suivante :

---

[2] On peut se passer des guillemets si la chaîne est une chaîne simple sans blancs

```
if ( condition )
{
   instruction
} ;
```

qui signifie que si la condition est remplie alors on effectue l'instruction.

On peut rajouter des instructions supplémentaires selon d'autres conditions à l'aide de `else` et `elsif` :

```
if (condition1)
{
   instruction1
}
else
{
   instruction2
};
```

ou

```
if (condition1)
{
   instruction1
}
elsif (condition2)
{
   instruction2
} ;
```

Chaque condition peut correspondre à une combinaison de conditions à l'aide des signes `&&` (et) et `||` (ou) :

```
if ((cond1 && cond2) || (cond3))
{
   instruction
} ;
```

qui signifie, si la condition1 et la condition2 sont remplies, ou si la condition3 est remplie, alors effectuer l'instruction.

Pour les boucles la syntaxe est similaire. La plus courante est

```
for $i (2..20,50,99)
{
   …
} ;
```

qui veut dire « effectuer les instructions entre crochets pour les valeurs de la variable `$i` comprises entre 2 et 20, ainsi que les valeurs 50 et 99 ». Il existe une forme encore plus compacte :

```
for (2..20,50,99)
{
   …
} ;
```

dans laquelle l'indice de boucle sera le scalaire `$_` qui pourra être utilisé sous ce nom à l'intérieur de la boucle.

La liste passée après l'instruction `for` peut aussi être un tableau. Si `@liste` est une liste d'identifiants d'individus présélectionnés selon un critère particulier, l'instruction

```
for (@liste)
{
   …
} ;
```

exécutera l'ensemble des instructions entre accolades pour l'ensemble des individus de la liste.

Il existe aussi des boucles `foreach`, `while`, `until` dont on ne détaillera pas ici la syntaxe.

Petit conseil pratique :
comment éviter le cauchemar des { }, [ ] et ( ) ?

Les accolades, parenthèses et crochets sont beaucoup utilisés par le langage Perl. Ces signes ont des utilités bien distinctes :

( ) encadre des conditions et des valeurs d'une boucle ou des paramètres d'une fonction.
{ } est utilisé pour délimiter des blocs d'instructions et des indices de hachages
[ ] n'est utilisé que pour préciser les indices d'une variable

La mauvaise utilisation de ces signes est source de nombreuses erreurs. Voici deux exemples de messages d'erreurs que l'on rencontre très souvent lorsqu'on programme en Perl :

```
syntax error at sortie scenarios.pl line 42, near "$i)"
```

Ce message indique à quel endroit la compilation a posé problème et s'est arrêtée. Il donne donc quelques indications sur l'endroit où chercher le problème, mais il ne dit pas que le problème est lié une parenthèse ou un crochet, même si l'expérience sur Destinie montre que le plus souvent il s'agit de ce type d'erreur.

```
Missing right curly or square bracket at sortie scenarios.pl line 218, at end of line
```

Ce message d'erreur quant à lui a l'avantage de préciser le type d'erreur, mais ne donne aucune indication sur l'endroit où le problème se situe. Il dit juste qu'arrivé à la fin, il lui manque une parenthèse (ou un crochet ou une accolade). Pour éviter ces problèmes, il faut contrôler le plus en amont possible la bonne fermeture des parenthèses, crochets et accolades. Il est aussi utile de s'astreindre aux règles d'indentation proposées et de commenter les fins des blocs les plus importants.

II.1.4 Opérateurs et traitement de chaînes de caractères

On a les opérateurs standards, y compris les opérateurs d'incrémentation de C++, de type `$i++` pour `$i=$i+1`.

S'agissant des opérateurs logiques, il faut faire attention au fait que les opérateurs de comparaison sont en principe différents pour les chaînes et les nombres, par exemple `eq` dans le premier cas et `==` dans le second. Ils fonctionnent de la même manière pour les nombres entiers, mais pas nécessairement pour des nombres réels. Par ailleurs, `==` traite de la même manière les valeurs zéro, `undef` (objet non défini) et la chaine vide `""` alors que `eq` différencie ces trois valeurs et s'avère donc préférable lorsqu'il y a risque d'ambiguïté.

Il faut par ailleurs être très attentif à l'erreur classique consistant à mettre un = simple à la place d'un ==. Soit par exemple, l'instruction :

```
for $i (1..IMax) {if ($age[$_]=60)  {Liq($_)}} ;
```

La plupart des langages diagnostiqueraient une erreur de syntaxe. La démarche de Perl est au contraire d'essayer de donner du sens à cette syntaxe. Au lieu de tester si l'âge de l'individu `$i` est égal à 60 ans, il va comprendre cette instruction comme voulant dire « affecter la valeur 60 au scalaire `$age[$i]` » puis « si le résultat de cette affectation est vrai (i.e. différent de zéro, ce qui va être le cas) alors réaliser l'instruction entre crochets ». Avec ce type d'erreur, c'est l'ensemble de la population qui se retrouve en retraite et âgée de 60 ans à la fin de la boucle `for`.

II.1.5 L'appel des sous-programmes

L'appel d'un sous-programme[3] se fait à l'aide de son nom suivi de la liste des arguments d'appel, entre parenthèses, séparés les uns des autres par des virgules.

Quelques points sont à connaître pour bien manipuler ces appels :

- Certains arguments peuvent être optionnels. Il s'agit dans ce cas des derniers arguments de la liste d'appel. Lorsque c'est le cas, ce sera précisé dans le descriptif du sous-programme.

- Beaucoup des sous-programmes des bibliothèques Destinie 2 sont appelés à l'intérieur de boucles, soit sur la période, soit sur la date. Dans une grande majorité des cas, le seul argument qui est passé à ces sous-programmes est soit cet indice de date, soit cet indice individuel. Par exemple :

  ```
  SimDir($i) ;
  ```

  indique de simuler les droits directs de l'individu `$i`. Il n'y a pas besoin de donner au sous programme les autres caractéristiques de l'individu `$i`, car elles sont stockées dans des variables globales qui contiennent toutes les informations individuelles nécessaires et qui sont visibles depuis l'ensemble des librairies Destinie 2 (voir infra section III.1). Dit d'une autre manière, il suffit de dire à Destinie sur quel individu il travaille, et les programmes se chargent du reste.

- Une précision est nécessaire sur le mode de passage des tableaux. En général, le passage d'un tableau doit se faire en indiquant son adresse. Or l'adresse du tableau `@tab` s'écrit `\@tab`. Il faudrait donc, en toute rigueur, avoir des appels de type :

  ```
  $PENSION_MOYENNE= MVar(\@pension) ;
  ```

  Il est cependant plus agréable de se contenter de la notation :

  ```
  $PENSION_MOYENNE= MVar(@pension) ;
  ```

  Ceci est possible dès lors que la spécification du sous-programme mentionne que l'argument qui lui est passé est une adresse de tableau. Dans ce cas, le sous-programme interprète `@tab` non pas comme l'ensemble du tableau, mais comme son adresse. En général, sauf mention contraire, les procédures Destinie 2 ont été spécifiées de sorte à permettre ce type d'appel.

- Il est possible de passer des instructions Perl comme argument à un sous programme. C'est ce que faisait par exemple l'instruction suivante du programme de la figure 2.

  ```
  $PENSION_MOYENNE[$t] = MVar(@pension,sub{$pension[$_]>0});
  ```

  On verra plus loin que cette possibilité est systématiquement utilisée dans les programmes de tabulation (section IV.3).

II.1.6 La documentation des programmes

L'ensemble des bibliothèques qui vont être présentées dans ce document sont fortement documentées. Les commentaires apparaissent en vert dans les programmes car il s'agit du code couleur par défaut de l'environnement PerlIDE.

---

[3] Dans tout le texte, on parlera indifféremment de sous-programmes, procédures ou fonctions.

Il en existe deux sortes :

- Des commentaires simples précédés du symbole #.
- Des commentaires précédés de séquences de contrôle en début de ligne tels que =head, =head1 (début de section) et =cut (fin de section). Cette écriture évite d'écrire un # en début de chaque ligne quand le commentaire est long. Les commentaires présentés sous cette forme peuvent aussi être extraits du programme par des utilitaires Pod2… qui les convertissent en divers types de fichiers texte (tels que .txt, .man, .tex). Une partie du présent document a été produite de cette manière.

II.2 Environnement Perl IDE

Comme la plupart des langages de même type, Perl utilise deux types de **fichiers source**,

- des fichiers correspondant aux programmes principaux. Leur nom d'extension est .pl
- des fichiers de librairies. Leur nom d'extension est .pm.

La création et la mise en œuvre de ces fichiers nécessitent de disposer d'un **environnement de développement intégré** et d'un **compilateur/exécuteur**. Plusieurs versions de l'un et l'autre sont disponibles sur le Web. Le développement de Destinie 2 s'est fait avec le compilateur ActivePerl accessible sur le site www.activestate.com et l'environnement de développement intégré Perl IDE téléchargeable sur http://www.open-perl-ide.sourceforge.net/.

En principe, mais ceci n'a pas été testé, le code source Perlde Destinie est directement transférable vers des plateformes Mac OS ou Linux (Perl est préinstallé sur tous les systèmes Unix).

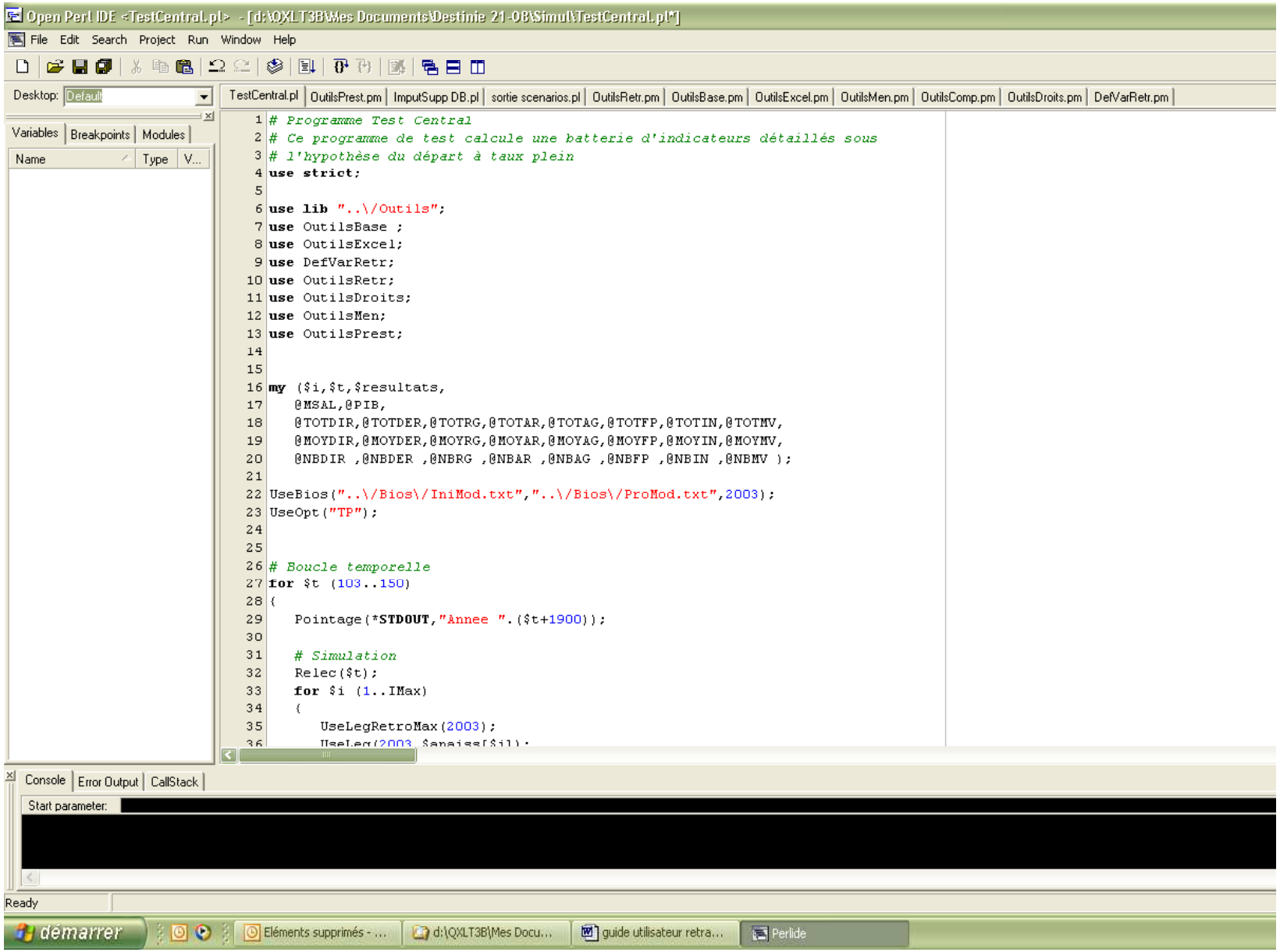
La figure 3 donne une image de l'écran de l'environnement Perl IDE. Les parties les plus utilisées pour le modèle Destinie 2 sont la partie droite de l'écran et la partie basse. La partie en haut à droite donne la liste des fichiers source sur lequel travaille l'utilisateur. Chaque onglet correspond au code source d'un programme principal ou d'une bibliothèque. On peut ouvrir autant d'onglets que l'on veut en même temps.

Pour lancer un programme plusieurs solutions existent. La solution la plus simple, une fois positionné sur le bon fichier, est de lancer la commande Run dans le menu déroulant du même nom.

Les résultats de l'exécution apparaissent dans deux fenêtres du bas de l'écran. La fenêtre Console est celle dans laquelle apparaissent les affichages écran demandés au fur et à mesure de la simulation, la fenêtre Error Output est celle où sont reportés les messages d'erreur.

Dans certains cas, on peut préférer lancer le programme Perl depuis l'extérieur de l'environnement IDE. Il faut dans ce cas que le contenu du fichier .pl ait été sauvegardé (par la commande Save du menu File). L'exécution directe de ce fichier .pl se fait en l'ouvrant avec la commande perl.exe. Ce mode d'exécution est préférable pour les programmes .pl interactifs et notamment l'explorateur. On notera qu'il n'existe pas de version .exe de cet explorateur. Perl est un langage interprété qui ne connaît que les fichiers de code source et les recompile automatiquement à chaque exécution.

Figure 3 : L'environnement Perl IDE



II.3 Organisation physique des fichiers

La figure 1 de l'introduction avait permis d'identifier les principaux éléments de la boite à outils Destinie, constituée de modules `.pm`, de l'explorateur, de fichiers de paramètres et des données individuelles récupérées du générateur de biographies. Pour que l'ensemble fonctionne, l'organisation de ces différents fichiers sur le disque de travail doit respecter quelques règles qui ont été choisies pour faciliter le portage du programme d'un ordinateur à un autre.

L'ensemble du travail se fait sur un répertoire ou un sous-répertoire du disque de travail devant comporter le mot `Destinie`, que ce soit en minuscule ou en minuscule. Ce sous-répertoire contient deux sous-répertoires obligatoires :

- Le sous-répertoire `Outils` contient l'ensemble des utilitaires `.pm` et l'explorateur `Explore.pl`. En usage standard, l'utilisateur n'accède à ce répertoire que pour l'exécution de l'explorateur, sauf s'il a créé un raccourci pointant vers ce fichier. Ce répertoire contient aussi les deux fichiers de paramètres `ParamSociaux.xls` et `ParamMortal.xls`. Il contient enfin trois sous-répertoires contenant les librairies d'interface Perl/Excel récupérées sur le site `http://www.cpan.org`[4] et qui sont utilisées par la libraire `OutilsExcel.pm`.
- Le sous-répertoire `Bios` sert à stocker les jeux de fichiers de biographies. Les biographies sont stockées dans des fichiers `BioEmp.txt` qui contiennent l'ensemble des trajectoires professionnelles et des fichiers `BioFam.txt` qui comprennent, année après année, l'ensemble des évolutions de la situation familiale. Typiquement, on pourra avoir au moins deux versions de chacun de ces fichiers : des versions de taille réduite pour les travaux de mise au point et des versions de grande taille pour les applications finales. Chaque version du couple `BioEmp`/`BioFam` devra être stockée dans un sous-répertoire différent du répertoire `Bios`.

Figure 4 : Arborescence type du répertoire Destinie 2
(en gras, noms ou éléments de noms obligatoires)

<table>
<tr><td rowspan="4">..\destinie v2</td><td rowspan="3">\Bios</td><td>\petit echantillon</td><td>BioEmp.txt
BioFam.txt</td><td rowspan="3">Fichiers de biographies</td></tr>
<tr><td>\gros echantillon</td><td>BioEmp.txt
BioFam.txt</td></tr>
<tr><td>…</td><td></td></tr>
<tr><td>\Outils</td><td colspan="2">\OLE-Storage_Lite-0.14
\Spreadsheet-ParseExcel-0.2603
\Spreadsheet-WriteExcel-2.15
Explore.pl
DefVarRetr.pm
OutilsRetr.pm
OutilsTab.pm
OutilsExcel.pm
OutilsDroits.pm
OutilsComp.pm
OutilsBase.pm
OutilsMen.pm
ParamSociaux.xls
ParamMortal.xls</td><td>Utilitaires</td></tr>
<tr><td>\Simul</td><td>\Appli 1</td><td>Simul1.pl
Resultats1.xls
Simul2.pl
Resultats2.xls
…</td><td rowspan="2">Programmes de simulation et résultats</td></tr>
<tr><td></td><td>\Appli 2</td><td>Simul.pl
Resultats.xls</td></tr>
</table>

[4] Il s'agit du site du *Comprehensive Perl Archive Network* qui mutualise l'ensemble des instruments produits pour et par la communauté Perl.

En dehors de ces deux contraintes, les programmes `.pl` de simulation et leurs résultats peuvent être stockés dans des emplacements quelconques du répertoire Destinie, mais on suggère de les cantonner dans un sous répertoire `Simul`, lui-même subdivisé en autant de sous-répertoires que de projets mobilisant le modèle.

La figure 4 donne un tel modèle d'arborescence en identifiant en caractères gras les fichiers fournis par Destinie et/ou noms de répertoires imposés et, en caractères normaux les fichiers et répertoires laissés à l'initiative de l'utilisateur.

III - Les variables

Les règles de base du langage ayant été posées, nous allons voir dans cette partie et dans les deux suivantes comment utiliser le bloc retraite *a minima*, sans rentrer dans les détails des calculs effectués par le modèle.

Cette partie est consacrée à la présentation exhaustive de la liste des variables prédéfinies gérées par le modèle Destinie et des règles à suivre pour la création de variables additionnelles. On commencera par expliciter la différence entre les notions de variables globales et locales (section III.1). Les variables prédéfinies par Destinie sont du premier type alors que les variables créées par l'utilisateur sont de deuxième type. On donnera ensuite une vue générale du type d'information que requiert une simulation de retraite (section III.2) ce qui permettra de mieux comprendre le pourquoi des différents types de variables prévus par Destinie dont la liste sera présentée exhaustivement section III.3. Puis on indiquera dans quel cas et de quelle manière on est amené à créer des variables additionnelles propres à chaque simulation (section III.4).

III.1 Préliminaire technique : les notions de variables locales et globales

Avant de détailler comment initialiser des variables dans Destinie 2, il est essentiel de comprendre les différents types de variables que gère Perl en général et que gère ce modèle en particulier. Le type de variables dépend de l'endroit où elles sont utilisées et de la façon dont elles sont initialisées. Perl permet de déclarer des variables à deux niveaux :

- Les variables locales qui ne sont utilisables que depuis le programme principal ou le sous-programme où elles ont été déclarées[5].
- Des variables globales partagées entre les différents modules et le programme principal.

Les variables locales sont déclarées par l'instruction `my` suivie de la liste de variables entre parenthèses, par exemple `my ($var1, @var2)`. Le terme `my` fait bien ressortir que ces variables sont propres au bloc de texte dans lequel elles sont déclarées. On a vu dans l'exemple de l'introduction que c'est cette instruction `my` qui sert à déclarer les variables du programme `.pl` qui ne sont pas des variables par défaut de Destinie.

Les bibliothèques contiennent également de grands nombres de variables locales qui sont uniquement destinées à des calculs internes à ces bibliothèques, et qui ne seront donc pas visibles depuis le programme principal, parce qu'on a considéré que cette visibilité n'était pas nécessaire. Par exemple, il existe dans la bibliothèque `OutilsDroits` une fonction `SalBase`, qui calcule le salaire de référence du régime général, i.e. la moyenne des salaires des meilleures années de la carrière. Pour ce faire, la fonction `SalBase` gère un tableau intermédiaire de salaires qui seront triés par ordre décroissant avant calcul de leur moyenne sur le nombre d'années choisi. Un tel tableau n'a pas à être conservé de manière permanente et il s'agit donc d'un tableau local à cette fonction `SalBase`, dont le contenu est détruit sitôt que Perl sort de cette fonction.

La déclaration en variables globales a été réservée à l'ensemble des variables dont on peut avoir besoin dans le programme principal. Pour ce faire, elles sont déclarées dans le module `DefVarRetr` par une instruction `our`, telle que `our (@var1, @var2..)` mais elles nécessitent aussi l'appel à une procédure d'exportation pour être visibles depuis d'autres modules. Plus précisément, `DefVarRetr` commence, sous forme très abrégée, par une séquence d'instructions de type :

---

[5] Le terme Perl exact est variables lexicales, i.e. dont la portée correspond à un certain bloc de texte du code source. Ce terme sert à les différencier d'un autre concept de variable locale qui existait dans les premières versions du langage. On utilisera néanmoins le terme de local car plus suggestif et plus familer.

```
require Exporter;
our @ISA    = qw (Exporter);
our @EXPORT = qw (@anaiss @sexe @findet @pere @mere @matri @conjoint);
```

qui signifie que `DefVarRetr` met les variables `@anaiss`, `@sexe`… à disposition de tous les programmes ou autres modules qui feront appel à lui.

Cette syntaxe est un peu complexe, mais sa maitrise n'est utile qu'à l'utilisateur qui souhaiterait développer ses propres modules et nous n'y reviendrons donc qu'à la partie VII. Pour les utilisations courantes, il suffit de savoir que `DefVarRetr` contient un jeu d'instructions de ce type qui met à disposition de l'utilisateur un certain nombre de variables. Le point important est surtout de savoir quelles sont ces variables, comment elles sont alimentées et comment elles peuvent être utilisées. C'est ce à quoi sont dédiées les sous-sections suivantes. Avant cela, il faut se poser la question de ce dont a besoin, en règle générale, un programme de calcul de droits à retraite. C'est l'objet de la section III.2.

III.2 De quelles informations a-t-on besoin pour simuler les retraites ?

Le calcul des pensions de retraite relève de règles complexes et variées selon les régimes de retraite. La simulation des pensions et des départs à la retraite nécessite de disposer de nombreuses informations sur les individus et sur la réglementation en vigueur. Nous ne verrons pas dans ce document la réglementation détaillée et les différents modes de calcul des pensions. Pour faciliter la lecture de ce guide, on va néanmoins rappeler les bases générales de ces calculs, afin de bien comprendre de quelles informations il est nécessaire de disposer dans le cadre du modèle Destinie.

La carrière professionnelle

Les pensions de base au régime général et dans la fonction publique, correspondent à une proportion d'un montant de salaire. Le montant de ce salaire utilisé dans le calcul de la pension dépend de la carrière des individus. Il peut s'agir d'un salaire annuel moyen (SAM) sur les n meilleures années de salaires, du dernier salaire selon le régime. L'importance de la part du salaire versée dépend des périodes pendant lesquelles l'individu peut être considéré comme affilié à ce régime : les durées validées dans ce régime.

Les salaires perçus et les périodes de la carrière (en emploi, au chômage, dans la fonction publique, au régime général…) sont donc les éléments clés pour le calcul de la retraite. Les durées prises en compte pour la retraite ne correspondent pas seulement aux années travaillées, mais aussi aux périodes de chômage et de préretraite. Pour le calcul du salaire à prendre en compte, il est nécessaire de disposer d'informations sur les salaires perçus par chaque individu tout au long de sa carrière, mais aussi des allocations de chômage et de préretraite, puisque ces périodes sont également prises en compte. Un point important est que les règles de calcul des retraites s'appliquent à des salaires nominaux. De manière générale, le séries gérées ou produites par Destinie sont des séries nominales. Les convertir en données réelles imposera de les rediviser par l'indice des prix, mais ceci est facilité par le fait que cet indice des prix fait partie des variables fournies par le modèle.

Les situations familiales

Le système de retraite accorde également des droits spécifiques aux individus selon leur situation familiale. Si la personne a eu des enfants, elle bénéficie d'une majoration de la durée validée (MDA). Si elle s'est arrêtée de travailler pour élever ses enfants, elle a le droit de valider certaines de ces années par le biais de l'allocation vieillesse aux parents au foyer (AVPF). Il faut donc disposer d'informations sur les enfants de chaque individu. De plus une pension de réversion peut être versée au conjoint survivant lorsqu'un individu décède. Il faut donc connaître la situation de couple des personnes dont on simule les retraites, en particulier connaître leur conjoint éventuel pour pouvoir, le moment venu, lui affecter une pension de réversion.

La législation

Ces principes généraux du calcul des retraites sont nettement plus complexes lorsqu'on applique la législation propre à chaque régime. De plus, le système de retraite a connu de nombreuses réformes au cours du temps et toutes les générations ne se voient pas appliquer les mêmes règles de calcul. Chaque année, la valeur de certains éléments de calcul de la pension sont également revalorisés : le plafond de la sécurité sociale, les valeurs des points… Pour simuler les retraites dans le temps comme le fait le modèle Destinie, il est nécessaire d'intégrer toutes les composantes de la législation pour tous les régimes, toutes les années et toutes les générations.

Les modalités de départ à la retraite

Enfin, le modèle Destinie simule le moment où un individu part à la retraite. Dans la réalité, le moment du départ résulte d'un choix individuel complexe (selon des critères financiers mais également des préférences personnelles) et de règles imposées aux individus (âge minimum, âge maximum…). Un des intérêts de Destinie est de permettre de simuler différentes méthodes de détermination du moment du départ à la retraite. Par exemple on peut imposer que tous les individus liquident leur pension lorsque le taux de liquidation est maximum et pas avant ni après, ou bien imposer un âge identique pour tous, ou proposer des modes de choix plus libres et plus complexes. Pour cela le modèle doit disposer de tous les éléments permettant de mettre en œuvre ces différents modes de départ à la retraite.

III.3 Les variables prédéfinies (bibliothèque DefVarRetr)

III.3.1 Vue d'ensemble

De très nombreuses variables sont prévues par Destinie pour stocker et gérer l'ensemble de ces informations. Ces variables sont définies par le module `DefVarRetr.pm` et sont essentiellement des variables individuelles et des paramètres de calcul des droits à retraite. L'utilisateur n'a pas à les redéclarer, mais il a besoin de les connaître pour pouvoir les utiliser. Plus précisément, on distinguera six rubriques :

- Les variables individuelles de type tableau qui vont servir d'input aux simulations de retraite. Il va s'agir pour l'essentiel de données récupérées du générateur de biographies, et aussi de quelques paramètres de comportement auxquels ce module attribue des valeurs par défaut, modifiables au besoin. Ces opérations de récupération ou d'initialisation mobiliseront deux programmes fournis dans `OutilsRetr` : les programmes `UseBios` et `Relec`.

- Des variables individuelles d'output, également de type tableau, et destinées à accueillir les résultats de la simulation des droits à retraite. Le calcul de ces droits constitue l'objet principal de la simulation. Il est réalisé à l'aide des autres fonctions de la bibliothèque `OutilsRetr`. `DefVarRetr` se contente de définir ces variables sans leur donner de valeurs par défaut.

- Des variables individuelles de travail de type scalaire. Contrairement aux deux groupes précédents qui sont des variables indicées par `$i`, ces variables sont des scalaires utilisés temporairement pour stocker des intermédiaires de calcul. Leurs valeurs sont écrasées lorsque le programme passe à la simulation de l'individu suivant. Ce système économise la place mémoire. Mais, en fonction des besoins, on peut consulter ces variables en cours de simulation ou créer des variables permanentes dans le programme appelant pour en stocker les résultats.

- Les paramètres de calcul des droits à retraite qui se présentent sous forme de séries temporelles lues dans des fichiers Excel. Les valeurs par défaut de ces paramètres sont lues par `DefVarRetr`, et peuvent être modifiées ensuite au gré des scénarios.

- Des paramètres de calcul des droits à retraite plus complexes, traités comme scalaires. Leurs valeurs sont fixées par appel du programme `UseLeg`, lui aussi fourni dans `OutilsRetr`.

- Il existe enfin divers types de variables auxiliaires, telles que des variables stockant les codes utilisés pour les variables individuelles qualitatives, ce qui dispense de connaître les valeurs numériques de ces codes, ou encore les noms des fichiers ou des répertoires de travail, ainsi que diverses options de simulation.

Ces six groupes de variables vont être examinés dans les sous-sections III.3.2 à III.3.7. Les listes complètes n'en seront pas systématiquement données. Le lecteur est renvoyé aux annexes pour ces listes complètes, données soit sous forme alphabétique soit par catégorie.

Indépendamment de `DefVarRetr`, l'utilisateur peut vouloir rajouter à cette liste d'autres variables qui lui seront utiles. La section III.4 sera consacrée aux exemples de variables micro-additionnelles ou variables macro qu'on est amené à déclarer et utiliser dans le programme principal.

III.3.2 Variables individuelles d'input

Les variables individuelles d'input sont les variables individuelles sur lesquelles va s'appuyer la simulation des droits à retraite de chaque personne de l'échantillon. Leurs valeurs seront renseignées par la procédure `Relec` de la bibliothèque `OutilsRetr` qui sera présentée plus loin.

Ce groupe de variables comprend lui-même trois sous-groupes.

- les variables lues directement dans les fichiers issus du générateur de biographies,
- quelques variables dérivées des précédentes, calculées ou imputées par `Relec`,
- les variables de préférences utilisées dans certains types de simulation des comportements de départs à la retraite. Elles ne sont pas issues du générateur de biographies. `Relec` se charge de les initialiser à des valeurs par défaut.

Même si une bonne part de ces variables n'ont pas à être utilisées directement en fonctionnement standard, on donne leur liste complète, compte tenu de leur rôle central dans le programme, et aussi parce que c'est l'occasion de donner l'aperçu de ce que produit le générateur de biographies. Il s'agit dans tous les cas de tableaux indicés par l'identifiant individuel, qu'on notera en général `$i`, l'indiçage commençant à la position 1 (il n'y a pas d'individu 0) et éventuellement par un deuxième indice. On donne directement le format d'appel de leurs éléments scalaires, plutôt que leur nom de tableau commençant par `@`.

Le premier sous-groupe comprend :

`$anaiss[$i]`	: l'année de naissance
`$sexe[$i]`	: le sexe
`$findet[$i]`	: l'âge de fin d'études
`$pere[$i]`	: l'identifiant du père
`$mere[$i]`	: l'identifiant de la mère
`$matri[$i]`	: le statut matrimonial
`$conjoint[$i]`	: l'identifiant du conjoint
`$enf[$i][$e]`	: les identifiants des enfants
`$statut_[$i][$a]`	: la séquence des statuts passés par rapport à l'emploi
`$salaire_[$i][$a]`	: la séquence des salaires nominaux passés par âge
`$taux_prim[$i]`	: le taux de prime si l'individu travaille dans la fonction publique

Le deuxième sous-groupe comprend :

`$trim[$i]`	: le trimestre de naissance

`$age[$i]`	: l'âge courant
`$present[$i]`	: une indicatrice de présence dans l'échantillon à la date courante (entre la naissance ou la migration et le décès)
`$statut[$i]`	: le statut courant (i.e. `$statut_[$i][$age[$i]]`)
`$salaire[$i]`	: le salaire nominal courant (i.e. `$salaire_[$i][$age[$i]]`)
`$deces[$i]`	: indique si l'individu est décédé l'année en cours
`$migrant[$i]`	: indique si l'individu est un migrant de l'année

Le troisième sous-groupe comprend enfin :

`$taux[$i]`	: la préférence pour le présent
`$k[$i]`	: la préférence pour le loisir/coefficient de pénibilité du travail
`$gamma[$i]`	: l'aversion pour le risque
`$seuil[$i]`	: seuil de déclenchement du report du départ en retraite

La signification exacte des variables `@taux`, `@k`, `@gamma` et `@seuil` apparaitra plus loin. Il s'agit de variables commandant le comportement de départ en retraite. Elles sont initialisées à des valeurs par défaut mais modifiables par le programme appelant si ce paramétrage ne convient pas ou si on veut introduire une hétérogénéité individuelle de ces paramètres.

Concernant les autres variables, deux points sont à préciser.

Tout d'abord, les identifiants des personnes apparentées à l'individu `$i` peuvent eux-mêmes être utilisés comme indices. Cela signifie que Destinie accepte des syntaxes de type `$age[$pere[$i]]` qui donne directement l'âge du père de l'individu `$i`. Ce type d'appel n'est pas courant en utilisation standard, mais on devine facilement les utilisations qui peuvent en être faites. Par exemple, les procédures de la bibliothèque `OutilsMen` qui sera présentée en partie IV utilisent cette fonctionnalité.

Le second point concerne les variables à double-indice. Il y a deux cas de figure.

- Le premier est celui de la variable `@enf`, dont le premier indice est l'individu, et le deuxième indice est le rang de naissance de l'enfant. Cela veut dire que le tableau `@{$enf[$i]}` donne directement la liste des identifiants des enfants de l'individu `$i`. La bibliothèque `OutilsMen` qui sera présentée en partie VI utilise là encore cette fonctionnalité.
- Le second cas est celui des variables rétrospectives. Le principe est de leur donner un nom se terminant par un blanc souligné. Par défaut, Destinie en gère deux, celles qui sont indispensables aux calculs de droits à retraite. Il s'agit des variables `@salaire_` et `@statut_` qui donnent un rétrospectif des carrières. Pour l'individu `$i`, le statut et le salaire à l'âge `$a` sont accessibles par `$statut_[$i][$a]` et `$salaire_[$i][$a]`. Les séquences complètes correspondantes sont les tableaux simples `@{$statut_[$i]}` et `@{$salaire_[$i]}`. Les procédures de calcul des retraites font un large appel à ces données rétrospectives.

À ces variables `@statut_` et `@salaire_` sont associées les variables simples `@statut` et `@salaire` qui donnent le salaire et le statut courants. Ces deux variables n'apportent aucune information supplémentaire : on a redondance entre `$statut[$i]` et `$statut_[$i][$age[$i]]`, et entre `$salaire[$i]` et `$salaire_[$i][$age[$i]]`. Mais ces deux variables ont été introduites car elles donnent le statut et le salaire courant dans des formes qui seront utilisables par les procédures de tabulation, alors que les tableaux à double indice `@statut_` et `@salaire_` ne le sont pas.

Destinie n'a pas prévu d'autres variables à double indice pour ne pas saturer d'office la place mémoire. Mais l'utilisateur peut en créer d'autres en fonction de ses besoins. Par exemple, un programme pourrait créer et renseigner une variable `$conjoint_[$i][$a]` qui donnerait l'historique des conjoints successifs de cet individu, si cette information s'avérait nécessaire.

III.3.3 Variables individuelles d'output

Ces variables sont construites sur le même modèle que les variables présentées à la section précédente, mais à la différence des variables initialisées par `Relec`, il s'agit de variables qui vont découler de la simulation des retraites. Ici encore on donne leur liste complète, car c'est en général sur ces variables que porteront les demandes de tabulation. Il est donc nécessaire de toutes les connaître. Il s'agit de :

`$pension_rg[$i]`	: pension régime général
`$pension_ar[$i]`	: pension arrco
`$pension_ag[$i]`	: pension agirc
`$pension_fp[$i]`	: pension fonction publique
`$pension_in[$i]`	: pension d'indépendant
`$pension_cn_pri[$i]`	: pension comptes notionnels du secteur privé (si existe)
`$pension_cn_fp[$i]`	: pension comptes notionnels du secteur public (si existe)
`$pension[$i]`	: pension directe totale (somme des sept précédentes)
`$rev_rg[$i]`	: réversion régime général
`$rev_ar[$i]`	: réversion arrco
`$rev_ag[$i]`	: réversion agirc
`$rev_fp[$i]`	: réversion fonction publique
`$rev_in[$i]`	: réversion d'indépendant
`$pension_cn_pri[$i]`	: réversion comptes notionnels du secteur privé (si existe)
`$pension_cn_fp[$i]`	: réversion comptes notionnels du secteur public (si existe)
`$rev[$i]`	: réversion totale (somme des sept précédentes)
`$min_vieil[$i]`	: allocation du minimum vieillesse
`$ageliq[$i]`	: âge de la liquidation
`$pliq[$i]`	: pension à la liquidation

Cette liste n'est pas figée. L'utilisateur peut souhaiter rajouter des variables de même type ou bien des variables dérivées à double indice sur le modèle présenté à la section précédente. Par exemple une variable `$pension_[$i][$a]` peut servir à stocker la séquence des pensions successives de chaque individu et servir à des calculs de somme de droits actualisés durant la retraite ou à produire des profils d'évolution du niveau de vie par âge.

Un autre exemple de variable additionnelle sont les variables d'archivage. Une variable d'archivage peut par exemple conserver les valeurs de `@ageliq` obtenues dans une simulation de référence et les utiliser pour des variantes à âge de la retraite exogène, pour simuler l'impact de réformes à comportements inchangés. Les variables d'archivage peuvent aussi permettre la comparaison des résultats de scénarios individus par individus.

On reviendra sur les différents exemples possibles de variables additionnelles en partie III.4.

III.3.4 Variables individuelles temporaires

Les variables de retraite qu'on vient d'indiquer sont des variables de résultats finaux. Le calcul des retraites inclut aussi la production de beaucoup d'éléments intermédiaires, par exemple le SAM et la durée cotisée pour le régime général, ou encore la durée tous régimes.

Il n'a pas été prévu que ces variables intermédiaires soient systématiquement gardées dans des tableaux indicés par `$i`. Elles sont stockées dans des variables scalaires, ce qui veut dire que l'information est perdue lorsque le programme procède à un nouveau calcul de retraite pour un autre individu. Mais il peut-être utile et il est possible d'y accéder avant ce passage à l'individu suivant. À l'intérieur d'une boucle sur `$i`, elles sont visibles depuis le programme principal pour cet individu `$i` tant que l'on n'est pas passé à l'individu suivant de la boucle.

Ici encore, on donne leur liste complète :

`$duree_rg`	: durée cotisée au régime général
`$duree_fp`	: durée cotisée dans les régimes du secteur public
`$duree_fpa`	: idem, en service actif
`$duree_fps`	: idem, en service sédentaire
`$duree_in`	: durée cotisée en régime d'indépendant
`$duree_tot`	: durée cotisée totale
`$duree_avpf`	: durée de bénéfice de l'AVPF
`$duree_rg_maj`	: durée RG incluant AVPF et MDA (selon options)
`$duree_fp_maj`	: durée FP incluant MDA éventuelle (selon options)
`$duree_in_maj`	: durée indépendant incluant MDA éventuelle (selon options)
`$duree_tot_maj`	: durée totale incluant AVPF et MDA (selon options)
`$duree_cho`	: durée passée au chômage
`$duree_emp`	: durée passée en emploi
`$duree_PR`	: durée passée en préretraite
`$sam_rg`	: SAM servant au calcul de la pension RG
`$sam_in`	: SAM pour le calcul de la pension d'indépendant
`$sr_fp`	: salaire de référence pour le calcul de la pension FP
`$points_arrco`	: cumul de points ARRCO
`$points_agirc`	: cumul de points AGIRC
`$points_cn_pri`	: cumul de points comptes notionnels secteur privé
`$points_cn_fp`	: cumul de points comptes notionnels secteur public
`$min_cont`	: minimum contributif
`$min_garanti`	: minimum garanti
`$id_clone`	: id de l'individu de départ lorsqu'on travaille sur un clone

Ces variables peuvent soit être affichées à la volée (par exemple dans une phase de débogage où on a identifié un problème sur un individu particulier) soit être stockées dans un tableau qu'on aura pris soin de prédéclarer, et qui peut d'ailleurs prendre le même nom puisque Perl sait faire la différence entre un tableau `@x`, son `$i`-ème élément `$x[$i]` et le scalaire `$x`.

Autrement dit, on peut avoir, dans le programme appelant :

```
my (@points_arrco);

for $t..
{
   for $i...
   {
      Simulation
      $points_arrco[$i]=$points_arrco;
   }
}
```

après quoi la distribution des points ARRCO peut être calculée. Ce genre de manipulation suppose évidemment de bien savoir de quelle façon les différents programmes manipulent ces variables tampon.

III.3.5 Paramètres temporels lus dans des fichiers Excel.

Pour calculer les droits à retraite et procéder aux simulations de comportement qui vont alimenter ces variables d'output, la bibliothèque `OutilsRetr` et les bibliothèques qu'elle appelle vont avoir besoin de nombreux paramètres. Une partie de ces paramètres sont stockés dans deux fichiers Excel, et relus directement par `DefVarRetr`. Il s'agit des fichiers `ParamSociaux.xls` et `ParamMortal.xls`. Ces deux fichiers doivent se trouver sur le même répertoire que celui où figure `DefVarRetr.pm`.

Le fichier ParamSociaux.xls contient des éléments de barèmes qui ont la forme de séries temporelles simples[6]. Pour ces séries temporelles, la convention d'indiçage est de prendre l'année 1900 comme année de départ. Autrement dit l'indice 0 correspond à l'année 1900. Par exemple $PlafondSS[98] est le plafond de la sécurité sociale de l'année 1998, $PlafondSS[102] est celui de l'année 2002. On reprécise que, comme toutes les autres données monétaires gérées par Destinie, il s'agit de valeurs nominales, à la conversion près en euros des francs et anciens francs. Elles peuvent être reconverties en valeurs réelles en utilisant l'indice des prix également fourni dans le même fichier.

Ce fichier ParamSociaux.xls comprend sept feuilles correspondant à autant de groupes de paramètres :

- Paramètres généraux (ParamGene) : indice des prix, SMIC, plafond de la sécurité sociale et valeur du point dans la fonction publique.

- Paramètres de calcul des retraites de base et des pensions minimales (ParamRetrBase) : coefficients de revalorisation des pensions, taux minima, coefficient pour la MDA. Ces paramètres sont nécessaires au calcul de la pension de droit direct de base.

- Paramètres des régimes complémentaires (ParamComp) : salaires de référence, taux de cotisations, valeurs du point…

- Paramètres de calcul des réversions (ParamRev) : minimum, maximum, taux de réversion.

- Paramètres de calcul des préretraites (ParamPreRet) : taux de préretraite et âge d'accès.

- Paramètres des prestations familiales (ParamFam) : base mensuelle des allocations familiales et plafonds de ressources pour diverses prestations familiales (complément familial, APJE, PAJE…). Ces paramètres ne sont pas directement utilisés pour le calcul des pensions de retraite, mais peuvent être utiles pour un module sur les prestations sociales ou sur les revenus de façon plus générale.

- Autres paramètres (ParamAutres) : il s'agit de divers taux de cotisation sociale, autres que les cotisations retraite, servant aux calculs des salaires et pensions nettes.

La liste détaillée de ces paramètres se trouve en annexe de ce document.

Les valeurs lues dans ce fichier doivent être vues comme des valeurs de référence qui peuvent faire l'objet de modifications dans le programme appelant selon les besoins des scénarios. Par exemple, en projection, il est possible d'avoir une instruction :

```
$PlafondSS[$t]=1.03*$PlafondSS[$t-1];
```

qui imposerait une règle d'évolution du plafond différente de celle prévue par défaut dans le fichier ParamSociaux (cette hypothèse est une indexation sur la productivité, à savoir, par défaut, 1,5 % par an).

Le fichier ParamMortal.xls contient les tables de mortalité qui sont nécessaires aux calculs de droits à retraite actualisés et/ou aux fonctions d'utilité intertemporelles. On dispose à la fois des quotients de mortalité et des taux de survie qui sont tous deux stockés sous forme des tableaux indicés par le sexe, la date et l'âge, de termes génériques $quotient[$s][$t][$a] et $survie[$s][$t][$a].

La lecture des deux fichiers ParamSociaux.xls et ParamMortal.xls se fait grâce à la bibliothèque OutilExcel. Les procédures correspondantes ne sont pas à connaître pour les utilisations standard. Contrairement aux procédures d''écriture dans des fichiers Excel qui

[6] Une part importante des barèmes retrospectifs est reprise de Bozio (2008).

seront vues plus loin, elles ne sont pas détaillées dans cette présentation des utilisations usuelles du modèle et le lecteur intéressé par la description de ces procédures est renvoyé à la partie V.7.

III.3.6 Paramètres scalaires générés par la fonction Useleg.

Pour certains paramètres de calcul (surcote, décote, durée cible, nombre d'années prises en compte pour le SAM…), le stockage dans des fichiers Excel s'est avéré peu pratique. Il s'agit pour l'essentiel de paramètres dépendant à la fois de la période et de la génération, pour lesquels il aurait fallu prévoir des tableaux à double indice et des fichiers Excel de taille importante mais comportant beaucoup de cases redondantes pour les périodes de stabilité des barèmes.

On a donc plutôt choisi de traiter ces paramètres comme des scalaires simples, dont Destinie recalcule la valeur pour chaque cas individuel. Il le fait à l'aide d'une fonction fournie par la bibliothèque `OutilsRetr` qui est la fonction `UseLeg`. Cette fonction joue un rôle crucial pour le choix des scénarios et elle sera analysée plus loin (section IV.2.3).

Comme pour les paramètres lus dans les fichiers Excel, la liste détaillée de des paramètres générés par la fonction `UseLeg` se trouve en annexe de ce document.

III.3.7 Autres paramètres

Parmi les autres variables ou paramètres, on dispose tout d'abord de variables stockant les codes utilisés pour les variables micro-qualitatives. Par exemple, pour le sexe, au lieu d'identifier les hommes par le code 1 et les femmes par le code 2, on pourra les identifier respectivement par `$CodeHom` et `$CodeFem`. Cette codification est particulièrement utile pour les statuts professionnels qui comprennent beaucoup de modalités et sont souvent utilisées par les différents programmes.

Figure 5 : Liste des codes présents dans le modèle Destinie 2

Type de code	Nom	Signification	Valeur(s)
Codes sexe	$CodeHom $CodeFem	Homme Femme	1 2
Codes statut matrimonial	$CodeCel $CodeMar $CodeVeu $CodeSepar	Célibataire Marié(en couple) Veuf Séparé	1 2 3 4
Codes statut d'emploi	$CodeNC $CodeCad $CodeFPS $CodeFPA $CodInd $CodeCho $CodeIna $CodePr $CodeAvpf	Non Cadre du privé Cadre du privé Fonction publique sédentaire Fonction publique active Indépendant Chômeur Inactif Pré retraité Inactif avec AVPF	1 2 31 32 4 5 6 7 8
Codes regroupés	@CodesAct @CodesOcc @CodesFP	Actifs Actifs occupés Fonction publique	1,2,31,32,4,5 1,2,31,32,4 31,32

Ces variables permettent à l'utilisateur de construire des tabulations sans se soucier des conventions de codage numérique internes à Destinie. Par exemple, on verra plus loin qu'on pourra écrire :

```
$SALMOYNC        = MVar (@salaire,sub{$statut[$_]==$CodeNC});
```

pour calculer le salaire moyen des salariés du privé non cadres, ou encore :

```
$TAUXVEUVAGE[$a] = Taux (sub{$matri[$_]==$CodeVeu},
                         sub{$age[$_]==$a});
```

où `$CodeNC` = non cadre du secteur privé et `$CodeVeu` = veuf ou veuve.

Les codes regroupés permettent des écritures simplifiées, par exemple :

```
$PACT     = Count(sub{In($statut[$_],@CodesAct)});
$SALMOYFP = MVar (@salaire,sub{In($statut[$_],@CodesFP )});
```

où `@CodesAct` et `@CodesFP` sont des familles de codes regroupant respectivement l'ensemble des cas d'activité et l'ensemble des cas d'emploi public (actifs et sédentaires).

L'utilisateur peut générer d'autres types de codes regroupés dans le programme appelant en fonction de ses besoins.

On notera l'absence de code spécifique pour les enfants d'âge scolaire ou les retraités. Les premiers sont classés comme inactifs ou en emploi s'il y a cumul emploi-études (le générateur de biographies génère des individus qui travaillent avant d'avoir atteint leur âge de fin d'études).

Pour les seconds, pour éviter toute ambiguïté sur la notion de retraité, le fait d'être repéré comme tel tient uniquement au fait de toucher une pension positive plutôt qu'à un code statut spécifique. Éventuellement, le code statut d'un bénéficiaire d'une retraite pourrait être mis à une autre valeur que `$CodeIna`, dans des versions du modèle qui autoriseraient le cumul emploi et retraite.

Il est également possible d'afficher les codes en clair, grâce au hachage prédéfini `%Label`. Par exemple, si un individu est dans le statut 31 à l'âge `$a`, l'instruction

```
print $Label{"statut"}[$statut_[$i][$a]];
```

affichera la chaine `FPA`. Cette fonctionnalité est utilisée dans les affichages de l'explorateur qui sera examiné plus loin.

On mentionnera pour finir diverses variables de contrôle de la simulation mais, en utilisation standard, il est surtout utile de connaître les fonctions qui les renseignent puisqu'il n'est en général pas nécessaire d'y accéder directement. On renvoie donc aux tableaux de l'annexe pour leur liste détaillée. Parmi ces variables figurent notamment les noms des répertoires et fichiers de travail, directement reconstitués par le programme ou déterminés par la fonction `UseBios`, ainsi que des informations sur les données de départ également renseignées lors de l'appel de la fonction `UseBios` (taux de sondage et année de base).

III.4 Déclaration de variables micro ou macro additionnelles

À côté des variables globales initialisées dans `DefVarRetr` et directement mobilisables, l'utilisateur est en général amené à créer d'autres variables propres au programme `.pl` qu'il construit pour sa simulation.

Il faut d'abord qu'il définisse par lui-même les différents indices de boucle dont il aura besoin (*a minima* un indice `$i` et un indice `$t`).

Il faut ensuite qu'il définisse les variables stockant les résultats macro, c'est-à-dire en général des séries temporelles stockant des résultats globaux année après année, qui seront renseignées à l'aide des procédures de tabulation vues plus loin en IV.3, par exemple `@PENSION_MOYENNE` pour la série temporelle de la retraite moyenne, `@RATIO_DEM` pour la projection du ratio retraités/actifs. Aucune de ces variables n'a été prédéfinie en standard, car elles sont complètement dépendantes de l'exercice que l'utilisateur souhaite réaliser

On peut aussi être amené à déclarer des variables micro additionnelles. On a déjà vu plusieurs raisons de le faire :

- Pour récupérer des résultats de calculs intermédiaires des retraites et les conserver pour chaque individu de la population.

- Pour constituer des tableaux à double-indice de données rétrospectives sur le modèle de `@statut_` ou `@salaire_`

- Pour conserver le résultat d'un scénario en vue d'une réutilisation dans un autre scénario du même programme : par exemple l'âge de liquidation pour réutilisation dans un scénario où on contraindra les individus à partir au même âge, ou le niveau de pension pour des comparaisons individuelles avec le niveau observé dans les variantes.

On peut encore imaginer beaucoup d'autres cas de figure

- Pour créer des descripteurs sociodémographiques non prévus au départ par `Destinie`. Par exemple, si on veut utiliser le nombre d'enfants, il faut initialiser une variable `@nenf` qu'on pourra renseigner à l'aide des procédures proposées dans la bibliothèque `OutilsMen`.
- Pour créer des variables de retraite non standard : par exemple des taux de remplacement, des parts des différentes composantes de la pension dans la pension totale, un compteur du nombre de droits différents dont profite un individu, ou de son apport au revenu global du ménage, etc…

Enfin, mais ceci relève plutôt des utilisations très avancées, l'utilisateur peut être amené à ajouter des paramètres additionnels de calcul des retraites, ou des paramètres gouvernant l'évolution d'autres comportements que le départ en retraite.

Dans tous les cas, il faut déclarer ces variables à l'aide d'une instruction `my` en début de programme (voir III.1). Avec l'option `use strict`, l'omission de cette déclaration conduit en effet à un message d'erreur dès qu'on cherche à utiliser la variable, du type :

```
Global symbol "@MSAL " requires explicit package name….
```

Lorsqu'on crée ces variables additionnelles, il est suggéré de se tenir aux conventions d'écriture adoptées dans l'ensemble des procédures du modèle, à savoir :

- des minuscules pour les variables micro simples (comme `@statut`)
- des minuscules suivies d'un blanc souligné pour les variables micro à double indice (comme `@statut_`)
- une combinaison de majuscules et minuscules pour les paramètres législatifs (comme `@ValPtARRCO` pour la valeur du point ARRCO)
- des majuscules pour les variables macro (`@MSAL`)

Le respect de ces règles n'est évidemment pas indispensable, mais il est recommandé pour les programmes destinés à être partagés entre plusieurs utilisateurs (notamment en cas de développement de nouveaux modules).

IV - Utilisations courantes (1) : effectuer une simulation et éditer ses résultats

Les différentes catégories de variables Destinie ayant été précisées, on va maintenant pouvoir analyser en détail le contenu d'un programme de simulation.

Comme on l'a vu en introduction, la production d'une simulation nécessite l'écriture d'un programme principal (i.e. un fichier `.pl`) spécifiquement dédié à cette simulation dont la structure type est donnée par la figure 6. Sous sa forme la plus courante, il contiendra quatre éléments principaux :

- Un ensemble d'initialisations (préambule) : appels des bibliothèques, déclarations de variables autres que celles définies par défaut, ouvertures de fichiers ou choix de certaines options générales de simulation.

- La simulation des retraites proprement dite, qui prend la forme d'une boucle temporelle et d'une boucle individuelle emboîtées

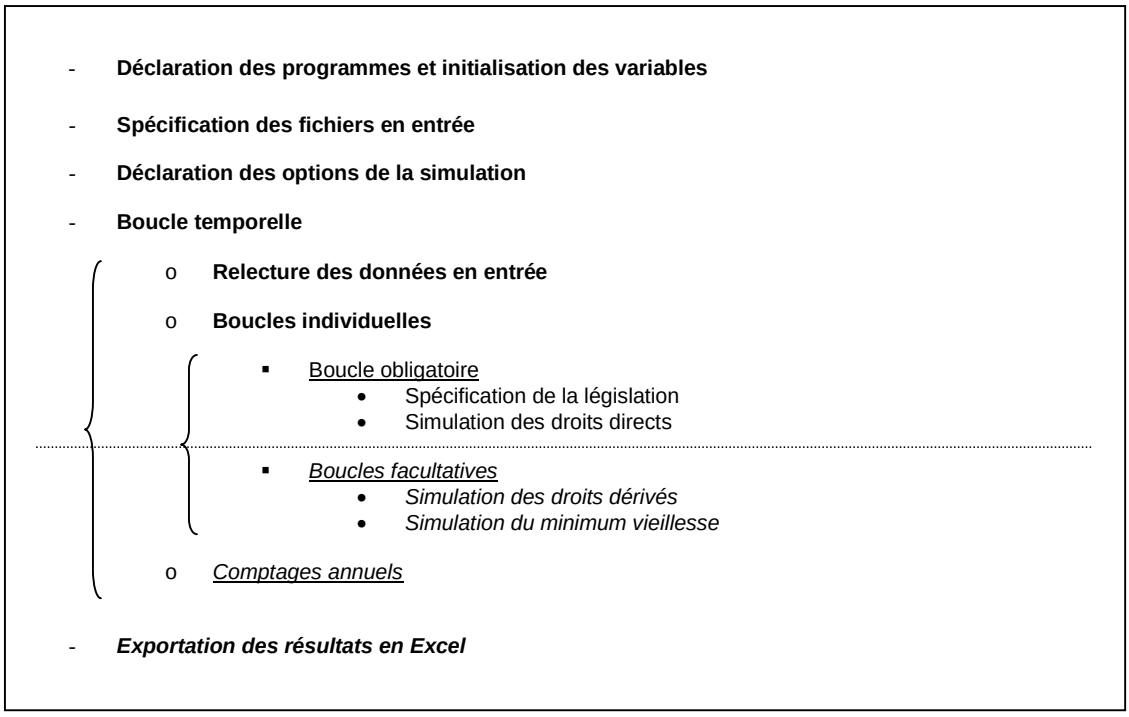
- La production de résultats agrégés par appel à des fonctions de tabulation

- La sauvegarde de ces résultats dans des fichiers Excel.

Cette forme de base peut elle-même connaître des variantes ou additions diverses.

- Il est possible d'imbriquer des boucles supplémentaires : par exemple insérer la boucle temporelle au sein d'une boucle balayant plusieurs scénarios successifs, ou enchainer plusieurs boucles individuelles au sein de la même boucle temporelle. Enchainer plusieurs boucles individuelles est notamment nécessaire pour le calcul de droits à retraite sous condition de ressource du ménage. Calculer l'accès d'un couple au minimum vieillesse suppose en effet la connaissance des retraites des deux conjoints. Il faut donc qu'une première boucle ait calculé les droits de l'ensemble de la population avant de pouvoir vérifier si tel ou tel couple d'individus a droit au minimum vieillesse.

Figure 6 : organisation d'un programme principal



- Il est possible de créer et renseigner d'autres variables micro que celles que le modèle gère par défaut. Par exemple, dans une simulation qui chaîne plusieurs scénarios, une variable `$taux_rep{$sc}[$i]` peut être créée pour stocker le taux de remplacement de l'individu `$i` dans le scénario `$sc`, après quoi il sera possible de calculer et analyser des écarts de taux de remplacement, d'un scénario à l'autre, individu par individu.

- Il est également possible de sauvegarder les résultats sous d'autres formats qu'en Excel.

Cette partie va présenter l'ensemble de ces opérations. La section IV.1 présentera le préambule type d'un programme de simulation. La section IV.2. portera sur le cœur de la simulation et sera donc surtout consacrée au contenu et à l'utilisation de la bibliothèque `OutilsRetr`. La section IV.3 expliquera comment générer et sauvegarder des variables de résultats à l'aide des programmes de tabulation et d'écriture des bibliothèques `OutilsTab` et `OutilsExcel`. La section IV.4 présentera enfin quelques exemples d'utilisation plus élaborés.

IV.1 Le préambule d'un programme de simulation

IV.1.1 Initialisations des programmes et des variables locales

Les premières lignes du programme prennent généralement la forme suivante :

```
# Ce programme effectue une projection des retraites sous l'hypothèse départ
# au taux plein avec la législation de 2003. Il calcule trois variables
# d'output :
# - le niveau de la retraite moyenne,
# - le ratio retraites/masse salariale
# - Le ratio retraités/actifs occupés
# Ces résultats sont sauvegardés dans le fichier Excel Exemple.xls

use strict;
```

Il est conseillé de commencer les programmes principaux par quelques lignes de commentaires. Les trois premières lignes du programme ci-dessus constituent des commentaires. Ils sont reconnaissables ici par la présence du signe # en début de ligne et ils apparaissent par défaut en vert dans l'environnement PerlIDE.

On a également vu qu'il est vivement conseillé d'utiliser la directive de compilation `use strict` située en début de code source, qui force l'utilisateur à prédéfinir toutes ses variables. Par défaut, aucune prédéclaration n'est requise dans Perl : on verra que le programme `Explore.pl` utilise cette faculté qui est particulièrement agréable en mode interactif, mais qui est en général déconseillée dans les programmes de grande taille.

La suite du préambule est constituée d'une séquence d'instructions du type :

```
use lib "..\/..\/Outils";
use DefVarRetr;
use OutilsRetr;
use OutilsTab;
use OutilsExcel;
my ($i,$t,@PENSION_MOYENNE,@RATIO_DEM,@RATIO_FIN,$resultats);
```

Dans cette séquence, l'instruction `use lib` sert à déclarer le répertoire dans lequel vont être trouvées ces différentes bibliothèques. On voit que le chemin d'accès est indiqué à l'aide de la syntaxe `..\/..\/` qui est celle à donner avec l'arborescence qui était proposée en section III.1. Chaque chaîne de caractère `..\/` signifie qu'il faut remonter d'un niveau dans l'arborescence pour trouver la librairie indiquée.

On fournit ensuite autant d'instructions `use` qu'il y aura de librairies directement mobilisées par le programme appelant. Si l'une des ces instructions `use` est oubliée dans le programme principal, des messages d'erreur apparaissent dans la fenêtre `Error Output`. Par exemple, si on oublie `DefVarRetr` qui initialise les variables, dès que le programme rencontre un appel d'une variable déclarée dans `DefvarRetr`, le programme ne la trouve pas et demande qu'elle soit déclarée de façon explicite, avec un message d'erreur du type :

```
Global symbol "@age" requires explicit package name at
../../Outils/OutilsDroits.pm line 561.
```

L'instruction `my` qui clôt cet en-tête sert à la fois à déclarer les variables additionnelles du type décrit en III.1.9, mais aussi les variables scalaires de boucle (indices `$i` et `$t`) et une variable `$resultats` qui servira à repérer le fichier Excel dans lequel seront sauvegardés les résultats, et dont la signification sera précisée plus loin en IV.4.

IV.1.2 Sélectionner les fichiers issus du générateur de biographies : la fonction UseBios

La fonction `UseBios` sert ensuite à définir les fichiers en entrée du bloc retraite comprenant les données issues du générateur de biographies et à donner quelques informations sur ces fichiers.

```
UseBios("Repertoire Bios",2003,1/9246);
```

Le premier argument est le nom du sous-répertoire du répertoire Bios sur lequel le programme doit aller lire les fichiers `BioEmp.txt` et `BioFam.txt`. Le deuxième argument indique à quelle année de départ se réfèrent ces fichiers. Dans la version actuelle du modèle, cette année de référence est l'année 2003, date de collecte de l'enquête patrimoine sur laquelle s'appuie le générateur de biographies. Le dernier paramètre est le taux de sondage de cet échantillon. Sur la base de ce taux de sondage, Destinie applique des coefficients de redressement à l'ensemble des programmes de tabulation assurant que les effectifs sortent systématiquement en milliers et les masses financières en milliards d'euros. Il est toujours possible de désactiver ces redressements, par exemple pour avoir des nombres d'individus effectivement présents dans l'échantillon, mais ceci oblige à faire appel à la fonction spécialisée `UseW` du module `OutilsTab` qui ne sera présentée qu'en partie VI.

On notera que ce n'est pas cette procédure `UseBios` qui procède à l'ouverture des fichiers. En effet, en cas de scénarios multiples, il faut procéder à autant de réouvertures initiales qu'il y a de scénarios différents. Ceci est donc plutôt fait lorsque la procédure `Relec` est appelée sur l'année initiale de projection, à l'intérieur de la ou des boucles temporelles. On renvoie à la section suivante pour la présentation de la procédure `Relec` et de son fonctionnement.

IV.1.3 Déclarer les options générales de la simulation : la fonction UseOpt

La fonction `UseOpt` sert ensuite à définir les options de la simulation.

```
UseOpt("TP");
```

Dans l'exemple ci-dessus, une seule option est présente : le type de comportement de départ à la retraite. `TP` indique que les personnes partent à la retraite lorsqu'elles atteignent le taux plein. Mais `UseOpt` sert aussi à préciser la liste de droits pris en compte dans la simulation et, le cas échéant, à lancer une transition vers un système en comptes notionnels.

L'ensemble des options disponibles sont les suivantes :

- Pour le comportement de liquidation (options exclusives les unes des autres)

`Exo`	: départ à un âge exogène
`Pmin`	: départ dès qu'un certain niveau de pension minimal est atteint

`TP` : départ au taux plein
`UInst` : départ si l'utilité instantanée à être en retraite dépasse l'utilité sans changement de statut
`UAct` : départ à l'âge qui maximise l'utilité actualisée sur le restant de l'existence (modèle de comportement de type Stock et Wise)
`SSW` : départ à l'âge qui maximise la masse des pensions actualisées sur l'ensemble de la durée de la retraite
`Accrual` : départ dès que le report d'un an de la liquidation fait baisser la masse des pensions actualisées sur l'ensemble de la durée de la retraite

- Pour le calcul des droits (options non exclusives)

`NoMC` : neutralisation du minimum contributif
`NoMDA` : neutralisation de la MDA
`NoBonif` : neutralisation de la bonif pour 3 enfants et plus
`NoAVPF` : neutralisation de l'AVPF
`CarTot` : prise en compte de l'ensemble de la carrière pour le calcul du SR fonction publique
`NoAssimil` : exclusion des années de préretraite et de chômage dans les durées validées

- Pour les comptes notionnels

`CN` : passage à une simulation en comptes notionnels

Pour cette dernière option, une deuxième instruction `UseOptCN` est utilisée pour préciser les caractéristiques de ce système en comptes notionnels. Cette fonction est décrite plus loin en VI.5.1.

L'ensemble des options demandées sont passées à `UseOpt` dans une chaîne de caractère unique dont la syntaxe est très libre : les options peuvent être indiquées dans un ordre quelconque, en majuscule ou minuscule, éventuellement au milieu d'autres caractères (par exemple au sein d'un nom de scénario en clair). Les appels suivants sont donc équivalents :

```
UseOpt ("TP NoMDA");
UseOpt ("tp nomda");
UseOpt ("Scenario 1 : TP/NoMda");
```

La fonction `UseOpt` a par ailleurs un second argument optionnel qu'il faut ajouter lorsqu'on a demandé les options `"Exo"` ou `"pmin"`. Dans ces deux cas, il faut définir le tableau qui fournit l'âge cible de départ pour chaque individu ou le niveau minimum de pension à partir duquel il décide de liquider. En général, ces informations sont des informations sauvegardées d'une simulation précédente du même programme puisque l'idée est soit de simuler une variante à âge de départ inchangé, soit une variante à niveau de retraite inchangé. Si on suppose par exemple que ce sont des tableaux `@ageref` ou `@pens_cible` qui ont servi à garder ces informations, on aura des appels de la forme :

```
UseOpt ("Exo",@ageref);
```

ou :

```
UseOpt ("pmin",@pens_cible);
```

Ces cas seront examinés plus en détail dans le paragraphe III.4 qui analyse des cas de programmation plus complexes. Mais le tableau `@ageref` pourrait aussi être n'importe quel autre tableau construit directement par l'utilisateur, par exemple un tableau fixant l'âge de 60 ans pour tout le monde.

IV.2 Les boucles : version élémentaire

IV.2.1 Vue d'ensemble

La double boucle emboîtée sur la date et la population simulée constitue le cœur de la simulation. L'encadré ci-dessous rappelle comment se présentait cette double boucle dans l'exemple de la figure 2.

```
# Boucle temporelle
for $t (103..150)
{
   # Simulation
   Pointage ("Simulation année ".$t);
   for $i (1..IMax)
   {
      UseLegRetroMax(2003);
      UseLeg($t,$anaiss[$i]);
      SimDir($i);
   }
}; # Fin boucle temporelle
```

La boucle temporelle est la boucle `for $t (103..150)`. Elle va de 2003 à 2050, mais l'indiçage tient compte de ce que, pour économiser la place, les tableaux de paramètres prennent l'année 1900 comme année d'origine. Ce principe est adopté dans l'ensemble du modèle, même si on verra plus loin que beaucoup de procédures acceptent qu'on leur passe des millésimes complets.

La fonction `pointage` ne joue pas de rôle dans la simulation. Elle sert seulement à suivre son avancement. Dans l'exemple ci-dessus, elle affichera un message à chaque année de simulation, en précisant l'heure à laquelle le programme a démarré sa simulation.

À l'intérieur de la boucle temporelle, la première opération consiste à récupérer les données nécessaires dans les fichiers issues du générateur de biographie. Ceci se fait par appel de la fonction `Relec` de la bibliothèque `OutilsRetr`, dont le seul argument est la date `$t`. Cette relecture est donc faite en dehors de la boucle individuelle. C'est la procédure `Relec` elle-même qui gère le balayage de l'ensemble de la population.

Une fois effectuée cette relecture, la fonction principale du programme est de simuler les départs en retraite. C'est cette opération qui est effectuée à l'intérieur d'une boucle individuelle `for $i (1..IMax)` où `IMax` est en fait une fonction de la bibliothèque `OutilsBase` qui indique le nombre total d'individus connus de Destinie. Cette simulation des liquidations fait intervenir trois fonctions :

- la fonction `UseLegRetroMax` qui indique le choix de la législation pour les reconstitutions initiales,
- la fonction `UseLeg` qui précise le choix de la législation en projection,
- la fonction `SimDir` qui lance la simulation des pensions de droits directs.

La dernière accolade ferme la boucle temporelle.

On va maintenant examiner plus en détail chacune des procédures appelées dans cette double boucle.

IV.2.2 Relire les biographies individuelles : la fonction Relec

La fonction `Relec` sert à relire les contenus des fichiers `BioEmp` et `BioFam`. Elle est lancée dans le programme principal dans la boucle temporelle, au début de chaque année de la simulation. Son seul argument est la date, sous forme millésimée ou modulo 1900. On peut donc avoir des appels de la forme:

```
Relec($t);
Relec(103);
Relec(2003);
```

Pour préciser à quoi a accès le programme après appel de cette procédure, il est nécessaire de donner quelques indications sur le contenu des deux fichiers auxquels elle accède.

Ce fichier `BioEmp` comprend trois lignes par individu. La première ligne comprend des informations qui sont indépendantes de l'âge :

- l'identifiant de l'individu
- le sexe
- l'année de naissance (en 4 chiffres)
- l'âge de fin d'études
- le taux de prime (en l'état, Destinie applique le même taux de prime sur l'ensemble de la carrière)

Les deux lignes suivantes donnent respectivement les séquences de statuts d'emploi et de salaires nominaux successifs entre l'âge de 15 ans et le décès. Le séparateur utilisé est le point-virgule.

Ce fichier `BioEmp` est lu en une seule fois dès la première année de simulation. Cela signifie que, dès le début de la simulation, on connait l'ensemble des carrières non seulement rétrospectives mais aussi prospectives de l'ensemble des individus, y compris d'ailleurs pour des individus qui ne sont pas encore actifs car devant naître (ou migrer) ultérieurement : autrement dit, à chaque date, il y a une vision d'ensemble des variables `$statut_[$i][$a]` et `$salaire_[$i][$a]` pour l'ensemble des individus simulés qu'ils soient déjà présents ou non dans l'échantillon à la date initiale.

En utilisation normale, ces carrières ne sont modifiées par le programme appelant qu'au moment du départ en retraite : lorsqu'on simule un départ en retraite, tous les statuts des années ultérieures sont mis d'office à la valeur `$CodeIna` et les salaires sont mis à zéro. Ceci sera fait automatiquement par la fonction `SimDir`. Mais il est aussi possible d'introduire d'autres modifications sur ces profils de carrière à d'autres stades du programme. Un exemple en sera donné plus loin à la section IV.3.4.

La structure du fichier `BioFam` est un peu plus complexe. Pour chaque année de simulation, il contient une ligne pour chaque individu connaissant un changement de situation familiale. Les informations, toujours séparées par un point-virgule sont :

- l'identifiant de l'individu
- celui de son père
- celui de sa mère
- sa situation matrimoniale
- l'identifiant du conjoint
- les identifiants des enfants (au maximum 6)

Pour la première année, aucune de ces informations n'est connue *ex ante*, il y a donc une ligne par individu de la base. Pour les années suivantes, seuls les individus qui connaissent un changement sont renseignés. Le nombre d'enregistrements est donc variable d'une année de projection à l'autre. Pour marquer la séparation entre les années, le fichier comprend donc des lignes additionnelles marquant le passage à l'année suivante. À la jonction entre deux années, on a donc un contenu du type :

```
7930;517;6299;1;;;;;;;
7931;736;6314;1;;;;;;;
7932;528;7160;1;;;;;;;
*** Fin année 2018
31;28;29;4;;;;;;;
32;28;29;2;3189;;;;;;
78;76;77;2;5266;;;;;;
96;-1338;1339;3;-97;98;99;100;;;
```

Dans cet exemple, le dernier individu connaissant un changement de situation en 2018 est l'individu 7932. Il s'agit en fait d'une naissance. Cet individu a pour parents les individus 528 et 7160, son statut matrimonial de départ est le statut de célibataire, avec une valeur qui est donc manquante pour l'identifiant du conjoint et une ancienneté de zéro année dans ce statut et il n'a aucun enfant.

Pour l'année 2019, les deux individus 31 et 32 qui ont les mêmes parents 28 et 29 connaissent respectivement une séparation et une mise en couple, dans le deuxième cas avec la personne d'identifiant 3189. La quatrième personne connaissant un événement démographique pour cette année 2019 est l'individu 96. Il n'a plus son conjoint (statut matrimonial à 3 et identifiant du conjoint négatif -97), et perd son père (identifiant -1338) mais a encore sa mère (1339). Conserver les identifiants des personnes liées à un individu donné après leur décès permet de conserver l'information sur les liens familiaux qui passaient par cette personne. Pour ce qui concerne l'individu décédé, il cesse d'être considéré comme présent dans la population : c'est le programme `Relec` qui se charge de mettre à zéro l'indicatrice `$present[$i]`. Ceci évite notamment que l'individu soit pris en compte dans les comptages relatifs à l'année courante.

En année courante, `Relec` se charge aussi de mettre à jour l'âge, les variables de statut et de salaire courant `$statut[$i] et $salaire[$i]` et les indicatrices permettant de savoir si l'individu est décédé au cours de l'année ou s'il est un migrant de l'année (`$deces[$i]` et `$migrant[$i]`). Elle remet par ailleurs à zéro l'indicatrice `$liq[$i]` de liquidation l'année courante : celle-ci sera recalculée juste après, pour les nouveaux liquidants, par la fonction `SimDir`.

IV.2.3 Choisir la législation : UseLeg et UseLegRetroMax

Comme déjà indiqué en III.3.6, la fonction `UseLeg` sert à initialiser les valeurs des paramètres législatifs que `DefVarRetr` ne lit pas directement sous forme de séries temporelles dans `ParamSociaux.xls`. Comme les paramètres de calcul de la retraite dépendront en général de l'individu, à année donnée, elle doit être appelée à l'intérieur de la boucle individuelle de calcul des pensions

La fonction `UseLeg` fixe les valeurs de ces différents paramètres en fonction de l'année et de la génération. Par exemple :

```
UseLeg($t,$anaiss[$i]);
```

signifie que, jusqu'au prochain appel de `UseLeg`, on va utiliser les paramètres que prévoyait la législation de l'année `$t` pour les individus nés en `$anaiss[$i]`.

Cette fonction permet de simuler d'autres règles de calcul des droits que celles s'appliquant ou devant effectivement s'appliquer à l'individu `$i`.

```
UseLeg(1992,$anaiss[$i]);
UseLeg(1992,1942);
```

mettent en œuvre le barème que prévoyait la législation de 1992 pour les individus nés en `$anaiss[$i]` ou pour la génération 1942. Si ceci ne suffit pas à décrire le scénario souhaité, il est toujours possible de modifier à la main un ou plusieurs paramètres après ces appels de `UseLeg`, comme on a vu pouvoir le faire pour les paramètres lus dans `ParamSociaux.xls`.

On notera que, dans ces appels, on utilise le millésime complet. En fait, pour pouvoir utiliser les mêmes compteurs de date que ceux qui indicent les séries temporelles le programme accepte aussi les dates ramenées à l'origine 1900, i.e. on aurait pu écrire :

```
UseLeg(103,$anaiss[$i]);
```

C'est ce qui était implicitement fait dans le premier exemple d'appel si `$t` était une variable de boucle comprise entre 103 et 150.

La fonction `UseLegRetroMax` est le pendant de `UseLeg` pour les reconstitutions rétrospectives de droits directs. La raison d'être de cette fonction est que le nouveau modèle Destinie 2 ne se contente pas de projeter les retraites. Il les réimpute également pour les personnes déjà retraitées dans l'échantillon initial. La raison principale de ce choix a été d'assurer une meilleure continuité des trajectoires en début de projection. En effet, chaîner des pensions calculées et des pensions initiales observées crée fatalement une discontinuité initiale, sauf à avoir un module parfait et exhaustif de calcul des droits et des droits courants également parfaitement mesurés. Aucune de ces deux conditions ne peut être parfaitement remplie et Destinie 2 résout le problème en imputant les pensions des retraités actuels à l'aide des mêmes modules de calcul que ceux utilisés en projection. Ceci assure une projection à « biais constant », qu'on a considéré préférable à une projection à biais variable.

Il s'est avéré que cette façon de procéder avait un deuxième avantage : dès lors que les pensions initiales sont réimputées, il est possible de le faire sur d'autres bases que la législation effective pour construire des scénarios « contrefactuels ». Un scénario contrefactuel consiste par exemple à calculer ce que seraient aujourd'hui les retraites et comment elles seraient destinées à évoluer si on en était resté à la législation de 1992.

Pour que ce type de calcul soit possible, il faut évidemment que l'utilisateur puisse choisir la législation à appliquer à ces imputations rétrospectives. C'est ce à quoi sert la fonction `UseLegRetroMax`. Elle a un argument unique qui fixe la date à laquelle on choisit de bloquer rétrospectivement la législation. Ainsi :

```
UseLegRetroMax(1992);
```

simulera les conditions initiales qui auraient résulté d'un maintien de la législation de 1992.

Une imputation des pensions initiales selon les règles effectives sera obtenue en demandant :

```
UseLegRetroMax(2003);
```

Comme pour `UseLeg`, ces appels ne fixent que les paramètres de calcul non lus dans `ParamSociaux.xls`. Les paramètres que `DefVarRetr` lit dans `ParamSociaux.xls` peuvent faire l'objet de modifications rétrospectives directes, dans une boucle temporelle spécifique en amont de la boucle temporelle principale.

Par exemple (cas d'école) :

```
for $t (70..150)
{
   $plafondSS[$t]=$plafondSS[70] ;
}
```

conduit à simuler l'évolution rétrospective et prospective du système de retraite dans l'hypothèse où le plafond aurait cessé d'évoluer depuis 1970.

IV.2.4 Simuler la liquidation : la fonction SimDir

Cette fonction est la fonction principale de Destinie 2. Elle initialise ou actualise les droits directs. Ce programme est appelé à l'intérieur de la boucle sur données individuelles. L'argument unique est l'identifiant de l'individu.

Les autres options de comportement ont été définies par `UseOpt`, s'il y a lieu. La législation mise en œuvre est celle définie par le dernier appel de `UseLeg` ou de `UseLegRetroMax`.

Cette fonction fait appel à d'autres fonctions créées dans d'autres sous-programmes que `OutilsRetr` (`TestLiq`, `Liq` et `Revalo`). Ces fonctions ne seront pas détaillées dans cette partie. Elles n'ont besoin d'être connues que pour une utilisation avancée du modèle et leur articulation exacte sera précisée plus loin en section VI.5.2.

À ce stade, il est surtout important de comprendre comment la fonction `SimDir` procède.

La fonction `SimDir` ne concerne que les individus en vie de plus de 54 ans. Si c'est le cas, la fonction `SimDir` distingue deux situations :

Si on est à l'année initiale de la projection (en général 2003), il s'agit dans ce cas de réimputer l'ensemble des pensions initiales, pour tout le stock d'individus en âge d'être retraité. La fonction `SimDir` calcule l'année de liquidation et le montant des pensions éventuelles. Elle remonte à l'année où l'individu a eu 55 ans et lance la fonction `TestLiq` qui teste si les conditions pour la liquidation sont remplies. Si c'est le cas, elle lance la fonction `Liq` qui calcule le montant des pensions. Si ce n'est pas le cas, elle reteste le fait qu'il y ait eu liquidation ou non pour tous les âges jusqu'au minimum de l'âge courant et de l'âge de 65 ans. Une fois l'année de liquidation déterminée les pensions des années suivantes (jusqu'à l'année de départ de la simulation) sont revalorisées à l'aide de la fonction `Revalo`.

- S'il s'agit d'une année postérieure à l'année de départ, il s'agit soit de revaloriser les pensions en cours, soit de simuler les nouvelles liquidations. Si l'individu n'est pas retraité, `SimDir` teste s'il s'agit de l'année de liquidation (`TestLiq`) et si c'est le cas affecte le montant des pensions (`Liq`). Si l'individu est déjà retraité `SimDir` revalorise les pensions avec `Revalo`.

Exemples d'appel combiné avec `UseLeg`, dans une double boucle sur `$t` et `$i` :

```
UseLegRetroMax(2003);
UseLeg ($t,$anaiss[$i]);
SimDir($i);
```

initialise ou projette les droits en `$t` avec les conditions réelles

```
UseLegRetroMax(1992);
UseLeg (1992,$anaiss[$i]);
SimDir($i);
```

initialise ou projette les droits en `$t` avec une législation bloquée à l'année 1992.

IV.3 Les boucles : quelques cas plus complexes

IV.3.1 Lancer plusieurs scénarios dans un seul programme

Très souvent, un programme ne sert pas à simuler un scénario unique mais plusieurs scénarios. Les deux boucles temporelle et individuelle sont donc emboitées dans une boucle balayant l'ensemble des scénarios qu'on veut examiner. Soit `$sc` la variable servant à identifier les scénarios. Il peut s'agit d'un simple numéro, mais il sera en général plus parlant d'utiliser une chaine de caractère puisqu'il est possible d'effectuer des boucles indicées par des chaines de caractères et on verra plus loin qu'on pourra ensuite en sauvegarder les résultats dans des

hachages eux-aussi indicés par les mêmes chaînes. On peut également choisir ces chaînes de sorte qu'elles puissent être directement passées comme arguments de `UseOpt`.

La démarche consiste donc à écrire, en début de programme (avant les boucles temporelles et avant les ouvertures de fichiers en sortie s'il y en a) une boucle du type :

```
for $sc ("TP","Uact")
{
   UseOpt($sc) ;
   for $t (103..150)
   {
      Relec($t) ;
      for $i (1..Imax)
      {
         …
      }
   }
}
```

La chaîne `$sc` peut se présenter de manière encore plus complexe. Par exemple, on peut labeller des scénarios sous la forme `$sc="TP/NoBonif-1992"` ou `"Uact/NoBonif-1992"` puis utiliser l'opérateur Perl `split` pour éclater `$sc` en deux variables `$option` et `$legislation` qui pourront ensuite être passées séparément à `UseOpt` et `UseLeg`, cette instruction `split` s'utilisant sous la forme :

```
($option,$legislation) = split /-/,$sc;
```

après quoi l'utilisateur pourra appeler successivement les deux fonctions `UseOpt($option)` et `UseLeg($legislation,$anaiss[$i])`.

IV.3.2 Sauvegarder une variable micro pour d'autres scénarios

Lorsqu'on simule plusieurs scénarios, on peut vouloir conserver un résultat micro d'un des scénarios pour utilisation dans un autre scénario. Il y a deux cas principaux : le premier est-celui où on veut procéder à des comparaisons des situations individuelles entre scénarios, le second est-celui où on a besoin des résultats d'un scénario de référence pour paramétrer les hypothèses d'un scénario alternatif. On a vu que ce deuxième cas de figure se présente lorsqu'on utilise les options `exo` et `pmin` pour la simulation de l'âge de liquidation.

L'exemple qui suit présente la façon de procéder dans les deux cas. On suppose deux scénarios `"ref"` et `"var"` correspondant à deux législations. On suppose qu'on veut simuler le second avec exactement le même âge de liquidation que le premier et qu'on veut calculer les variations individuelles de niveau de pension à la liquidation entre les deux scénarios. Si on a prédéclaré les variables `@age_ref`, `@p_ref` et `@delta_p`, ceci pourra être assuré par la séquence suivante :

```
my ($sc,$i,$t,@age_ref,@p_ref,@delta_p);
…
for $sc ("ref","var")
{
   if ($sc eq "ref")
   {
      UseOpt("TP")
   }
   else
   {
      UseOpt ("Exo",@age_ref)
   }
   for $t (103..150)
```

```
      {
         Relec($t);
         for $i (1..Imax)
         {
              SimDir($i)
         }
         if ($liq[$i]==1)
         {
            if ($sc eq "ref")
            {
               $p_ref[$i]=$pension[$i];
            }
            else
            {
               $delta_p[$i]=$pension[$i]-$p_ref[$i];
            }
         }
      }
      if ($sc eq "ref")
      {
         @age_ref=@ageliq ;
      }
   }
```

Dans cet exemple, les variables micro `@p_ref` et `@delta_p` ne sont calculées que pour les personnes liquidant en cours de projection (sous condition sur la variable `@liq` générée par `SimDir`). En revanche, on note que la variable `@age_ref` est renseignée en fin de boucle, de manière globale, par simple copie de `@ageliq`. Ceci permet de récupérer l'âge de liquidation du scénario de référence non seulement pour les personnes liquidant en cours de projection mais aussi pour les gens dont on simule la liquidation rétrospective avant l'année 2003. On aurait aussi pu le faire à l'intérieur de la boucle sur `$i` et `$t`, mais sans tester sur la valeur de `$liq`.

IV.3.3 Enchaîner plusieurs boucles individuelles : le cas des droits dérivés et du minimum vieillesse

Il est parfois nécessaire de faire plusieurs boucles sur les individus à l'intérieur d'une boucle temporelle. En effet, certaines fonctions sont dépendantes des calculs effectués sur l'ensemble des individus. En particulier, pour calculer les droits à la retraite dérivés (les pensions de réversion) il faut que la pension du conjoint de l'individu soit déjà connue. La fonction `SimDer` ne peut être mise en œuvre que si la fonction `SimDir` a déjà été appliquée à tous les individus. Le principe est identique pour le calcul du minimum vieillesse dont la perception dépend des revenus de l'individu et de ceux son conjoint. Dans la première boucle on calcule les revenus de tous et dans une deuxième on calcule le minimum vieillesse. On peut donc rajouter alors deux nouvelles boucles, après la boucle principale sur les individus :

```
for $i (1..IMax) {SimDer($i)};
for $i (1..IMax) {SimMinVieil($i)};
```

Comme cela a été signalé plus haut, ce mode de programmation permet de faire des simulations plus courtes dans le cas où l'on n'a pas besoin de tous les calculs de droits. Logiquement, le programme simple sans les fonctions `SimDer` et `SimMinVie` prend moins de temps que si on y rajoute les deux fonctions. Cet avantage n'est pas négligeable surtout dès lors que l'on travaille sur les bases complètes (gros échantillons).

On va préciser le format de ces deux fonctions.

La fonction SimDer

La fonction `SimDer` simule les pensions de reversions initiales ou projetées.

Pour l'initialisation, la réversion est imputée sur la base de la retraite d'un pseudo-conjoint désigné par le générateur de biographie, après quoi le lien avec ce pseudo-conjoint est effacé. Si le conjoint n'était pas encore retraité au moment de son décès, on calcule la pension qu'il aurait perçu à 60 ans pour évaluer la pension de réversion.

Ce n'est pas cette procédure qui effectue les revalorisations. Elles sont faites en même temps que pour les droits directs dans `SimDir`.

La fonction `Simder` a pour seul argument `$i`.

La fonction SimMinVieil

La fonction `SimMinViel` simule le minimum vieillesse pour les individus qui sont personnes de référence de leur ménage. Elle vérifie les conditions de ressources du ménage grâce à différentes fonctions de la bibliothèque `OutilsMen`.

Cette fonction a un seul argument : `$i`.

IV.3.4 Modifier des données individuelles en amont de la simulation des retraites

Une manipulation moins fréquente mais qui peut être utile consiste à introduire des modifications dans les données relues dans le fichier `BioEmp` ou le fichier `BioFam`, autres que celles correspondant à la liquidation des retraites. Par exemple, la séquence suivante simule un choc de chômage pour l'année 2010 augmentant de deux points le risque de chômage des salariés du privé.

```
for $t (103..150)
{
   Relec($t);
   for $i (1..Imax)
   {
      if (($t==110) && (In($statut[$i],$CodeNC,$CodeCad))
                    && (Booleen(0.02)))
      {
          $statut_[$i][$age[$i]=$CodeCho;
          $statut[$i]          =$CodeCho;
      }
      SimDir($i);
   }
}
```

Cet exemple montre que l'organisation à deux étages séparant la simulation des retraites et le générateur de biographies n'oblige pas nécessairement à remonter à ce dernier pour des variantes d'activité, tant que celles-ci restent simples.

On notera que, dans cet exemple, il faut penser à modifier de façon cohérente les variables `$statut` et `$statut_`. On relèvera aussi l'utilisation des deux fonctions `In` et `Booleen` qui sont de fonctions de la bibliothèque généraliste OutilsBase qui sera présentée plus loin en VI.4

Si on utilisait le même principe pour simuler des chocs sur la situation familiale, il faudrait tenir compte de la façon dont `Relec` lit le fichier `BioFam`. Une modification une année donnée dans une des variables de `BioFam` sera préservée tant que `Relec` ne lira aucun changement dans `BioFam` pour l'individu `$i`, et sera annulée lorsque `Relec` rencontrera une ligne dans `BioFam`

relative à cet individu `$i`. Il faut donc prévoir une façon de gérer ce problème, mais les variantes avec chocs sur la situation démographique et familiale sont *a priori* moins fréquentes. Les variantes démographiques, s'il y en a, auront plutôt intérêt à s'appuyer sur différents couples de fichiers `BioEmp` et `BioFam` résultant d'exécutions séparées du générateur de biographies.

IV.3.5 Simuler des cas-types

L'approche par cas-types est un complément utile de la microsimulation. La nouvelle conception de Destinie permet d'utiliser sa boite à outils sur d'autres carrières que celles générées par le générateur de biographies et ces carrières peuvent être des carrières de cas types totalement contrôlés par l'utilisateur. Dans ce cas, il n'y a pas d'appel ni à `UseBios` ni à `Relec`. On remplit les tableaux prédéfinis par `DefVarRetr` avec les caractéristiques des cas-types qu'on souhaite étudier. L'exemple ci-dessous montre par exemple comment construire trois cas types de non cadres à carrières complètes et ayant fait ces carrières à 0,5, une fois ou 1,5 fois le plafond de la sécurité sociale. On calcule leurs droits pour un départ à 60 ans en 2010.

```
my ($i,$a);

for $i (1..3)
{
   $anaiss[$i]=1950;
   $sexe[$i]=$CodeHom;
   for $a (20..60 )
   {
      $statut_[$i][$a] =$CodeNC;
      $salaire_[$i][$a]=0.5*$i*$PlafondSS[$anaiss[$i]+$a-1900];
   }
   $age[$i]=60;
   $statut[$i] =$statut_[$i][$age[$i]];
   $salaire[$i]=$salaire_[$i][$age[$i]];
   UseLeg(2010,$anaiss[$i]);
   SimDir($i);
}
```

À partir de là, on peut afficher ce que l'on veut des caractéristiques de ces individus et on voit facilement comment le programme peut être complexifié pour analyser des cas-types plus nombreux et plus élaborés.

IV.4 Création de résultats agrégés

IV.4.1 Fonctions de tabulation génériques

Les résultats macro sont le plus souvent des résultats annuels. Ils ne dépendent donc pas de `$i`, mais de `$t`. Ils doivent donc être calculés dans la boucle temporelle, mais en dehors des boucles individuelles.

Dans l'exemple de la figure 2, ceci correspondait au bloc d'instructions :

```
$PENSION_MOYENNE[$t] = MVar(@pension, sub{$pension[$_]>0   });
$RATIO_FINANCIER[$t] = SVar(@pension)/Svar(@salaire);
$RATIO_DEMOGR[$t]    = Ratio(sub{$pension[$_]>0},sub{$salaire[$_]>0});
```

situé juste avant la fin de la boucle temporelle.

Un tel segment de programme s'appuie sur des fonctions de tabulation instantanées proposées par la bibliothèque `OutilsTab`. En règle générale, ces fonctions ont pour argument la variable qu'il s'agit de tabuler, suivi d'un filtre passé sous forme d'une instruction Perl appliquée à l'individu d'indice générique `$_` et passée entre accolades après le mot clé **sub**. Cette syntaxe est détaillée complètement en VII.2.6. Pour les utilisations courantes il suffit de comprendre que le texte passé après le mot-clé **sub** est un mini-sous-programme que la fonction va évaluer dans une instruction `if` à l'intérieur de sa propre boucle de balayage de la population. Lorsqu'il s'agit de simples comptages, on ne passe que le filtre, ou deux filtres lorsqu'il s'agit de calculer des taux ou des ratios.

On va se limiter ici aux cas les plus usuels, une présentation complète de la bibliothèque étant faite en partie IV

Fonction SVar

Cette fonction somme des valeurs d'une variable sous condition sur les valeurs d'une ou plusieurs autres variables.

Exemples d'appel :

Masse salariale des individus du groupe 15-65 ans :

```
$MASSE_SAL = SVar(@salaire,sub{($age[$_]>=15) && ($age[$_]<=65)});
```

ou

```
$MASSE_SAL = SVar(@salaire,sub {In($age[$_],(15..65))});
```

Masse salariale de l'ensemble de la population :

```
$MASSE_SAL = SVar (@salaire);
```

Effectif de la population âgée de 15 à 64 ans

```
$POP15_64 = SVar (@pop, sub{In($_,(15..64))});
```

Fonction Mvar

Cette procédure fonctionne selon le même principe que SVar mais calcule une moyenne. Pour alléger les affichages, le résultat est fourni avec une seule décimale.

Exemples d'appel

Salaire moyen du groupe 15-65 ans. Comme précédemment, les deux formes d'écriture suivantes sont équivalentes :

```
$SAL_MOY = MVar(@salaire,sub{($salaire[$_]>0)
                          && (($age[$_]>=15) && ($age[$_]<=65))});
$SAL_MOY = MVar(@salaire,sub{($salaire[$_]>0)
                                 && (In($age[$_],(15..65)))});
```

On notera que, dans cet exemple, on filtre sur les salaires positifs pour éviter d'inclure dans la moyenne les individus à salaire nul. Le même problème ne se posait pas pour la mise en œuvre de la fonction `SVar`.

Fonction EVar

Cette fonction suit le même principe que `SVar` et `MVar` mais calcule un écart-type.

Fonction Count

Cette fonction compte le nombre d'individus remplissant la condition passée en argument. Sans argument, elle retourne le nombre total d'individus présents à la date courante.

Exemples d'appel

```
@POP    = Count;
@SALPOS = Count (sub {($age[$_]>15) && ($salaire[$_]>0)});
```

Fonction Taux

Cette fonction calcule un taux, i.e. la proportion d'individus remplissant la première des conditions données en argument parmi les individus remplissant la seconde. S'il n'y a pas de deuxième condition, le taux est calculé par rapport à la population totale. Il est retourné directement en %, avec une seule décimale.

Exemples d'appel

```
@TAUX_ACT{"20-40"} = Taux(sub {In($statut[$_],@CodesAct)},
                          sub {In($age[$_],(20..40))});
```

calculera la part des actifs (premier **sub** : `In($statut[$_],@CodesAct)` parmi la population des individus dont l'âge est compris entre 20 et 40 ans (deuxième **sub** : `In($age[$_], (20..40))`).

```
@TAUX_ACT_GLOB = Taux(sub {In($statut[$_],@CodesAct)});
```

calculera la part des actifs sur l'ensemble de la population.

Fonction Ratio

Cette fonction calcule un ratio, i.e. le rapport entre les effectifs de deux populations. Le principe d'appel est le même que pour la fonction `Taux`, au fait près que la condition définissant la population du dénominateur n'est plus facultative et que les deux conditions sont utilisées de manière disjointe (dans la fonction `Taux`, le numérateur est la sous-population remplissant simultanément les deux conditions qui étaient passées en argument).

Exemple d'appel :

```
@Ratio_dep = Ratio(sub {$age[$_]>59},
                   sub {($age[$_]>=20) && ($age[$_]<60)});
```

calculera le rapport entre le nombre d'individus d'âge supérieur à 59 ans et le nombre d'individus entre 20 et 59 ans.

IV.4.2 Un exemple

L'extrait ci-dessous présente d'autres exemples de mobilisation de ces fonctions pour des calculs d'agrégats.

```
# Comptages année courante
$MSAL[$t]   = SVar(@salaire);
if ($t==103) {$PIB[$t]=1800000}
else  {$PIB[$t]=$PIB[103]*$MSAL[$t]/$MSAL[103]};
$TOTDIR[$t] = SVar  (@pension   )/$PIB[$t];
$TOTDER[$t] = SVar  (@rev       )/$PIB[$t];
$MOYDIR[$t] = MVar  (@pension , sub{$pension[$_]>0  })/$Prix[$t];
```

```
   $MOYDER[$t] = MVar  (@rev     , sub{$rev[$_]>0      })/$Prix[$t];
   $NBDIR[$t]  = Count (sub{$pension[$_]>0   });
   $NBDER[$t]  = Count (sub{$rev[$_]>0       });
```

Cet exemple sert à calculer des sommes de pensions par rapport au PIB, des pensions moyennes et des nombres de retraités, en distinguant les pensions de droit direct et les pensions de réversion.

On calcule d'abord une masse salariale `$MSAL[$t]` par `SVar`. C'est sur la base de cette masse salariale qu'on calcule une évaluation de PIB à chaque date, sous l'hypothèse d'une constance du rapport salaires/PIB à compter de 2003. Ceci suppose bien sûr que l'agrégat `@PIB` ait été déclaré dans le programme.

On notera que aussi bien `@MSAL` que `@PIB` sont ici des grandeurs nominales, ce qui ne pose pas de problème pour le calcul des variables `$TOTDIR` et `$TOTDER` qui sont des ratios. En revanche, les pensions moyennes `$MOYDIR` et `$MOYDER` calculées par `MVar` sont divisées par l'indice des prix `$Prix[$t]` pour apparaître en valeurs réelles. Cette correction joue surtout sur les premières années : le démarrage de la projection se fait en effet avec les valeurs effectives des paramètres de gestion des retraites enregistrées jusqu'à la date courante, par exemple les valeurs du point ou les taux de revalorisation effectivement appliqués jusqu'en 2009. Au-delà, Destinie fige la variable `$Prix[$t]` et cesse de distinguer évolutions réelles et nominales. Mais la correction par les prix est nécessaire au moins au titre des premières années.

La fonction `Count` sert enfin à calculer les effectifs de bénéficiaires de droits directs et de droits dérivés.

Si on voulait différencier ces variables par sexe, on peut avoir la séquence suivante, moyennant déclaration préalable des deux hachages de tableaux `%MOYDIRS` et `%MOYDERF` :

```
for $s ($CodeHom,$CodeFem)
{
   $buf=$Label{"sexe"}[$s];
   $MOYDIRS{$buf}[$t]=MVar(@pension,
                           sub{($pension[$_]>0)&& ($sexe[$_] eq $s)});
   $MOYDERS{$buf}[$t]=MVar(@pension,
                           sub{($rev[$_]>0)    && ($sexe[$_] eq $s)});
}
```

Ces deux hachages sont ensuite sauvegardables directement dans des feuilles Excel par une procédure `SavHS` qui sera vue dès la section suivante.

De la même façon, si on était à l'intérieur d'une boucle sur plusieurs scénarios indicés par la variable `$sc`, on pourrait avoir des tabulations sauvegardant dans des hachages de séries temporelles indicés par le nom du scénario.

IV.4.3 Fonctions de tabulation directe des salaires et des retraites

Les tabulations de masses ou de moyennes de salaires et de retraites sont celles qui reviennent le plus souvent. Or la syntaxe de ces tabulations peut être assez lourde lorsqu'on veut obtenir ces valeurs pour des sous-populations spécifiques et/ou, dans le cas des pensions, pour des composantes particulières de la pension. Pour simplifier ces calculs, Destinie 2 propose quatre fonctions de raccourcis : les fonctions `SSal`, `Msal`, `SPens` et `MPens`.

Procédure SSal

Cette fonction retourne une masse salariale. Le premier argument optionnel est une chaîne de caractères incluant le champ (`"pri"`, `"in"`, `"fp"`) et/ou du sexe (`"hom"` ou `"fem"`). Le

second argument, également optionnel, est une condition additionnelle passée sous la même forme que pour les autres procédures de tabulation.

Exemples

```
$MSAL[$t]=SSal;
$MSAL[$t]=SSal("pri/fem");
$MSAL[$t]=SSal(sub{$age[$_]>40});
$MSAL[$t]=SSal("fp/hom",sub{$age[$_]>40});
```

Procédure MSal

Cette fonction retourne un salaire moyen avec les mêmes règles d'appel que pour `SSal`. La tabulation ne prendra en compte que les individus du champ pour qui le salaire est positif.

Exemples

```
$MSAL[$t]=MSal;
$MSAL[$t]=MSal("pri/fem");
$MSAL[$t]=MSal(sub{$age[$_]>40});
$MSAL[$t]=MSal("fp/hom",sub{$age[$_]>40});
```

Procédure SPens

Cette fonction retourne une masse de pensions. Le premier argument optionnel est une chaine de caractère incluant la définition du régime (`"rg"`, `"ar"`, `"ag"`, `"in"` ou `"fp"`) et/ou du sexe (`"hom"` ou `"fem"`). Par défaut, la pension calculée est la pension totale pour les deux sexes. Le second argument, également optionnel, est une condition additionnelle passée sous la même forme que pour les autres procédures de tabulation.

Exemples d'appel :

```
$PTOT[$t]=SPens;
$PTOT[$t]=SPens("rg/fem");
$PTOT[$t]=SPens(sub{$liq[$_]==1});
$PTOT[$t]=SPens("rg/hom",sub{$liq[$_]==1});
```

Procedure MPens

Cette fonction retourne une pension moyenne, avec les mêmes types d'arguments que pour la fonction `SPens`.

Exemples d'appel :

```
$PMOY[$t]=MPens;
$PMOY[$t]=MPens("rg/fem");
$PMOY[$t]=MPens(sub{$liq[$_]==1});
$PMOY[$t]=MPens("rg/hom",sub{$liq[$_]==1});
```

IV.5 Sauvegarde des résultats macros dans des fichiers Excel

Pour exporter les résultats des fonctions de tabulation, l'utilisateur dispose des fonctions de la la bibliothèque `OutilsExcel.pm`. Cette bibliothèque lit ou écrit différents formats de tableaux dans des fichiers Excel. Ses programmes s'appellent en toute fin de programme. Dans l'exemple introductif, ceci correspondait aux lignes de code :

```
$resultats = ClassOut("Exemple.xls");
SavSer($resultats,"Resultats","Principaux agregats",
                  "Pension moyenne    ",@PENSION_MOYENNE,
```

```
                    "Ratio financier    ",@RATIO_FIN,
                    "Ratio démographique",@RATIO_DEM) ;
CloseOut($resultats);
```

L'autre exemple suivant, un peu plus riche, combine sorties de résultats se présentant sous formes de séries temporelles ou sous forme de profils par âge :

```
$resultats = ClassOut("TestCentral.xls");

# sauvegarde de séries temporelles
SavSer($resultats,"Totaux","Masses de pensions (en % du PIB)",
                    "Ensemble des droits directs ",@TOTDIR,
                    "Ensemble des droits dérivés ",@TOTDER);
SavSer($resultats,"Moyennes","Pensions annuelles moyennes (en euros)",
                    "Ensemble des droits directs ",@MOYDIR,
                    "Ensemble des droits dérivés ",@MOYDER);
SavSer($resultats,"Effectifs","Effectifs de bénéficiaires",
                    "Ensemble des droits directs ",@NBDIR,
                    "Ensemble des droits dérivés ",@NBDER);

# sauvegarde de séries par génération
SavSer($resultats,"MoyLiq","Pensions moyennes des liquidants",
                         "Nbre liquidants            ",@NBLIQ_G,
                         "Pension de droits directs  ",@PENSLIQ_G);

# sauvegarde de séries par âge
SavPro($resultats,"Pension_A","Données par age en 2003 et 2008",
                    "Population                  ",@POP_2003,
                    "Taux d'actifs occupés       ",@TXOCC_2003,
                    "Pension RG en 2003          ",@MOYRG_2003);

CloseOut($resultats);
```

Mais la bibliothèque permet aussi l'écriture de tableaux plus complexes : tableaux par âge et période, hachages de series temporelles, séries temporelles de hachages, hachages de hachages… On va se borner ici à présenter les plus usuelles des procédures de cette bibliothèque, le reste des procédures disponibles étant présenté en section VI.7.

Fonction ClassOut

Cette fonction ouvre un nouveau classeur en écriture.

Exemple d'appel

```
$resul= ClassOut("resultats.xls");
```

On crée ici une variable `$resul` que l'on indiquera lorsque l'on utilisera les fonctions d'écriture des résultats. Ce fichier s'appelle dans cet exemple `resultats.xls`. Il est créé automatiquement dans le répertoire où se trouve le programme principal.

Attention, il y a toujours création d'un nouveau fichier (la bibliothèque ne gère pas la réécriture dans un fichier existant).

Fonction CloseOut

Cette fonction ferme un classeur ouvert en écriture.

Exemple d'appel

```
CloseOut($resul);
```

Fonction SavSer

La fonction `SavSer` sauvegarde un ensemble de séries par années ou par génération dans une feuille Excel. Les trois premiers arguments sont le nom du classeur (du fichier Excel), le nom de la feuille et le titre de la feuille. Les arguments facultatifs sont répétés autant de fois qu'il y a de variables de calculs à insérer dans la feuille : le titre de la colonne et le nom de la variable contenant le résultat du calcul.

Exemple d'appel

```
SavSer($resul,"Feuil2","Titre",
              "Poptot   ",@poptot,
              "Pib      ",@pib,
              "Salaires ",@sal);
```

Dans cet exemple on sauvegarde, dans le fichier défini par la variable `$resul`, dans la feuille intitulée `Feuil2`, un tableau dont le titre est `titre` :

- une colonne intitulée `Poptot` comprenant la série correspondant au tableau `@poptot`,
- une colonne intitulée `Pib` comprenant la série correspondant au tableau `@pib`,
- une colonne intitulée `Salaires` comprenant la série correspondant au tableau `@sal`.

Le résultat se présentera sous la forme suivante

	A	B	C	D	E	F	G
1	Titre						
2		Poptot	Pib	Salaires			
3	2003	..	..	..			
4	2004	..	..	..			
5	2005	..	..	..			

Fonction SavPro

Cette fonction sauvegarde un ensemble de profils par âge dans une feuille Excel. On obtient un fichier Excel contenant dans chaque colonne un âge et chaque ligne une variable.

Exemple d'appel

```
SavPro($resul,"Feuil2","Titre"
              "Nombre"  ,@prod,
              "Salaire ",@sal);
```

Le résultat se présentant cette fois sous la forme suivante, avec les séries en ligne plutôt qu'en colonne :

	A	B	C	D	E	F	G
1	Titre						
2		0	1	2	3	4	5
3	Nombre	..	..	..	..	..	..
4	Salaire	..	..	..	..	..	..

Fonction SavHS

Cette fonction est celle qui permet de sauvegarder directement les hachages de séries temporelles dont on a vu des exemples en section IV.4.2. Par exemple si, dans une boucle sur `$sc`, on a généré le hachage `%pop` d'élément générique `$pop{$sc}[$t]`, on pourra sauvegarder d'un bloc ce hachage par l'instruction :

```
SavHS($resul,"Popul","Population, selon scenarios",%pop);
```

Le tableau contient autant de colonnes que de scénarios simulés.

	A	B	C	D	E	F	G
1	Population, selon scénarios						
2		Scénario 1	Scénario 2	Scénario 3			
3	2003	..	..	..			
4	2004	..	..	..			
5	2005	..	..	..			

On l'utilise de la même façon pour sauvegarder des hachages donnant la même série temporelle pour diverses sous populations, par exemple, avec la variable `%MOYDIRS` proposée en section IV.4.2

```
SavHS($resul,"MOYDIRHF","Pensions de droits directs ",%MOYDIRS);
```

qui donnera la feuille Excel :

	A	B	C	D	E	F	G
1	Pensions de droits directs						
2		Hommes	Femmes				
3	2003	..	..				
4	2004	..	..	..			
5	2005	..	..	..			

IV.6 Sauvegarde de résultats macros dans des fichiers texte

L'utilisateur de Destinie peut souhaiter sauvegarder ses résultats dans un fichier texte plutôt que dans un fichier Excel s'il considère que les fonctions de sauvegarde dans Excel sont trop limitées. Ceci suppose qu'il maîtrise les entrées-sorties sous Perl. On renvoie à des documentations plus complètes du langage pour une vue complète de ces entrées-sorties et on se borne à donner ci-après l'exemple d'un élément de programme donnant un tableau de résultats pour un nombre limité d'années.

```
open (SOR,">compar.txt");

@dates=(106,115,120,130,140,150);

printf SOR "\nNombre de pensionnés\n" ;
for (@dates) { $ann=1900+$_;  printf SOR "\t $ann"; };
printf SOR "\n CNAV" ;for (@dates) {printf SOR "\t $NBRG[$_]"};
printf SOR "\n FP"   ;for (@dates) {printf SOR "\t $NBFP[$_]"};
printf SOR "\n ARRCO";for (@dates) {printf SOR "\t $NBAR[$_]"};
printf SOR "\n AGIRC";for (@dates) {printf SOR "\t $NBAG[$_]"};

printf SOR "\nPension moyenne\n" ;
for (@dates) { $ann=1900+$_;  printf SOR "\t $ann";};
printf SOR "\n CNAV" ;for (@dates) {printf SOR "\t $MOYRG[$_]"};
printf SOR "\n FP"   ;for (@dates) {printf SOR "\t $MOYFP[$_]"};
printf SOR "\n ARRCO";for (@dates) {printf SOR "\t $MOYAR[$_]"};
printf SOR "\n AGIRC";for (@dates) {printf SOR "\t $MOYAG[$_]"};
```

Ce segment de programme utilise la fonction Perl `open` qui ouvre un fichier en lui associant ici le nom logique `SOR`. Contrairement aux autres variables, il n'a pas besoin d'être prédéclaré même sous la directive `use strict`. On note aussi le `">"` avant le nom du fichier indiquant que celui-ci est ouvert en écriture.

L'autre fonction est la fonction `print` qui utilise par ailleurs les séquences d'échappement `\n` (saut de ligne) et `\t` (tabulation). On notera également la forme

```
print SOR "\t $MOYRG[$_]"
```

qui correspond à ce qu'on appelle une interpolation de variable dans une chaîne : la valeur `$MOYRG[$_]` est traduite en chaîne de caractère au sein de la chaîne entre doubles cotes. Une écriture également correcte aurait été :

```
print SOR "\t", $MOYRG[$_]
```

mais la première est plus légère lorsqu'on enchaine la sortie de plusieurs variables séparées par autant d'éléments de texte.

L'output de ce bloc de programme aura l'apparence suivante :

```
Nombre de pensionnés
        2006   2015   2020   2030   2040   2050
CNAV    9127   10828  12298  14693  16339  17736
FP      1812   2321   2608   3135   3662   4050
ARRCO   7934   10116  11808  14416  16099  17597
AGIRC   1794   2099   2164   2469   2728   3236

Pension moyenne
        2006   2015   2020   2030   2040   2050
CNAV    7923   7993   8139   8832   10137  11407
FP      23702  22217  22903  23203  27581  31904
ARRCO   3747   3438   3233   3112   3310   3590
AGIRC   10808  9928   9831   11514  13354  13681
```

Pour des mises en page encore plus travaillées, il existe également en Perl une instruction `printf` permettant des impressions formatées.

V - Utilisations courantes (2) : inspecter les données individuelles

L'interprétation des résultats d'une simulation est parfois complexe et peut nécessiter l'examen de cas individuels spécifiques. Il arrive en effet que les barèmes produisent des résultats peu intuitifs pour certains types de profils individuels.

Pour examiner des résultats individu par individu, il existe deux possibilités. L'une consiste à prévoir des affichages ou des sauvegardes de résultats individuels en cours de simulation, à l'intérieur donc du programme `.pl`. L'autre consiste à recourir à un programme interactif, l'explorateur, qui permet de balayer le fichier `BioEmp` individu par individu et de tester différentes hypothèses de départ en retraite sur les individus qu'on examine. On va commencer par présenter le fonctionnement de cet explorateur (section V.1) puis on donnera quelques indications sur la façon d'afficher ou sauvegarder des résultats individuels depuis un programme `.pl`.

V.1 L'explorateur

L'explorateur a été mis au point pour permettre un examen rapide et interactif des données individuelles. Il ne procède pas à une simulation complète, mais il lit et stocke en mémoire l'ensemble des carrières figurant dans un fichier `BioEmp` et la situation familiale initiale du fichier `BioFam`. Avec ces informations en mémoire, il est possible d'afficher toutes sortes de données individuelles à la demande, et de procéder à divers essais de simulation du départ en retraite à n'importe quel horizon, puisque `BioEmp` contient à la fois les carrières rétrospectives et prospectives, avec affichage de l'ensemble des résultats associés.

On peut aussi utiliser l'explorateur pour afficher n'importe quelle statistique relative à l'état initial de la population, en demandant l'impression à l'écran de résultats d'une quelconque des procédures de la bibliothèque `OutilsTab`. En fait, c'est l'intégralité des procédures des bibliothèques Destinie qui sont accessibles depuis l'explorateur, dès lors que leur application à un sens sur les données que cet explorateur stocke en mémoire et dès lors qu'elles ne mobilisent pas plus d'une ligne de code.

L'explorateur se lance en ouvrant le fichier Explore.pl du sous-répertoire Outils avec le compilateur Perl. Ceci déclenche l'ouverture d'une fenêtre MS-DOS à l'intérieur de laquelle se déroule l'ensemble des opérations qui suivent. La figure 6 l'exemple du début d'une session de l'explorateur.

La première demande de l'explorateur est de lui indiquer dans quel sous-répertoire de biographies se trouvent les fichiers `BioEmp` et `BioFam` à charger. Il donne pour ce faire la liste complète des sous-répertoires du répertoire `Bios`, et la sélection se fait par numéro. Une fois les contenus des deux fichiers chargés, l'explorateur informe du nombre d'individus lus dans chacun d'entre eux, dont le nombre d'individus effectivement présents dans la population à la date de départ. Le premier est supérieur au second puisque le fichier `BioEmp.txt` comprend également les débuts de trajectoires des individus dont la date de naissance est supérieure à cette date de départ.

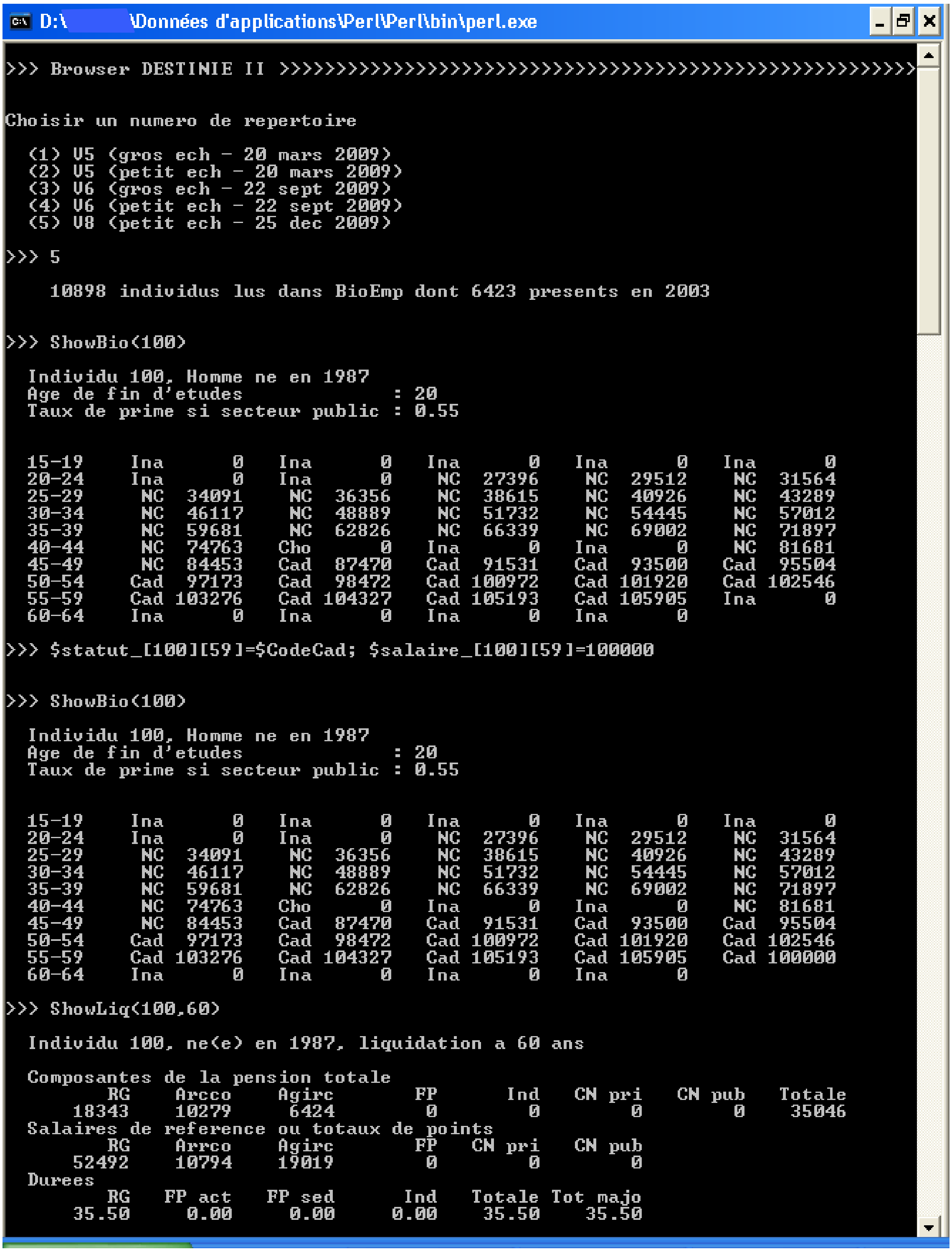
Une fois ceci fait, l'explorateur émet des invites de commande auxquelles on répond en soumettant des textes de commandes Perl utilisant les différentes variables Destinie. Par exemple :

```
>>> print $age[$pere[27]]
```

qui affichera l'âge du père de l'individu 27, s'il est dans la population. On peut aussi bien écrire :

```
>>> $id=27 ;
>>> print $age[$pere[$id]]
```

Figure 7 : Exemple de mise en œuvre de l'explorateur



```
D:\[redacted]\Données d'applications\Perl\Perl\bin\perl.exe

>>> Browser DESTINIE II >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

Choisir un numero de repertoire

  (1) V5 (gros ech - 20 mars 2009)
  (2) V5 (petit ech - 20 mars 2009)
  (3) V6 (gros ech - 22 sept 2009)
  (4) V6 (petit ech - 22 sept 2009)
  (5) V8 (petit ech - 25 dec 2009)

>>> 5

    10898 individus lus dans BioEmp dont 6423 presents en 2003

>>> ShowBio(100)

  Individu 100, Homme ne en 1987
  Age de fin d'etudes            : 20
  Taux de prime si secteur public : 0.55

  15-19    Ina      0   Ina      0   Ina      0   Ina      0   Ina      0
  20-24    Ina      0   Ina      0    NC  27396    NC  29512    NC  31564
  25-29     NC  34091    NC  36356    NC  38615    NC  40926    NC  43289
  30-34     NC  46117    NC  48889    NC  51732    NC  54445    NC  57012
  35-39     NC  59681    NC  62826    NC  66339    NC  69002    NC  71897
  40-44     NC  74763   Cho      0   Ina      0   Ina      0    NC  81681
  45-49     NC  84453   Cad  87470   Cad  91531   Cad  93500   Cad  95504
  50-54    Cad  97173   Cad  98472   Cad 100972   Cad 101920   Cad 102546
  55-59    Cad 103276   Cad 104327   Cad 105193   Cad 105905   Ina      0
  60-64    Ina      0   Ina      0   Ina      0   Ina      0

>>> $statut_[100][59]=$CodeCad; $salaire_[100][59]=100000

>>> ShowBio(100)

  Individu 100, Homme ne en 1987
  Age de fin d'etudes            : 20
  Taux de prime si secteur public : 0.55

  15-19    Ina      0   Ina      0   Ina      0   Ina      0   Ina      0
  20-24    Ina      0   Ina      0    NC  27396    NC  29512    NC  31564
  25-29     NC  34091    NC  36356    NC  38615    NC  40926    NC  43289
  30-34     NC  46117    NC  48889    NC  51732    NC  54445    NC  57012
  35-39     NC  59681    NC  62826    NC  66339    NC  69002    NC  71897
  40-44     NC  74763   Cho      0   Ina      0   Ina      0    NC  81681
  45-49     NC  84453   Cad  87470   Cad  91531   Cad  93500   Cad  95504
  50-54    Cad  97173   Cad  98472   Cad 100972   Cad 101920   Cad 102546
  55-59    Cad 103276   Cad 104327   Cad 105193   Cad 105905   Cad 100000
  60-64    Ina      0   Ina      0   Ina      0   Ina      0

>>> ShowLiq(100,60)

  Individu 100, ne(e) en 1987, liquidation a 60 ans

  Composantes de la pension totale
         RG     Arrco     Agirc        FP       Ind    CN pri    CN pub    Totale
      18343     10279      6424         0         0         0         0     35046
  Salaires de reference ou totaux de points
         RG     Arrco     Agirc        FP    CN pri    CN pub
      52492     10794     19019         0         0         0
  Durees
         RG    FP act    FP sed       Ind   Totale Tot majo
      35.50      0.00      0.00      0.00    35.50    35.50
```

Lecture : Dans cet exemple, l'explorateur affiche la biographie de l'individu n° 100, modifie les données professionnelles relatives à la 59eme année de cet individu, affiche la biographie modifiée et donne la pension qui en découle dans l'hypothèse d'un départ à 60 ans

Dans cet exemple, la variable `$id` n'a pas eu besoin d'être prédéclarée car l'explorateur fonctionne sans la directive `use strict` qui force la déclaration des variables. Pour la même raison, on pourra écrire une séquence :

```
>>> $effectif=Count;
>>> print "L'effectif initial est de ",$effectif
```

qui équivaudra à :

```
>>> print "L'effectif initial est de ", Count
```

On peut également entrer une suite de séquences générant des résultats, puis ouvrant un classeur Excel et stockant ces résultats dans ce classeur. Dès qu'on ferme ce classeur par `CloseOut`, il est récupérable sous Excel pour une analyse immédiate.

Mais l'usage principal de l'explorateur est l'affichage de profils individuels ou de détails du calcul des droits à retraite, qui se fait à l'aide des deux fonctions `ShowBio` et `ShowRet` qui sont des fonctions de la bibliothèque `OutilsRetr`, et donc également accessibles depuis un programme quelconque, mais qu'on présente dans cette section puisque c'est dans le cadre de l'explorateur que leur usage est le plus naturel.

Fonction ShowBio

Cette fonction affiche la biographie professionnelle complète d'un individu. Les éléments affichés sont l'identifiant, le sexe et l'année de naissance, l'âge de fin d'études et le taux de prime (indépendant de l'âge) dans le cas où l'individu travaille dans le secteur public. Elle affiche ensuite un tableau complet de ses statuts (en clair) et salaires par âge.

La fonction a deux arguments d'appel. Le premier argument obligatoire est l'identifiant de l'individu. Le second, optionnel, définit le mode d'affichage des salaires. Ceux-ci peuvent être affichés en valeur nominale (option par défaut), en valeur réelle (option `"reel"`) ou en pourcentage de la valeur courante du plafond de la sécurité sociale à chaque âge (option `"%plaf"`).

Fonction ShowPens

Cette fonction affiche la pension qu'aurait l'individu en liquidant à un âge donné et les différents éléments intermédiaires de son calcul. Les deux arguments obligatoires sont l'identifiant de l'individu et l'âge auquel on veut simuler la liquidation. Un troisième argument optionnel définit la législation à mettre en œuvre pour ce calcul de la pension. Par défaut, il s'agit de la législation en vigueur à l'âge de liquidation. Les deux autres options sont soit une autre date, soit la chaine vide `""` qui désactive tout choix de législation par ce programme `ShowPens` et permet donc de simuler l'incidence d'une législation sur mesure prédéfinie par des instructions précédentes.

Par exemple, supposons que l'individu 20 soit né en 1950. L'instruction

```
>>> ShowPens(20,60)
```

donnera les différents éléments de la liquidation pour l'individu 20 s'il part, en 2010, à l'âge de 60 ans, avec la législation prévue à cette date pour les individus de cette génération.

L'instruction alternative

```
>>> ShowPens(20,60,1992)
```

donne les éléments de liquidation au même âge avec une législation bloquée aux règles prévues en 1992.

La séquence d'instructions

```
>>> UseLeg(1992,1950)
>>> $DureeCalcSam=15
>>> ShowPens(20,60, "")
```

donnera enfin la retraite avec un départ selon la législation de 1992 mais une durée de calcul du SAM portée à 15 ans au lieu de 10.

L'utilisation de la fonction `ShowPens` appelle deux remarques supplémentaires :

- Cette fonction `ShowPens` ne travaille pas directement sur les données de l'individu passé en argument mais sur les données d'une copie de cet individu d'identifiant 0. Les données relatives à l'individu traité sont donc laissées inchangées, ce qui autorise des appels successifs avec différentes hypothèses d'âge ou de législation. Si on voulait simuler un départ effectif en retraite d'un individu sous l'explorateur, il faudrait recourir à la fonction `SimDir`, comme dans les programmes `.pl` vus jusqu'ici.

- Sauf désactivation préalable par `UseOpt`, cette fonction intègre les avantages familiaux, mais sur la base de la situation familiale observée au départ de la simulation, qui n'intègre donc pas les évolutions ultérieures de cette situation familiale. Cela suffit néanmoins à capter une part substantielle de ces droits familiaux. On peut aussi forcer des droits supplémentaires en manipulant les variables de liens familiaux de l'individu considéré même si ceci est à faire avec prudence.

Autres fonctionnalités de l'explorateur

L'explorateur accepte la commande `help` qui, à ce stade, se borne à retourner la syntaxe des deux fonctions `ShowBio` et `ShowRet`.

À l'intérieur de la fenêtre ouverte par l'explorateur, on dispose des facilités de base d'une fenêtre MS-DOS, par exemple l'ascenseur, le rappel et la modification d'une commande antérieure à l'aide des curseurs ↑ et ←, la fonction de recherche d'une zone de texte ou le copier-coller vers un autre fichier.

La sortie de l'explorateur se fait soit par fermeture de la fenêtre de travail soit par la commande `exit`.

V.2 Sauvegarder des résultats individuels dans un fichier texte

L'autre façon d'accéder aux résultats individuels est de demander leur affichage ou leur impression dans un fichier à l'intérieur du programme de simulation. Supposons par exemple qu'on veuille se focaliser sur les individus avec des valeurs du taux de remplacement atypique. On peut insérer, après `SimDir`, un bloc du type :

```
If (($liq[$i]==1) && (TauxRemp($i)>100))
{
   print join ("\t",$i,$anaiss[$i],$pension[$i],$sam,$duree_rg), "\n"
}
```

Dans cette instruction, on notera l'utilisation de la fonction `join` qui sert à concaténer un ensemble de valeurs dans une chaîne de caractères unique avec comme séparateur le premier argument d'appel.

Si le programme est lancé depuis l'environnement `PerlIde`, il y aura affichage dans sa fenêtre `console` d'une ligne pour chaque individu liquidant l'année en cours avec un taux de remplacement supérieur à l'unité. Les éléments affichés seront l'identifiant, l'année de naissance, la pension, le SAM et durée au régime général. On peut alternativement demander dans la même fenêtre l'affichage complet des biographies et éléments de pension de ces individus par un appel à `ShowBio` ou `ShowPens`, ou demander l'impression des mêmes éléments dans un fichier `.txt` ouvert avec l'instruction `open` décrite plus haut en IV.6.

On peut également procéder à des affichages individuels en fin de boucles, sur des variables individuelles permanentes. Par exemple, une variable micro telle que `@pension` peut avoir été déclinée en un hachage `%pension` contenant la valeur de la pension à la liquidation pour plusieurs scénarios, et on peut, en fin de programme, constituer un fichier donnant, sur la même ligne, les niveaux de pension des différents scénarios individu par individu, en même temps que d'autres descripteurs des caractéristiques individuelles, le tout pouvant être ensuite relu par d'autres instruments, tels que SAS, pour des analyses plus détaillées des caractéristiques des perdants ou gagnants aux différents types de réformes.

La séquence pour parvenir à ce type de résultats sera :

```
my ($sc,$i,$t,%pension);
    …
for $sc ("ref","var1","var2")
{
   …
   for $t (103..150)
   {
      Relec($t);
      for $i (1..Imax)
      {
           SimDir($i)
      }
      if ($liq[$i]==1)
      {
         $pension{$sc}[$i]=$pension[$i]
      }
   }
}

open COMP,">compar.txt";
for $i (1..IMax)
{
   print COMP join("\t",$i,$anaiss[$i],$sexe[$i],$findet[$i],
                       $pension{"ref"}[$i],
                       $pension{"var1"}[$i],
                       $pension{"var2"}[$i]),"\n";
}
close COMP;
```

VI - Utilisations avancées (1) : présentation détaillée de l'ensemble des bibliothèques

Cette partie du document va maintenant détailler l'ensemble des fonctions disponibles dans la boite à outils Destinie 2. On va commencer par le contenu intégral de quatre bibliothèques qui n'ont pas encore été détaillées car non indispensables pour les utilisations les plus courantes, à savoir `OutilsDroits`, `OutilsComp`, `OutilsMen` et `OutilsBase`. Puis on examinera les éléments des bibliothèques `OutilsRetr`, `OutilsTab` et `OutilsExcel` qui n'ont pas été encore présentés dans les sections précédentes.

Cette partie permettra d'avoir une vue d'ensemble des instruments préfabriqués dont dispose l'utilisateur pour construire ses programmes `.pl` et/ou effectuer des calculs interactifs à l'intérieur de l'explorateur. En revanche, elle ne détaille pas la façon dont ces différents instruments sont eux-mêmes programmés. Pour quelques exemples de cette programmation, on renvoie à la partie VII.

VI.1 La bibliothèque OutilsDroits : les outils de calcul des droits

Cette bibliothèque est celle qui assure le calcul des droits à retraite, à la fois pour le secteur privé et le secteur public. Il s'agit de calculs de droits à âge de liquidation donné. Les âges de liquidation sont calculés par la bibliothèque `OutilsComp`. `OutilsRetr` (surtout `SimDir`) combine `OutilsComp` et `OutilsDroits` pour simuler la liquidation et l'évolution des pensions.

Cette bibliothèque OutilsDroits travaille à la fois sur les variables individuelles indicées et non indicées de `DefVarRetr.`

Un premier sous-ensemble de fonctions sert à calculer les variables intermédiaires non indicées. Il s'agit de :

`Duree`	: calcul de la durée passée dans un statut quelconque
`DurBase`	: calcul des durées de base (durées cotisées)
`DurMajo`	: calcul des durées majorées des droits familiaux
`AllocChom`	: calcul des allocations chômage pour le SAM
`SalBase`	: calcul du SAM et du salaire de référence FP
`Points`	: calcul des décomptes de points ARRCO et AGIRC (et comptes notionnels, le cas échant)
`MinCont`	: calcul du minimum contributif
`MinGaranti`	: calcul du minimum garanti

Les quatre fonctions suivantes gèrent les âges d'accès à la retraite. Il s'agit de :

`AgeTrim`	: retourne un âge trimestrialisé
`AgeMin`	: indique si l'individu a atteint l'âge minimum d'ouverture des droits
`AgeMax`	: indique si l'individu a dépassé l'âge maximum d'activité
`TauxPlein`	: indique si l'individu a atteint les conditions du taux plein

Les deux fonctions suivantes s'appuient sur les précédentes pour calculer les droits à retraite ou les réévaluer. Il s'agit de :

`Liq`	: calcul des droits complets à la liquidation
`Revalo`	: revalorisation de l'ensemble des droits (y.c. réversion)

Il existe également des fonctions calculant différents indicateurs dérivés pouvant servir à l'analyse des droits :

`CotRet`	: calcule les cotisations retraite sur salaires
`CotAut`	: calcule la somme des autres cotisations sur salaires

`CSGSal`	: calcule la CSG/CRDS sur salaires
`CSGRet`	: calcule la CSG/CRDS sur pensions
`SalNet`	: calcule le salaire net
`PNet`	: calcule la pension nette
`SalMoy`	: calcule la moyenne des n derniers salaires
`SalCum`	: calcule la somme des n derniers salaires
`Pente`	: calcule un indice de pente de la carrière
`TauxRemp`	: calcule un taux de remplacement rapporté aux n derniers salaires
`TauxAnn`	: calcule un taux d'annuité effectif, défini comme le ratio entre le niveau de pension et la masse des salaires touchés durant la vie active

Toutes ces procédures ont au moins un argument d'appel qui est l'identifiant de l'individu. Le calcul se fait par défaut à l'âge courant de cet individu, soit `$age[$i]`. Ainsi, si on se trouve à l'intérieur d'une double boucle sur `$t` et `$i`, la fonction `TauxPlein($i)`, par exemple, nous indique si l'individu `$i` dont l'âge en `$t` est `$age[$i]` a atteint, à cette date, les conditions du taux plein, telles que définies par le dernier appel de la fonction `UseLeg` de `DefVarRetr`. Mais il est aussi possible, dans certain cas, de forcer le calcul pour un âge autre que l'âge courant, qui est alors fourni par le deuxième paramètre d'appel. Il existe enfin d'autres paramètres optionnels qu'on décrit ci-après au cas par cas.

VI.1.1 Fonctions de calcul de variables intermédiaires des droits directs

Fonction Duree

La fonction Duree est une fonction utilitaire qui permet de calculer la durée passée par un individu dans un statut donné. Elle permet de calculer combien de temps un individu a été dans un statut donné entre 15 ans et un âge quelconque. Les arguments de cette fonction sont l'identifiant de l'individu, l'âge auquel on arrête le comptage de l'individu (le plus souvent `$age[$i]`) et le ou les statut(s) à prendre en compte. Cette fonction `Duree` est essentiellement appelée dans les fonctions `DurBase` et `DurMajo`, présentées ci-dessous. Mais elle peut également être appelée depuis le programme principal si l'utilisateur a besoin d'une durée que ces fonctions ne calculent pas.

Exemple d'appel :

```
$duree_cho=Duree($i,$age[$i],$CodeCho) ;
```

calculera le temps pendant lequel un individu a été au chômage jusqu'à son âge courant.

```
$duree_act=Duree($i,$age[$i],@CodesAct) ;
```

calculera de la même manière la durée passée en activité.

Fonction DurBase

La fonction `DurBase` calcule les durées cotisées dans les différents régimes, les durées en emploi, au chômage, en préretraite et la durée pendant laquelle l'individu a validé des années à l'AVPF. Elle passe en revue toutes les années de la carrière et incrémente les durées à calculer au fur et à mesure.

Elle modifie les valeurs des scalaires `$duree_rg`, `$duree_fp`, `$duree_fpa`, `$duree_fps`, `$duree_in`, `$duree_tot`, `$duree_avpf`, `$duree_emp`, `$duree_PR` et `$duree_cho`.

Fonction DurMajo

La fonction `DurMajo` repart des durées calculées par `DurBase`, et calcule les durées majorées prenant en compte la Majoration de Durée d'Assurance (MDA) et/ou l'Allocation Vieillesse des Parents au Foyer (AVPF) selon les options qui ont été précisées dans `UseOpt`. Elle met à jour les différentes durées déjà calculées par `DurBase`.

Fonction SalBase

La fonction `SalBase` calcule les salaires servant de base au calcul des pensions de base, c'est-à-dire le SAM pour le RG et les régimes d'indépendants et le salaire de référence pour la fonction publique. Les résultats sont stockés dans `$sam_rg`, `$sam_in` et `$sr_fp`.

Fonction Points

La fonction `Points` calcule le nombre de points accumulés dans les régimes ARRCO et AGIRC et stocke les résultats dans les variables `$points_arrco` et `$points_agirc`. Elle calcule aussi, s'il y a lieu, les totaux de points dans les régimes en comptes notionnels, si on simule la mise en place de tels régimes, selon les options présentées plus loin en VI.5.

Fonction Mincont

La fonction `Mincont` calcule le montant attribuable au titre du minimum contributif. Elle suppose que les durées de cotisation sont connues, donc des appels préalables de `DurBase` et `DurMajo` pour pouvoir utiliser les durées majorées au régime général et les durées validées tous régimes.

Le calcul utilise les paramètres `$Mincont1[$t]`, `$Mincont2[$t]` de `ParamSociaux` et `$DureeProratRG` créé par `UseLeg`.

Fonction MinGaranti

La fonction `MinGaranti` calcule le montant attribuable au titre du minimum garanti. Elle tient compte de la période transitoire entre 2003 et 2013 avec application progressive de la réforme. Elle suppose que `$duree_fp` ait été renseignée, i.e. un appel préalable de `DurBase`.

VI.1.2 Fonctions de calcul de variables intermédiaires pour l'âge de liquidation

Fonction AgeTrim

Cette fonction retourne l'âge trimestrialisé de l'individu `$i` sur la base de son âge annuel et de son trimestre de naissance. Destinie travaille en pas annuel, mais l'utilisation d'un âge trimestrialisé est nécessaire lorsqu'on veut éviter des sauts trop importants de comportements moyens lorsque les seuils d'âge franchissent des valeurs entières. Il utilise en input la variable `$trim[$i]` donnant le trimestre de naissance impute au moment de l'appel de `Relec`.

Fonction AgeMin

La fonction `AgeMin` indique si l'individu a atteint l'âge minimum d'ouverture des droits (60 ans dans le privé et 55 ans dans la fonction publique active). Elle relance pour ce faire les procédures `DurBase`. Elle retourne un booléen mais ne modifie aucune variable. Elle est utilisée dans la fonction `TestLiq`.

Exemple d'appel :

```
if (AgeMin($i)) {... }
```

Fonction AgeMax

Retourne l'âge maximal de départ en retraite applicable à l'individu `$i.`

Exemple d'appel :

```
if ($age[$i]>=AgeMax($i)) {... }
```

Fonction TauxPlein

Indique si l'individu remplit les conditions du taux plein. Elle relance pour ce faire les procédures `DurBase` et `DurMajo`. Elle retourne un booléen mais ne modifie aucune variable.

Exemple d'appel :

```
if (TauxPlein($i)) {... }
```

VI.1.3 Fonctions de calcul des montants de pension

Fonction Liq

Simule l'effet de la liquidation à l'âge courant. Cette fonction n'a pas de valeur de retour. Elle modifie directement les variables globales `$pension_rg[$i]`, `$pension_fp[$i]` ... ainsi que `$pension[$i]` et `$ageliq[$i]`. Elle modifie aussi les variables `$statut_[$i][$age[$i]]` et `$salaire_[$i][$age[$i]]` ainsi que leurs équivalents `$statut[$i]` et `$salaire[$i].`

Après avoir lancé toutes les fonctions intermédiaires (durées validées, salaire de référence et SAM, points, minimum garanti et minimum contributif) cette fonction `Liq` calcule la distance au taux plein.

Elle comprend ensuite plusieurs sous-modules : pour le calcul de la pension au RG, pour les pensions ARRCO et AGIRC, pour les pensions civiles, pour les pensions des indépendants.

Pour finir, elle procède aux mises à jours finales des variables de montant des pensions, du statut et des salaires et de l'âge de liquidation.

Exemple d'appel :

```
Liq($i);
```

Fonction Revalo

La fonction `Revalo`, comme son nom l'indique, revalorise les composantes de la pension de l'individu `$i`. Par défaut la revalorisation est faite de `$age[$i]-1` à `$age[$i]`. Si on veut une revalorisation cumulée sur plusieurs années, on précise à la fois l'âge d'arrivée et l'âge de départ

Elle revalorise les pensions de base et les pensions de réversion en appliquant le coefficient de revalorisation du régime général ou de la fonction publique, dont les valeurs sont fournies dans `ParamSociaux`.

Elle revalorise également les pensions des régimes complémentaires en tenant compte de l'évolution de la valeur des points dans chaque régime.

Cette fonction s'applique à la fois aux droits directs et à la réversion.

Exemples d'appel

```
Revalo($i);
```

calcule les composantes de la pension à `$age[$i]` en supposant que leurs valeurs avant appel sont celles de l'âge `$age[$i]-1`.

```
Revalo($i,60,65)
```

calcule les composantes de la pension à 65 ans en supposant que leurs valeurs avant appel sont celles atteintes par `$i` à 60 ans.

VI.1.4 Calcul d'indicateurs dérivés

Fonction CotRet

Cette fonction calcule la somme des cotisations retraite de l'individu `$i` à l'âge courant ou (optionnel) à un âge quelconque (il s'agit des cotisations salariés)

Exemples d'appel :

```
$cotret_[$i][$a]= CotRet($i,$a);
$cotret[$i]     = CotRet($i);
```

Fonction CotAut

Cette fonction calcule les cotisations sur salaire autres que retraite. Même principe que `CotRet.`

Fonction CSGSal

Cette fonction calcule la CSG-CRDS sur le salaire. Même principe que `CotRet` et `CotAut`.

Fonction CSGRet

Cette fonction calcule la CSG-CRDS sur la pension. Contrairement à `CotRet` et `CotAut`, ne s'applique qu'à la pension courante.

Exemple d'appel :

```
$csg[$i] = $CSGRet($i);
```

Fonction SalNet

Cette fonction calcule le salaire net de tous prélèvements (cotisations retraites, autres cotisations sociales et CSG/CRDS), à l'âge courant ou à un âge quelconque (optionnel).

Fonction PNet

Cette fonction calcule la pension nette de prélèvements (CSG et CRDS), uniquement à l'âge courant.

Fonction SalMoy

Cette fonction calcule la moyenne des salaires des n dernières années de la carrière (par défaut l'ensemble de la carrière). Le premier argument est l'indice de l'individu. Le second indique si on veut des salaires bruts ou nets. Le troisième argument, optionnel, donne le nombre n s'il ne s'agit pas de l'ensemble de la carrière. Un quatrième argument optionnel donne la série temporelle sur la base de laquelle on revalorise ces salaires passés. Par défaut, il s'agit des prix. L'ordre des deux arguments optionnels est indifférent (ce qui permet de ne passer qu'un seul des deux).

Exemples d'appel :

```
$sal_moy[$i]=SalMoy($i,"brut",10);
$sal_moy[$i]=SalMoy($i,"net",@PlafondSS);
```

Fonction SalCum

Cette fonction calcule la somme des salaires des n dernières années de la carrière (par défaut l'ensemble de la carrière). Le principe de l'appel est le même que pour la fonction `SalMoy`.

Fonction Pente

Calcule un indice de pente du profil de salaire brut. Cet indice de pente est le ratio en % des n dernières années de carrière rapporté au salaire moyen de l'ensemble de la carrière. Par défaut, n est égal à 1. Les salaires moyens sont revalorisés par défaut comme les prix, ou selon la variable passée comme second argument optionnel

Exemples d'appel :

```
$pente[$i]=Pente($i);
$pente[$i]=Pente($i,10,@PlafondSS);
```

Fonction TauxRemp

Calcule le taux de remplacement, à n'utiliser donc que si `$liq[$i]==1`. Les arguments d'appel sont les mêmes que pour `SalCum` ou `SalMoy`.

Exemples d'appel :

```
if ($liq[$i]) {$taux_r_b[$i]=TauxRemp($i,"brut")};
if ($liq[$i]) {$taux_r_n[$i]=TauxRemp($i,"net",5,@PlafondSS)};
```

Fonction TauxAnn

Calcule le taux d'annuité apparent, i.e. le rapport entre la pension à la liquidation et le cumul des salaires passés. Le principe d'appel est le même que pour `TauxRemp`.

Exemples d'appel :

```
if ($liq[$i]) {$taux_ann[$i]=TauxAnn($i,"brut")};
if ($liq[$i]) {$taux_ann[$i]=TauxAnn($i,"net",@PlafondSS)};
```

VI.2 La bibliothèque OutilsComp : la simulation des comportements de départ à la retraite

Cette bibliothèque propose des outils de simulation des comportements de départ en retraite. Elle utilise les bibliothèques `OutilsBase.pm`, `DefVarRetr.pm` et `OutilsRetr.pm`. Contrairement aux programmes fournis dans `OutilsRetr` qui sont spécifiques au système français, les outils de cette bibliothèque sont des outils généraux qui pourraient être utilisés pour simuler des comportements dans des régimes entièrement différents.

La liste détaillée des fonctions disponibles est la suivante :

`Uinst`	: utilité instantanée selon statut courant
`Uact`	: utilité actualisée sur durée de vie restante selon âge de départ anticipé
`SSW`	: somme actualisée des droits en fonction de l'âge de départ anticipé
`Accrual`	: calcule le gain sur SSW lorsqu'on reporte la liquidation d'un an par rapport à l'âge courant
`Peak`	: calcule le SSW max pour tous les âges de départ au-delà de l'âge courant
`TestLiq`	: teste la décision de liquidation à l'âge courant avec cinq règles de comportement possibles
`AgeOpt`	: calcule âge optimal selon critères SSW ou UAct. Retourne également la valeur atteinte pour le critère qui est maximisé

VI.2.1 Les fonctions de premier niveau

Fonction UInst

La fonction `Uinst` retourne l'utilité dérivée du revenu dans l'état courant, du type :

$$U = revenu^{1-\gamma}/(1-\gamma)$$

où γ correspond au paramètre `$gamma[$i]` défini par `DefVarRetr` et initialisé par défaut à 0,5. La fonction gère le cas particulier γ=1. Le revenu est soit le revenu d'activité soit la pension. Mais la fonction ne prend pas en compte la préférence pour le loisir. Celle-ci est rajoutée au calcul par `TestLiq` lorsque l'option `Uinst` est activée.

Exemple d'appel :

```
UInst($i);
```

Fonction UAct

La fonction `Uact` retourne l'utilité actualisée sur la durée de vie restante selon l'âge de liquidation anticipé. Pour un âge anticipé de a_r sa forme est :

$$U_a(a,a_r) = \sum_{u=a}^{a_r}(1+\tau)^{a-u}s(u)\frac{w(u)^{1-\gamma}}{1-\gamma} + \sum_{u=a_r}^{\omega}(1+\tau)^{a-u}s(u)\frac{kp(a_r)^{1-\gamma}}{1-\gamma}$$

Les coefficients τ, k et γ correspondent aux variables `$taux[$i]`, `$k[$i]` et `$gamma[$i]` déclarées dans `DefVarRetr`, initialisés par défaut à 0,03, 1.5 et 0,5. Pour l'interprétation du rôle de ces paramètres, le lecteur est renvoyé au document général de présentation du modèle (Blanchet et al., 2010). La fonction de survie s(a) est prise égale à la survie à l'âge courant telle que lue dans `ParamMortal.xls`, différenciée selon le sexe. Les revenus générateurs d'utilité sont des revenus réels. La pension $p(a_r)$ est telle qu'évaluée avec la législation active au moment du lancement de la fonction.

Les paramètres passés à la fonction sont l'identifiant de l'individu et l'âge de liquidation anticipé.

Exemple d'appel :

```
UAct($i,$ageliq);
```

Fonction SSW

La fonction SSW retourne la richesse sécurité sociale sur la durée de vie restante selon l'âge de liquidation anticipé avec le taux `$taux[$i]`. Les paramètres sont l'identifiant de l'individu et l'âge de liquidation anticipée Si l'âge de liquidation anticipé n'est pas précisé, on suppose qu'il s'agit d'un individu déjà retraité et la procédure sert à calculer la valeur actualisée des droits restant à lui servir.

Cette fonction SSW calcule les droits cumulés en valeur à la date d'évaluation, mais corrigés de l'inflation anticipée (pas d'illusion monétaire en projection).

Exemple d'appel :

```
SSW($i,$ageliq);
```

Fonction Accrual

La fonction `Accrual` retourne le gain relatif de SSW à reporter d'un an au-delà de l'âge courant. Elle revient donc à appeler deux fois la fonction SSW, sous les hypothèses de départ à l'âge courant et un an plus tard, et à faire la différence des deux résultats. Un résultat nul correspond à la neutralité actuarielle (pour le taux d'actualisation retenu).

Exemple d'appel

```
$accrual= Accrual($i,60);
```

VI.2.2 Fonctions de calcul de l'âge de départ à la retraite : TestLiq et AgeOpt

La fonction `TestLiq` teste si le départ en retraite de l'individu `$i` a lieu à l'âge courant selon l'option de comportement choisie par `UseOpt`. La mise à la retraite est systématique à `$AgeMaxFP` ou `$AgeMaxRG`, selon le statut courant, et quelle que soit l'option choisie

Selon l'option indiquée dans `UseOpt`, la fonction `TestLiq` lance des fonctions liées à ce type de comportement de liquidation (`Uact`, `UInst`, SSW) ou la fonction de test du taux plein. La plupart de ces fonctions lancent la fonction `Liq` car elles nécessitent de connaître le montant des pensions si la personne liquidait. L'articulation entre cette fonction, la fonction `Liq` et la fonction `DroitsDir` sera réexpliquée plus en détail plus loin à la section VI.5.2.

La fonction `TestLiq` retourne une variable qui vaut 1 si la personne liquide l'année courante et 0 sinon.

Exemple d'appel :

```
if TestLiq($i) {...};
```

Fonction AgeOpt

La fonction `AgeOpt` retourne l'âge optimal au départ selon les critères `Uact` et SSW. Les paramètres sont les mêmes que pour `Liq` (sauf le paramètre `$seuil`, qui est ici sans objet).

VI.3 La bibliothèque OutilsMen : l'environnement familial des individus

Cette bibliothèque propose un certain nombre de programmes permettant de reconstituer les caractéristiques des ménages ou de l'environnement familial des individus. Il s'agit pour l'essentiel de caractéristiques démographiques (taille, nombre d'enfants). Mais elle contient aussi quelques caractéristiques économiques (revenu global, niveau de vie). En l'état, ces calculs de niveaux de vie ne prennent en compte que les salaires, les retraites et une version très sommaire des prestations chômage. La modélisation des autres composantes du revenu a été laissée à des développements ultérieurs.

Pour ces fonctions, le premier argument d'appel est en général l'indice de l'individu du ménage auquel on s'intéresse, suivi éventuellement d'un certain nombre de paramètres optionnels. Par exemple `TailMen($i)` donnera la taille du ménage de l'individu `$i`, `NbEnf($i,0,14)` donnera le nombre d'enfants de l'individu `$i`, âgés de 0 à 14 ans.

La liste détaillée des fonctions disponibles est la suivante :

Fonctions démographiques

`Acharg`	: indique si l'individu est un enfant à charge
`PersRef`	: retourne la personne de référence du ménage d'appartenance
`EcartAge`	: calcule l'écart d'âge entre l'individu et son conjoint
`NbEnf`	: nombre d'enfants
`NbEnfC`	: nombre d'enfants à charge
`TailMen`	: taille du ménage
`ListEnf`	: liste des identifiants des enfants
`ListEnfC`	: liste des identifiants des enfants à charge
`ListMen`	: liste des identifiants des membres du ménage

Fonctions de calcul du niveau de vie

`NbUC`	: Nombre d'unités de consommation
`RevMen`	: Revenu global du ménage
`NVMen`	: Niveau de vie du ménage

VI.3.1 Fonctions de calcul démographique

Fonction ACharg

Cette fonction indique si l'individu passé en référence est un enfant à charge. Elle retourne un booleen.

Exemple d'appel :

```
if (ACharg($i)) {...}
```

Fonction PersRef

La fonction `PersRef` retourne l'identifiant de la personne de référence du ménage auquel appartient l'individu `$i`.

Nous avons adopté les conventions habituelles des enquêtes auprès des ménages pour déterminer la personne de référence. Si l'individu est un enfant à charge, la personne de référence de son ménage est son père s'il vit avec lui, sinon sa mère. Si la personne est un adulte, si elle ne vit pas en couple, elle est elle-même la personne de référence, si elle a un conjoint et si c'est une femme son conjoint est la personne de référence.

Exemples d'appel :

```
$pr[$i]=PersRef($i);
```

```
if (PersRef($i) eq $i) { } # teste si $i est la personne
                           # de référence de son ménage
```

Fonction EcartAge

Cette fonction calcule l'écart d'âge entre deux individus, par exemple l'écart d'âge entre conjoints, ou l'âge d'un individu à la naissance d'un de ses enfants.

Exemples d'appel :

```
$ecartcj=EcartAge($i,$conjoint[$i]);
$age_premier_enfant=EcartAge($i,$enf[$i][1]);
```

Fonction NbEnf

La fonction `NbEnf` calcule le nombre d'enfants. Elle comprend deux arguments. Le premier argument est l'indice de l'individu. Le deuxième permet de sélectionner les enfants pris en compte dans le calcul. Par défaut, il s'agit de l'ensemble des enfants encore en vie. Mais le second argument peut servir à spécifier une tranche d'âge, ou prendre la modalité `"tous"` qui permet d'inclure aussi les enfants décédés (calcul d'une descendance complète).

Exemples d'appel :

```
$nenf[$i]=NbEnf($i,[4..19]);
$nenf[$i]=NbEnf($i,"Tous");
```

Fonction NbEnfC

La fonction `NbEnfC` fonctionne de la même façon que `NbEnf`, mais elle se limite aux enfants à charge. De ce fait, l'option `"tous"` n'est pas prévue.

Exemple d'appel :

```
$nenf[$i]=NbEnfC($i,[4..19]);
```

Fonction TailMen

Taille du ménage auquel appartient l'individu `$i`.

La taille du ménage est égale au nombre d'enfants à charge de la personne de référence auquel on ajoute 1 pour la personne de référence elle-même et encore 1 si la personne de référence a un conjoint (si l'identifiant du conjoint de la personne de référence est positif).

Exemple d'appel :

```
$taille=TailMen($i);
```

Fonction ListEnf

La fonction `ListEnf` suit le même principe que la fonction `NbEnf`, mais crée une liste contenant les identifiants des enfants. Par défaut, il s'agit de la liste des enfants encore en vie. Le second argument est soit une tranche d'âge, soit l'argument `"tous"` qui permet d'inclure aussi les enfants décédés.

Exemples d'appel :

```
@liste=ListEnf($i,[4..19]);
@liste=ListEnf($i,"Tous");
```

Fonction ListEnfC

La fonction `ListEnfC` est identique à la fonction `ListEnf` mais elle se limite aux enfants à charge. De ce fait, comme pour `NbEnfC`, l'option `"tous"` n'est pas prévue.

Cette fonction sert entre autres à obtenir la liste des individus d'un ménage (fonction `ListMen` qui ne retient que les enfants à charge) qui permet de calculer la somme des revenus des membres du ménage (fonction `RevMen`).

Exemple d'appel :

```
@liste=ListEnfC($i,[4..19]);
```

Fonction ListMen

La fonction `ListMen` retourne la liste des membres du ménage de l'individu `$i`.

Elle ajoute à la liste l'individu, son conjoint s'il existe (identifiant positif) et la liste des enfants, grâce à la fonction `ListEnfC`.

Exemple d'appel :

```
@liste=ListMen($i);
```

VI.3.2 Fonctions relatives au niveau de vie

Fonction NbUC

La fonction `NbUC` calcule le nombre d'unités de consommation du ménage de l'individu dont le numéro est passé en argument, d'après l'échelle d'équivalence OCDE modifiée. Elle affecte donc un poids de 1 à la personne de référence, un poids de 0,5 à son conjoint éventuel et à ses enfants de 14 ans et plus et un poids de 0,3 aux enfants de moins de 14 ans.

Fonction RevMen

La fonction `RevMen` calcule le revenu global du ménage de l'individu `$i`. Elle fait la somme des revenus de tous les membres du ménage retenus dans `ListMen`.

Dans cette fonction les revenus pris en compte sont les salaires, les allocations chômage, les pensions de retraite directes et de réversion et le minimum vieillesse.

Exemple d'appel :

```
RevMen($i)
```

Fonction NVMen

La fonction `NVMen` calcule le niveau de vie du ménage, en divisant le revenu du ménage (`RevMen($i)`) par le nombre d'unités de consommation (`NbUC($i)`).

VI.4 La bibliothèque OutilsBase

La bibliothèque `OutilsBase` propose un ensemble de programmes très généraux destinés à la microsimulation et à la manipulation de données individuelles des types présentés aux sections précédentes. Elle contient également quelques fonctions arithmétiques de base. La liste résumée de ces fonctions est donnée ci-dessous, et les détails de chaque fonction sont donnés dans les sous-sections suivantes.

Initialisations

`UseVRef` : Définition d'une variable de référence

Opérations élémentaires

`In` : test d'appartenance à un ensemble de taille quelconque
`Out` : test de non-appartenance
`Min` : minimum d'un ensemble quelconque de valeurs
`Max` : maximum d'un ensemble quelconque de valeurs
`IMin` : indice du premier élément non vide d'un tableau
`IMax` : indice du dernier élément d'un tableau
`Arr` : arrondi d'un scalaire à un nombre donné de décimales
`ArrAlea` : arrondi aléatoire
`Pointage` : affichage de l'heure

Valeurs de fonctions-type

`Esc` : valeur en x d'une fonction en escalier
`Affn` : valeur en x d'une fonction affine par morceaux
`Sigm` : valeur en x d'une fonction sigmoïde
`Part` : part de x comprise entre deux seuils
`MinMax` : plancher si x<plancher, plafond si x>plafond, x entre les deux
`Redr` : transformation monotone d'une variable x sur support [0,1]

Tirages de variables aléatoires

`Alea` : tirage pseudo-aléatoire dans loi [0,1]
`Booleen` : tirage booleen
`MultiNom` : tirage selon une loi multinomiale
`Logist` : tirage selon loi logistique
`Pareto` : tirage selon une loi de Pareto
`LogLogist` : tirage selon une loi log-logistique (définie par deux quelconques de ses quantiles)

Création et manipulation de variables individuelles et de listes d'identifiants

`Sel` : création liste d'identifiants selon critère déterministe
`Tirage` : création liste d'identifiants à partir d'une proba d'inclusion
`Trunc` : troncation d'une liste à ses n premiers éléments
`Excl` : exclusion d'éléments d'une liste
`EltAlea` : tirage d'un élément au hasard dans une liste
`TriAsc` : tri ascendant d'une liste selon valeurs d'une variable
`TriDesc` : tri descendant d'une liste selon valeurs d'une variable
`TriAlea` : permutation aléatoire d'une liste
`TriSect` : permutation d'une liste par section

VI.4.1 Initialisations

Fonction UseVRef

On commence par une fonction que l'utilisateur n'a, en général, pas besoin de manipuler, mais qu'il est utile de connaître pour bien comprendre le fonctionnement d'un certain nombre de fonctions, notamment les fonctions de tabulation. Cette procédure sert à définir une variable de référence parmi l'ensemble des variables micro de Destinie. La variable de référence a deux utilités :

- Elle sert à connaître l'intervalle d'indices individuels pertinents. À un instant donné, l'indice maximal pour lequel des données micro sont renseignées correspond au dernier indice actif de cette variable de référence et les variables de tabulation se serviront par défaut de cette borne. C'est aussi sur cette base qu'est construite la fonction `IMax` dont on a vu l'utilisation dans la boucle individuelle du programme principal.

- Par ailleurs, elle sert à filtrer les identifiants effectivement pertinents. Le besoin d'un tel filtrage vient de ce que Destinie ne détruit pas les données de l'individu `$i` une fois qu'il est décédé car ces données peuvent servir à différents niveaux : pour des calculs de réversion, pour des reconstitutions de réseaux de parenté etc… Or il serait évidemment problématique que les procédures de tabulation mélangent des données sur des individus vivants et décédés. D'où la nécessité du filtrage.

Pour ce faire, la procédure `UseVRef` a deux arguments, l'adresse de la variable de référence et la valeur seuil de cette variable en dessous de laquelle les individus sont considérés comme non pertinents.

Par défaut, c'est le programme `DefVarRetr` qui se charge de cette initialisation. Il comprend une instruction :

```
UseVRef(@present,0);
```

qui signifie que les comptages s'effectueront en balayant les positions occupées par la variable `@present` en ignorant tous les individus `$i` pour qui `$present[$i]` sera inférieur ou égal à zéro.

Par exemple :

```
$POP60etplus=Count(sub{$age[$_]>60});
```

calculera la population en vie et d'âge supérieur ou égal à 60.

Très exceptionnellement, l'utilisateur peut vouloir changer ces valeurs par défaut. Par exemple, la séquence :

```
UseVRef(@present,-1);
$TOTAL_NES_APRES_1920 = Count(sub{$anaiss[$_]>=1920});
```

calculera le nombre d'individus qui sont ou ont été traités par le modèle et qui sont nés depuis 1920, qu'ils soient encore en vie ou non à la date du comptage.

Il existe d'ailleurs un cas où cette désactivation du filtrage par la variable `$present` est prévue d'office par Destinie qui est le cas de tabulations par génération. Celles-ci interviennent nécessairement en toute fin de simulation et doivent incorporer l'ensemble des individus, qu'ils soient encore en vie ou non. Les fonctions qui effectuent ces tabulations sont présentées plus loin en section IV.6.4.

VI.4.2 Opérations élémentaires

Fonction In

La fonction `In` teste l'appartenance d'un scalaire à un ensemble. Ce scalaire est le premier argument. L'ensemble auquel on teste l'appartenance est fourni à la suite.

Exemple d'appel :

```
if (In("toto",("titi","tata","toto",1))) {...};
if (In($statut[$i],@CodesAct) {...} ;
```

(NB : dans le premier cas, la deuxième paire de parenthèses n'est mise que pour la lisibilité, elle est facultative).

Fonction Out

La fonction `Out` teste si le premier argument est absent de la liste des arguments suivants.

Exemple d'appel :

```
if (Out("toto",("titi","tata","toto",1))) {...};
```

Fonction Min

La fonction `Min` calcule le minimum d'un nombre quelconque de scalaires (et/ou d'un tableau passé par valeurs).

Exemple d'appel :

```
$x = Min (2,$y,$z);
```

Fonction Max

La fonction Max calcule le max d'un nombre quelconque de scalaires (et/ou d'un tableau passé par valeurs).

Exemple d'appel :

```
$x = Max (2,$y,$z, @w);
```

Fonction IMin

La fonction `Imin` retourne l'indice du premier élément non vide d'un tableau monodimensionnel quelconque

Exemple d'appel :

```
$debutSeriePIB = IMin(@seriePib);
```

Fonction IMax

La fonction `Imax` retourne l'indice max d'une variable. S'il n'y a pas d'argument, on utilise la variable de référence définie par `UseVRef` (*cf. supra*).

Exemple d'appel :

```
$n=IMax(@var);
```

Fonction Arr

La fonction `Arr` arrondi un scalaire à `$ndec` decimales (0 par défaut). Le premier argument correspond au scalaire et le deuxième au nombre de décimales.

Exemple d'appel :

```
print Arr(2.3456,2);
```

imprimera 2.35

Fonction ArrAlea

Arrondit un scalaire a ses valeurs entières encadrantes de manière aléatoire.

Exemple d'appel :

```
$x=ArrAlea(2.2);
```

retournera la valeur 2 avec une probabilité 0.8 et la valeur 3 avec une probabilité de 0.2.

Fonction Pointage

La fonction Pointage affiche l'heure courante avec le texte de légende passé en premier argument, soit à l'écran (par défaut) soit dans le fichier dont le nom logique est passé en deuxième argument (optionnel)

Exemples d'appels :

```
Pointage("Debut de la simulation année ".$t);
Pointage("Debut de la simulation année ".$t,LOG);
```

VI.4.3 Valeurs de fonctions élémentaires

Fonction Esc

La fonction `Esc` calcule la valeur en un point donné d'une fonction en escalier. On donne le point où on souhaite l'évaluer et la séquence valeurs/points de discontinuité (avec une valeur de plus que de points de discontinuité).

Exemple d'appel :

```
$x= Esc($g,0,1933,3,1945,2,1970,1.5);
```

donne 0 si $g<1933, 3 s'il est compris entre 1933 et 1944 inclus...

Fonction Affn

La fonction `Affn` calcule la valeur en un point donné d'une fonction affine par morceaux. On donne le point où on souhaite l'évaluer et un nombre arbitraire de paires `$x[$_]`, `$y[$_]` (avec `$x` croissant) correspondant aux points anguleux. La fonction est supposée constante avant et après le dernier point fourni.

Exemple d'appel :

```
$dureecib = Affn($g,1933,150,1943,160);
```

donne 150 si $g<1933, 160 si $g>1943 et interpole linéairement dans l'intervalle.

```
$dureecib = Affn($g,1933,150,1943,160,1948,160,1952,164);
```

idem avec trois segments.

Fonction Sigm

La fonction `Sigm` retourne la valeur en `$x` d'une fonction sigmoïde variant de `$y0` à `$y1` entre `$x0` et `$x1` et constante de part et d'autre de ces bornes.

Exemple d'appel :

```
Sigm($x,2,0,3,1);
```

Fonction Part

La fonction Part calcule la part d'une grandeur scalaire `$x` comprise entre deux seuils `$s1` et `$s2`, c'est-à-dire 0 si `$x<$x1`, `$x-$s1` si `$x` entre `$s1` et `$s2` et `$s2-$s1` si `$x>$s2`.

Exemple d'appel :

```
$SalTR2 = Part($salaire,$PlafondSS,3*$PlafondSS);
```

Fonction MinMax

La fonction `MinMax` retourne `$x`, si `$x` est compris entre `$s1` et `$s2`, `$s1` si `$x<$s1` et `$s2` si `$x>$s2`. Cette fonction est utile pour calculer des grandeurs ayant à la fois une valeur plancher et une valeur plafond

Exemple d'appel :

```
$prestation = MinMax($prestation,$plancher,$plafond);
```

Fonction Redr

La fonction `Redr` redresse une proba scalaire *p* d'un facteur *k* selon la formule :

$t(p)=kp/(1+(k-1)p)$.

Cette formule laisse à 1 ou 0 les probas égales à ces valeurs. Pour $0<p<1$ elle préserve les rapports des odds-ratio, autrement dit, si on a les probas *p* et *p'* pour deux individus, alors on a :

$[t(p)/(1-t(p))]/[t(p')/(1-t(p'))] = [p/(1-p)]/[p'/(1-p')]$

Cette fonction sert à la fonction `Tirage` (voir plus bas).

Exemple d'appel :

```
$proba[$_]=Redr($proba[$_],$coeff);
```

VI.4.4 Tirages de variables aléatoires

Fonction Alea

La fonction `Alea` effectue un tirage selon une loi uniforme [0,1]. Elle utilise une racine `$seed` qui est une variable de la bibliothèque visible depuis le programme appelant dont on peut éventuellement modifier l'initialisation en début de programme.

Exemple d'appel :

```
$seed = 12345689; # Reinitialisation de $seed
for $i (1..IMax)
{
   $x[$i]=Alea;
}
```

Fonction Booleen

La fonction `Booleen` effectue un tirage d'un évènement selon une probabilité donnée.

Exemple d'appel :

```
if (&Booleen($qmor{$sexe[$i]}[$age[$i]]) {$statut[$i]=0};
```

NB : elle fonctionne aussi avec des valeurs hors de l'intervalle [0,1] : le retour est vrai si l'argument est supérieur à un, faux si l'argument est inférieur à zéro.

Fonction Multinom

Cette fonction retourne une variable valant 1, 2.. ou n, avec les n probabilités passées en argument. Si la somme des probabilités est inférieure à un, les cas restants sont mis à 99.

Fonction Logist

Cette fonction effectue un tirage centré et réduit selon une loi logistique de moyenne et écart-type donnés.

Fonction Pareto

La fonction Pareto effectue un tirage selon loi de Pareto en fonction de son seuil et de son paramètre de forme.

Exemple d'appel :

```
$x=Pareto(1000,2);
```

Fonction LogLogist

Cette fonction effectue le tirage d'une variable aléatoire selon une loi LogLogistique, i.e. de fonction de répartition $z^a/(b+z^a)$ paramétrée par deux quelconques de ses quantiles. Par exemple

```
$x=LogLogist(.25,2,.5,3)
```

tirera une VA dont le premier quartile est à 2 et la médiane est à 3.

VI.4.5 Création et gestion de listes d'identifiants

Les listes d'indices sont des listes d'entiers servant à repérer des sous-populations. Pour ces listes, l'ensemble des indices sont supposés actifs, y compris l'indice zéro, ce qui permet d'utiliser ces listes dans des instructions shift, par exemple.

Exemple : si on a une population de 10 individus telle que

```
@statut[1..10]=(1,3,4,2,5,2,3,4,5,4);
```

La liste `@liste=(3,8,10)` correspondra à la liste des numéros des individus tels que `$statut[$i]==4`.

Les opérations disponibles pour les listes sont la création, de manière déterministe ou aléatoire, la troncation, ou des opérations de tri.

Fonction Sel

Cette fonction construit une sous liste de l'ensemble d'indices d'une variable selon un critère déterministe sur cette variable et/ou d'autres variables. Comme pour les fonctions de comptage la condition est fournie sous forme d'une instruction Perl passée entre accolades après le mot clé `sub`.

Exemples d'appels :

```
@liste = Sel (sub {($age[$_]>15) && ($statut[$_]==1)});
@liste = Sel (sub {($age[$_]>15) && ($_<100)});
```

Fonction Tirage

Cette fonction effectue le tirage d'une liste d'individus selon une variable `@proba` donnant la probabilité de sélection. Le second paramètre, optionnel, donne le nombre d'individus que l'on veut tirer (en espérance). Si ces deux paramètres sont incohérents, on applique à `@proba` une déformation conservant les risques relatifs et permettant d'atteindre, en moyenne, la cible d'effectifs recherchée. Si aucune cible n'est donnée, les probas sont utilisées telles quelles, sans redressement.

Exemple d'appel :

```
@liste=Tirage(@proba,200,*LOG);
```

Cas particuliers :

- S'il y a des individus avec `$proba[$i]=1` en nombre supérieur à la cible, la liste sera celle de ces individus, et la cible sera donc dépassée (mais on peut ensuite réduire la liste par la procédure `Trunc`)
- Si la cible est supérieure au nombre total d'individus pour qui `$proba[$i]>0`, l'ensemble de ces probabilités positives sont portées à 1, mais la contrainte reste non saturée.

Le dernier paramètre, optionnel, est l'adresse d'un fichier servant à garder trace du déroulement des itérations, lorsque ajustement il y a.

Fonction Trunc

Cette fonction effectue une troncation d'une liste à un effectif total donné (si l'effectif initial est supérieur à cette cible). Si on veut que les individus éliminés le soient au hasard, il faut prévoir un appel préalable de `TriAlea`.

Exemple d'appel :

```
Trunc(@liste,100);
```

Fonction Excl

Cette fonction exclut d'une liste, avec une probabilité donnée, les individus remplissant une condition. Si la probabilité n'est pas déclarée, elle est supposée égale à un et l'exclusion est donc déterministe.

Exemple d'appel :

```
Excl(@liste,sub {Out($statut[$_],@CodesAct)},0.8);
```

exclura de `@liste_ind` les individus actifs avec une probabilité de 0.8.

Fonction EltAlea

Cette fonction renvoie un élément au hasard dans une liste.

Exemple d'appel :

```
$numero=EltAlea(@liste);
```

Fonction TriAsc

Cette fonction trie une liste d'identifiants selon les valeurs croissantes d'une variable associée à ces identifiants.

Exemples d'appels :

```
@salaire[1..5]=(4,5,2,7,9);
@liste=(1,3,4);
TriAsc(@liste,@salaire);
```

change la liste d'identifiants en (3, 1, 4) (pour qui les salaires respectifs sont de 2, 4 et 7)

Fonction TriDesc

Cette fonction trie une liste d'identifiants selon les valeurs décroissantes d'une variable associée à ces identifiants, selon le même principe que `TriAsc`.

Exemple d'appel :

```
TriDesc(@liste,@salaire);
```

Fonction TriAlea

La fonction `TriAlea` trie une liste selon un ordre aléatoire.

Exemple d'appel :

```
TriAlea(@liste);
```

Fonction TriSect

Cette fonction effectue une permutation d'une liste par section : on tire un élément au hasard, et on replace en début de liste cet élément et tous les suivants. Cette fonction permet de parcourir une liste à partir d'une position aléatoire d'une manière plus rapide que par `TriAlea`.

Exemple d'appel :

```
TriSect(@liste);
```

VI.5 Fonctionnalités supplémentaires de la bibliothèque OutilsRetr

L'essentiel de cette bibliothèque a été vu en parties IV et V. Les programmes sont les suivants :

Programmes précisant les hypothèses de simulation des retraites

`UseBios`	: définition des fichiers dans lesquels sont récupérées les biographies
`UseOpt`	: choix des options générales de simulation
`UseOptCN`	: choix des options comptes notionnels, s'il y a lieu
`UseLeg`	: choix de la législation en projection
`UseLegRetroMax`	: choix de la législation pour les reconstitutions initiales
`UseConv`	: choix des coefficients de conversion si comptes notionnels

Programme de récupération des résultats de l'étape Bios

`Relec`	: relecture des données initiales ou projetées

Programmes d'initialisation ou de projection des pensions

`SimDir`	: initialisation ou projection des droits directs
`SimDer`	: initialisation ou projection des droits dérivés
`SimMinVieil`	: initialisation ou projection du minimum vieillesse

Programmes d'affichage des données individuelles (surtout utilisés par l'explorateur)

`ShowBio`	: affichage de la carrière complète
`ShowPens`	: affichage des différents éléments de la pension pour un départ à âge donné

Les deux seules fonctions qui n'ont pas été présentées dans la partie précédente sont les fonctions relatives à la simulation de la retraite en comptes notionnels. Elles sont donc présentées ci-après, à la suite de quoi on donnera quelques éléments complémentaires sur le fonctionnement interne de la fonction principale `SimDir`, et notamment la façon dont elles s'appuient sur les autres fonctions de `OutilsDroits` et `OutilsComp` vues dans les sections VI.1 et VI.2.

VI.5.1 Simulation de systèmes en comptes notionnels

Fonction UseOptCN

Cette fonction spécifie les options supplémentaires si l'option `"CN"` a été demandée avec `UseOpt`. Il y a deux arguments. Le premier est le millésime de démarrage du basculement. Le second est une chaine de caractère, avec les mêmes principes que dans `UseOpt`, pouvant contenir les éléments suivants :

Choix de l'étendue du système

`"rg"`	: basculement pour le régime général et assimilés uniquement
`"base"`	: basculement pour RG, assimilés et pour la part des salaires FP sous plafond. Dans ce cas, un régime chapeau est créé pour garantir le maintien des pensions FP
`"etendu"`	: basculement pour RG, assimilés et ensemble de la FP
`"global"`	: basculement général, y compris ARRCO/AGIRC.

Choix du calendrier de basculement

`"immediat"`	: dès l'année de lancement, toutes les pensions sont calculées avec le nouveau système

"progressif" : le nouveau système ne s'applique qu'aux années cotisées à partir de l'année de lancement

NB : le décodage des options se fait sur les 2 premiers caractères, ce qui autorise des appels abrégés.
Exemples d'appel :

```
UseOptCN(2010,"Immediat/rg");
UseOptCN(2010,"glob prog");
```

Fonction UseConv

Cette fonction met à jour les coefficients de conversion si on simule un régime en comptes notionnels. Il y a deux modes d'appel. Le premier mode doit être utilisé lorsqu'on veut des coefficients de conversion basés sur l'espérance de vie courante. On écrira par exemple :

```
UseConv(60,70,"EV",$t);
```

qui calcule les coefficients de conversion entre `$agemin` et `$agemax` égaux à 60 et 70 ans sur la base de la mortalité de la date `$t`.

Le second mode consiste à simplement passer les valeurs des coefficients, soit :

```
UseConv(60,65,0.05,0.055,0.060,0.065,0.070,0.075);
```

Dans les deux cas, les coefficients sont mis à zéro avant `$agemin`, et maintenus à leur valeur de `$agemax` au-delà de `$agemax`

VI.5.2 Compléments sur la fonction SimDir

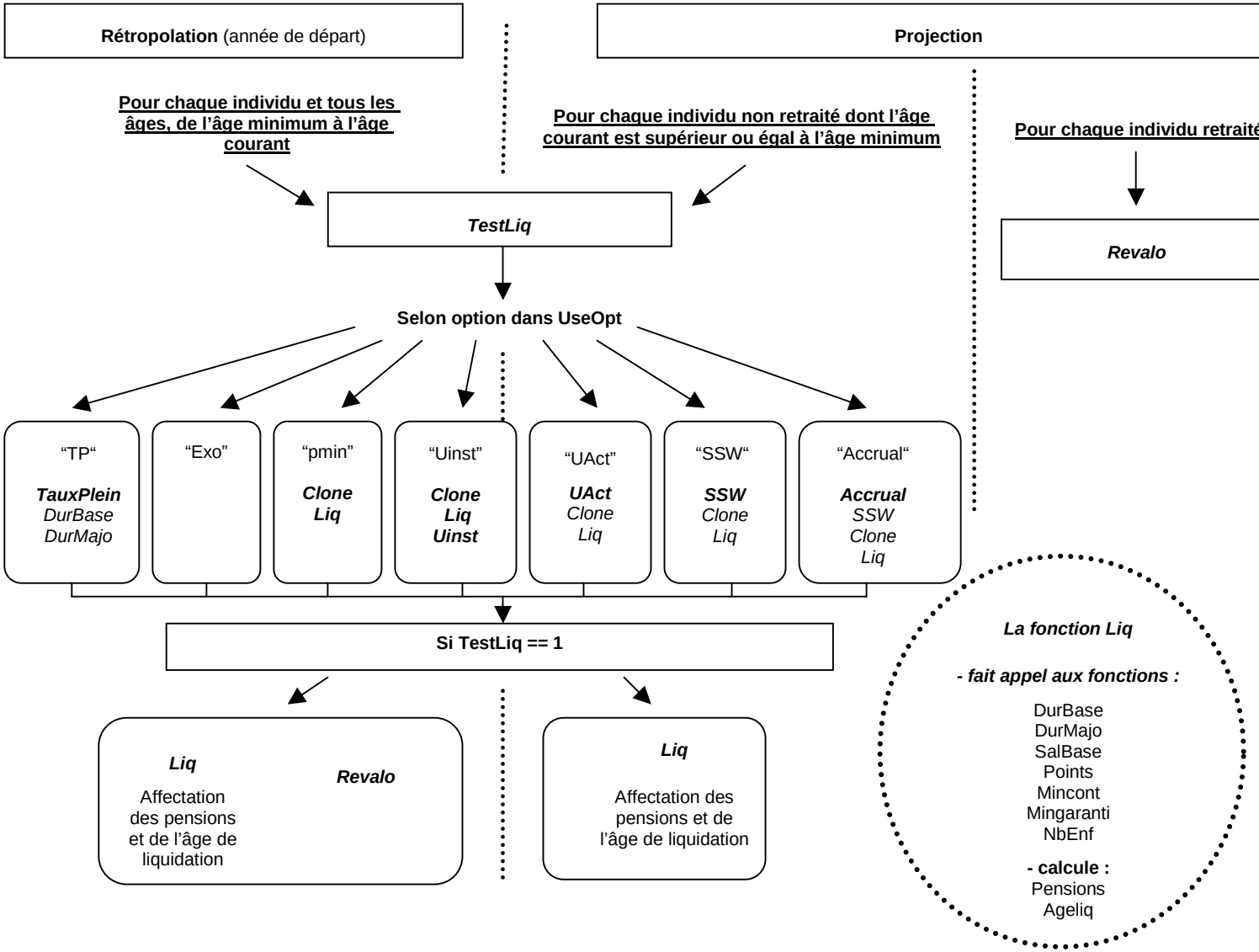
Nous avons vu dans la partie précédente que le lancement de la simulation des pensions et des départs à la retraite s'effectue par le lancement dans le programme principal de la fonction `SimDir`. En utilisation normale, le fonctionnement interne de cette fonction n'a pas besoin d'être connu. Mais il peut être utile de connaître ce fonctionnement interne et, ayant présenté les contenus des bibliothèques `OutilsDroits` et `OutilsComp`, on peut maintenant expliquer l'emboîtement complet de l'ensemble des fonctions qui interviennent dans la simulation des droits à retraite.

`SimDir` effectue plusieurs opérations, également illustrées par la figure 8 :

- Elle teste à quel âge l'individu a liquidé -en phase de rétropolation- ou s'il liquide à l'âge courant -en phase de projection.
- Si liquidation il y a, elle calcule la pension associée.
- Si la liquidation est antérieure à la date courante, elle revalorise la pension pour cette année courante, soit depuis la date de liquidation -en phase de rétropolation-, soit entre l'année précédente et l'année courante, en phase de projection.

La première étape fait appel à la fonction `TestLiq` de `OutilsComp`. Dans les options `TP` et `Exo`, le calcul est assez direct et débouche immédiatement sur un appel à la fonction `DroitsDir` dans l'hypothèse où le départ a effectivement lieu à l'âge considéré. Dans l'option `UInst`, il suffit de comparer les niveaux d'utilité instantanés avec ou sans départ. Ceci suppose déjà de procéder à une liquidation fictive sur un clone de l'individu considéré. C'est sur ce clone que se fait la comparaison des niveaux d'utilité pour ne pas modifier les caractéristiques de l'individu `$i` avant d'être sûr qu'il liquide effectivement. Le même système est utilisé avec les options `Uact`, `SSW` et `Accrual` qui supposent de comparer des niveaux d'utilité ou des cumuls de droits sous différentes hypothèses d'âge de départ du même individu. Les appels à `SimDir` et `TestLiq` avec ces options conduisent donc à cumuler le recours aux fonctions `TestClone`, `DroitsDir` et, selon le cas `Uinst`, `SSW` et `Accrual`.

Figure 8 : programmes appelés directement ou indirectement par la fonction SimDir, selon phase et options de la simulation
(appels directs en gras, appels indirects en caractères normaux)



Une fois que la fonction `TestLiq` a retourné une valeur positive le reste du déroulement du programme est beaucoup plus linéaire. Il se limite à mettre en œuvre une dernière fois la fonction `DroitsDir`, cette fois-ci sur l'individu `$i` lui-même plutôt que son clone, puis à mobiliser la fonction `Revalo` pour toutes les années ultérieures.

VI.6 Fonctionnalités supplémentaires de la bibliothèque OutilsTab

Une partie substantielle du contenu de cette bibliothèque a été examinée *supra* à la section IV.4. On redonne ci-dessous l'ensemble de son contenu et on détaille le fonctionnement des procédures non présentées à la section IV.4.

Initialisations

`UseW` : définit poids pour calculs des effectifs et des masses

Tabulations générales simples

`SVar` : somme des valeurs d'une variable (avec possibilité de filtrage)
`MVar` : moyenne des valeurs d'une variable (avec possibilité de filtrage)
`QVar` : quantiles (au choix) d'une variable (avec possibilité de filtrage)
`EVar` : écart-type d'une variable (avec possibilité de filtrage)
`Count` : comptage nombre d'identifiants vérifiant condition de filtrage
`Taux` : calcul d'un taux
`Ratio` : calcul d'un ratio
`Distrib` : distribution d'une variable discrète

Raccourcis pour tabulations spécifiques

`SSal` : Somme des salaires
`MSal` : Salaire moyen
`SPens` : Somme des pensions
`MPens` : Pension moyenne

Tabulations par âge

`UseGrAge` : définit les classes d'âge
`GrAge` : calcule le groupe d'âge auquel appartient l'individu `$i`
`MAge` : moyenne par âge des valeurs d'une variable (avec possibilité de filtrage)
`CountAge` : comptage nombre d'identifiants par âge vérifiant condition de filtrage
`TauxAge` : calcul d'un taux par âge

Tabulations par génération

`UseGrGen` : définit les classes de générations
`GrGen` : calcule le groupe de génération auquel appartient l'individu `$i`
`MGen` : moyenne par génération des valeurs d'une variable (avec possibilité de filtrage)
`CountGen` : comptage nombre d'identifiants par génération vérifiant condition de filtrage
`TauxGen` : calcul d'un taux par génération

Autres manipulations de tableaux

`Coupe` : extraction d'une partie d'un tableau à double indice (y.c. coupes diagonales)
`STab` : somme des valeurs des cases d'un tableau à une dimension, ou de produits de cases de deux tableaux ou plus
`MTab` : moyenne des valeurs
`SAct` : idem `STab` avec actualisation
`TRI` : calcul d'un taux de rendement interne associé à un tableau simple

VI.6.1 Initialisation

Fonction UseW

Cette fonction définit les pondérations à utiliser pour les fonctions de comptage et de somme. Permet à la fois de redresser la taille de l'échantillon simulé et de contrôler l'unité d'affichage des résultats de ces deux fonctions (il s'agit d'un poids uniforme, il ne sert donc pas à repondérer les individus de la simulation les uns par rapport aux autres). Si aucun argument n'est passé, on utilise ou réactive la pondération par défaut qui permet d'obtenir des effectifs en milliers et des masses financières en milliards d'euros. Sinon, on utilise les valeurs passées en argument.

Exemple d'appel :

```
UseW(0.924,0.924/1000000);
```

VI.6.2 Tabulations simples : la fonction Distrib

La seule fonction de tabulation simple de la liste ci-dessus qui n'ait pas déjà été présentée à la section III.3 est la fonction `Distrib`. Cette fonction retourne un hachage donnant la distribution des valeurs d'une variable discrète (numérique ou non numérique).

Exemples d'appel :

```
%DistAgeLiqH = Distrib(@ageliq,
                sub{($ageliq[$_]>0) && ($sexe[$_]==$CodeHom)});
```

Cette demande retournera les pourcentages de liquidants hommes par âge de liquidation. Par exemple `$DistAgeLiqH{62}` sera le pourcentage de liquidants à 62 ans au sein de cette population.

L'instruction précédente peut être réécrite sous une forme générant un hachage pour chaque sexe. On écrira :

```
for $s ($CodeHom,$CodeFem)
{
   %{$DistAgeLiq{$Label{sexe,$s}}} =
        Distrib(@ageliq,sub{($ageliq[$_]>0) && ($sexe[$_]==$s)})
};
```

à la suite de quoi le double hachage `%DistAgeLiq` pourra être sauvegardé par l'instruction

```
SavHH($resultats,agesliq,ages de liquidation,%DistAgeLiq) ;
```

VI.6.3 Construire des séries de résultats macro par âge

Trois fonctions permettent d'effectuer des calculs par âge : `MAge`, `CountAge` et `TauxAge`. Elles supposent d'avoir préalablement défini des groupes d'âge à l'aide de la fonction `UseGrAge`. S'il n'y a eu aucun appel préalable à cette fonction, la tabulation se fera par défaut par âge simple. Le résultat se présente sous forme hachage même quand les classes d'âge sont des classes d'âges simples. La procédure auxiliaire `GrAge` calcule à quel groupe d'âge appartient un individu donné.

On présente succinctement ces différentes procédures puis un exemple de leur utilisation.

Procédure UseGrAge

Cette fonction définit les tranches d'âge pour les fonctions qui suivent. Le premier argument est l'âge de départ, le deuxième est l'âge de fin, le troisième est le pas des tranches d'âge. Si le pas est supérieur à un, cette fonction crée aussi les labels décrivant les tranches d'âge

Exemple d'appel :

```
UseGrAge (20,49,10);
```

crée les groupes compris entre 20 et 29 ans, 30 et 39 ans, ainsi que 40 et 49 ans, avec les labels associés.

Procédure GrAge

Calcule le groupe d'âge auquel appartient un individu donné. Le seul argument d'appel est l'individu `$i`. Si aucun regroupement par âge n'a été spécifié, son résultat est identique à `$age[$i]`.

Fonction MAge

La fonction `MAge` fonctionne sur le même principe que `MVar` mais calcule des moyennes séparées par âge simple et les retourne sous forme de hachage.

Exemple d'appel :

```
%sal_age = MAge(@salaire,sub{$salaire[$_]>0});
```

calculera le salaire moyen par âge parmi les individus qui ont un salaire supérieur à zéro.

Fonction CountAge

La fonction `CountAge` suit le même principe que `MAge`, mais elle effectue un comptage par âge plutôt qu'une moyenne. Le résultat est fourni avec une décimale.

Exemple d'appel

```
%POPACT = CountAge (sub {In($statut[$_],@CodesAct});
```

calculera le nombre d'individus en vie qui occupent un emploi.

Fonction TauxAge

La fonction `TauxAge`, comme `CountAge` calcule des taux mais par âge (avec donc deux conditions, comme pour la fonction `Taux` de `OutilsBase`, la deuxième étant optionnelle).

Exemple d'appel

```
%TAUX_CHO = TauxAge(sub {In($statut[$_],$CodeCho)}
                    sub {In($statut[$_],@CodesAct)});
```

calculera la part des chômeurs parmi les actifs, aux différents âges.

Exemple d'utilisation avancée

En général, ces tabulations par âge se font à date donnée, donc en fin de boucle sur `$t`, après la dernière boucle sur `$i`, comme pour les autres fonctions de tabulation vues jusqu'à présent.

Par exemple, la séquence

```
if ($t==103)
{
   %POP_2003    = CountAge;
   %TXOCC_2003  = TauxAge(sub{In($statut[$_],$CodesOcc)});
   %MOYRG_2003  = MAge(@pension_rg,sub{$pension_rg[$_]>0});
}
```

calcule pour l'année 2003 (`$t==103`) le nombre d'individus vivants par âge simple (`%POP_2003`), la part des actifs occupés pour chaque âge (`%TXOCC_2003`) et la pension au régime général moyenne parmi ceux qui ont une pension au régime général positive (`%MOYRG_2003`).

Si on veut sauvegarder l'ensemble des pyramides des âges à toutes les dates, il faut prévoir un appel de la forme :

```
%{$POP[$t]} = CountAge ;
```

qui créera un tableau à double indice `$POP[$t]{$a}` sauvegardable sous Excel par la procédure `SavSH` qui sera vue plus loin en section VI.7.

VI.6.4 Construire des séries de résultats macro par génération

Pour procéder à des calculs par génération, on procède de manière très similaire à ce qui est fait pour la production de données par âge. Les fonctions `UseGrGen`, `GrGen`, `Mgen`, `CountGen` et `TauxGen`, sont le pendant exact de `UseGrAge`, `GrAge`, `MAge`, `CountAge` et `TauxAge` et leurs syntaxe ne sera donc pas détaillée davantage. La seule différence est que leur appel se fait en principe en toute fin de simulation, après la fin de la boucle sur `$t` et que les tabulations qu'elles produisent incorporent les individus décédés à cette date. Elles n'ont évidemment de sens que sur des variables conservées jusqu'en fin de simulation.

Pour donner un exemple, la séquence suivante en fin de programme produit des âges de liquidation par genre et groupes quinquennaux de générations, entre les générations 1940 et 1984, sur la base de la variable `$ageliq` récupérée en sortie de SimDir.

```
UseGrGen(40,84,5) ;
for $s ($CodeHom,$CodeFem)
{
   @{$AGEGEN{$Label(sexe,$s)}}=MGen(@ageliq,sub{$sexe[$_]=$s});
}
```

après quoi `$AGEGEN{"Homme"}{"50-54"}` donnera l'âge moyen à la liquidation des hommes de la génération 1950-1954, et le double hachage `%AGEGEN` sera sauvegardable par `SavHH`.

VI.6.5 Autres manipulations de tableaux

À côté de ses fonctions de tabulation de données micro, `OutilsTab` propose des fonctions de manipulation de tableaux simples de données par âge, par date ou par génération ou des tableaux à double indice croisant deux de ces dimensions. Contrairement à la convention retenue pour les variables, la position zéro est utilisée (il n'y a pas d'individu n° zéro mais il peut y avoir un âge ou une date zéro).

Fonction Coupe

La fonction `Coupe` extrait des éléments d'un vecteur (données par date, génération ou âge) ou d'un tableau à double entrée (date ou génération x âge) et les empile dans un tableau à une dimension (indicé à partir de zéro)

Le principe général est de passer un nom de vecteur ou de tableau à double entrée, puis des intervalles à sélectionner, passés entre crochets.

On aura par exemple, pour des tableaux simples :

```
@popageactif = Coupe(@pop,[15..65]);
@popdep      = Coupe(@pop,[0..14,65..110]);
@PIB8090     = Coupe(@PIB,[1980..1990]);
```

On note que les millésimes peuvent être passés soit en année absolue (plus lisible), soit comptés à partir de 1900 : le programme transcrit automatiquement. La dernière écriture est donc équivalente à

```
Coupe(@PIB,[80..90]);
```

Pour des tableaux à double entrée, on a le choix entre des expressions de type

```
Coupe(@pop,[1980,[15..65]]);     # Extraction coupe transversale
Coupe(@pop,[[1980..1990],40]);     # Extraction serie temporelle
Coupe(@pop,[[1980..1990],[40..50]]);# Extraction coupe diagonale
```

Cette fonction appelle deux remarques :

- Ce que retournent ces programmes sont de simples empilements des valeurs prélevées dans les tableaux d'origine, indicés à partir de 0 : l'indiçage initial est donc perdu.
- Pour les sélections en coupe diagonale, le programme ne contrôle pas la cohérence des intervalles horizontaux et verticaux : il part du point date x âge minimum et empile les données jusqu'à l'âge maximal, tant qu'elles existent.

Fonction STab

La fonction Stab fait la somme des valeurs sélectionnées dans un tableau ou les produits terme à terme de valeurs extraites de plusieurs tableaux (jusqu'à 5).

Exemples d'appel :

```
$popadu =STab(@pop,[20..110]);
$popadu[$t]=STab(@pop,[$t,[20..110]]);
$popact[$t]=STab(@pop,[$t,[15..65]],@txact,[$t,[15..65]]);
$popact[$t]=STab(@pop,[$t,[15..65]],@txact,[15..65]);
$popemp[$t]=STab(@pop,[$t,[15..110]],@txact,[15..65],
                                      @txemp,[15..65]);
```

On notera que les bornes de sélection pour les différentes coupes sont indépendantes les unes des autres : le programme sommera les produits terme à terme des N premières positions de chaque sélection avec N la longueur de la sélection la plus courte.

Fonction MTab

La fonction Mtab fonctionne de la même manière que STab mais calcule une moyenne. Elle ne prend en compte que les valeurs strictement positives (pour gérer les éventuelles troncations).

Fonction SAct

La fonction Sact est identique à la fonction STab mais le premier paramètre est un taux d'actualisation exprimé en fraction de un).

Fonction TRI

Cette fonction calcule un taux de rendement interne associé à un profil simple par date ou âge. Les paramètres sont le profil temporel et les indices entre lesquels on réalise le calcul

Exemple d'appel

Si `@transfert` est un tableau par âge donnant, pour un individu, le solde instantané de ses transferts vis-à-vis du système, avec par exemple les cotisations comptées négativement et les prestations reçues comptées positivement, l'appel

```
$r = TRI(@transfert,15,80) ;
```

retourne le taux de rendement interne sur les cotisations versées, avec des prestations prises en compte jusqu'à l'âge de 80 ans.

VI.7 Fonctionnalités supplémentaires de la bibliothèque OutilsExcel

La bibliothèque `OutilsExcel` lit ou écrit différents formats de tableaux dans des fichiers Excel. La liste complète des procédures disponibles est la suivante :

Procédure de gestion générale de classeurs Excel

`ClassIn`	: ouvre un classeur en lecture
`ClassOut`	: ouvre un nouveau classeur en écriture
`CloseOut`	: ferme un classeur ouvert en écriture
`ListClass`	: liste des classeurs d'un répertoire et leur contenu
`ListFeuil`	: liste des feuilles d'un classeur ouvert en entrée

Lecture de données

`GetAP`	: lecture d'un tableau par âge et période
`GetSer`	: lecture d'une série temporelle
`GetPro`	: lecture d'un profil par âge
`GetH`	: lecture d'un hachage simple
`GetHH`	: lecture d'un double hachage

Écriture de données

`SavAP`	: sauvegarde d'un tableau par âge et période
`SavPro`	: sauvegarde d'un ensemble de profils par âge
`SavSer`	: sauvegarde d'un ensemble de séries temporelles
`SavHS`	: sauvegarde d'un hachage de séries temporelles
`SavSH`	: sauvegarde d'une série temporelle de hachages
`SavHH`	: sauvegarde d'un hachage de hachages

VI.7.1 Gestion générale de classeurs Excel

Les fonctions `ClassOut` et `CloseOut` ont été décrites dans la présentation des utilisations standard du modèle. Elles ne sont pas détaillées à nouveau ici. On ne présente que les fonctions `ClassIn`, `ListClass` et `ListeFeuil`.

Fonction ClassIn

Cette fonction ouvre un classeur en lecture.

Exemple d'appel

```
$popul= ClassIn("Population.xls");
```

Fonction ListClass

La fonction `ListClass` fait la liste de tous les fichiers .xls du répertoire passé en entrée et sort des informations résumées sur leurs contenus dans un fichier ListeClass.txt (si aucun argument, le répertoire est le répertoire courant)

Exemple d'appel :

```
ListClass("C:\Donnees");
```

Fonction ListFeuil

Cette fonction retourne un tableau contenant les noms de l'ensemble des feuilles du classeur passé en argument. Peut notamment servir à construire un balayage systématique de toutes ces feuilles, si elles sont construites selon le même format (exemple : feuilles contenant les variantes d'une même projection)

Exemple d'appel :

```
@liste = ListFeuil($classeur);
```

VI.7.2 Lecture de données

Fonction GetAP

La fonction `GetAP` lit un tableau de données par âge et période dans une feuille Excel. Elle est utilisée par exemple pour la lecture des données de mortalité du fichier `ParamMortal.xls`.

La présentation de la feuille Excel doit être de la forme

	A	B	C	D	E	F	G
1	Descriptif feuille						
2		0-4	5-9	10	11	12-80	80-99
3	1980	..	..	..	..	..	..
4	1981	..	..	..	..	..	..
5	1982	..	..	..	..	..	..

Dans ce cas, il y a lecture de l'intégralité de la feuille, et les données par âges regroupés (du type nn-mm) sont éclatées en données par âge simple soit par réplication du contenu de la cellule pour chaque âge simple (cas par exemple où la série donne une moyenne ou un taux) soit par équidivision entre ces âges (cas où la série est un total). Dans ce deuxième cas on précise l'option `"div"`. L'appel est, par exemple

```
@poptot=GetAP($classeur,"NomFeuille","Div");
```

Fonction GetSer

La fonction `GetSer` met une série temporelle (en colonne) dans une feuille Excel. La présentation de la feuille Excel doit-être de la forme :

	A	B	C	D	E	F	G
1	Descriptif feuille						
2		Pop	Popac	Pib	Pib/t	Conso	Salai
3	1980	..	..	..	..	..	..
4	1981	..	..	..	..	..	..
5	1982	..	..	..	..	..	..

Elle est entre autres utilisée pour lire les paramètres du fichier `ParamSociaux.xls` par `DefVarRetr`.

Dans ce cas, la fonction lit la série contenue dans la colonne qu'on lui spécifie.

Exemple d'appel :

```
@PIB = GetSer($classeur,"NomFeuil",3);
```

Fonction GetPro

Cette fonction lit un profil par âge (en ligne) dans une feuille Excel. La présentation de la feuille Excel doit être de la forme :

	A	B	C	D	E	F	G
1	Descriptif feuille						
2		0-4	5-9	10	11	12-80	80-99
3	MSalH	..	..	..	..	..	..
4	MSalF	..	..	..	..	..	..

Dans ce cas, la fonction lit la série contenue dans la ligne qu'on lui spécifie, en la désagrégeant par âge simple, avec ou sans l'option "Div", selon le cas. L'appel est donc de la forme :

```
@SalairesFemmes = GetPro($classeur,"NomFeuil",2,"Div");
```

Fonction GetH

La fonction `GetH` lit un hachage (en ligne) dans une feuille Excel.

La présentation de la feuille Excel doit être de la forme :

	A	B	C	D	E	F	G
1	Descriptif feuille						
2		EmpH	EmpF	ChoH	ChoF	InaH	InaF
3	Coef1	..	..	..	..	..	..
4	Coef2	..	..	..	..	..	..

Dans ce cas, la fonction lit la série contenue dans la ligne qu'on lui spécifie.

Exemple d'appel :

```
%COEF1 = GetH($classeur,"NomFeuil",3);
```

Fonction GetHH

La fonction `GetHH` lit un hachage à double dimension dans un fichier Excel

Les arguments d'appel sont la référence du classeur (initialisée par la fonction `ClassIn`) et le nom de la feuille.

Le format de la feuille doit être du type suivant :

	A	B	C	D	E	F	G
1	Descriptif du tableau						
2		Hom	Fem				
3	Ina	21	40				
4	Act	120	100				
5	Cho	30	30				

Exemple d'appel :

```
%matrice =GetHH($classeur,"NomFeuil");
```

VI.7.3 Écriture de données

Les trois fonctions qui n'ont pas été présentées à la section IV.4 sont les fonctions `SavAP`, `SavSH` et `SavHH`.

Fonction SavAP

La fonction SavAP sauvegarde un tableau par âge et période dans une feuille Excel. On obtient un fichier Excel contenant dans chaque colonne un âge et chaque ligne une année.

Exemple d'appel :

```
SavAP($resul,"Population","Population par âge",@POP);
```

	A	B	C	D	E	F	G
1	Descriptif feuille						
2		0	1	2	3	4	5
3	1980	..	..	..	..	..	..
4	1981	..	..	..	..	..	..
5	1982	..	..	..	..	..	..

Dans cet exemple, les résultats sont sauvegardés dans le fichier `$resul` que l'on aura ouvert au préalable, dans une feuille appelée ici `Feuil2.` Le titre de la feuille est `Titre` et le calcul concerne la variable `@POP.`

Pour que cette sauvegarde soit possible, il faut par exemple que la ligne du tableau `@pop` ait été sauvegardée au fur et à mesure de la simulation, chaque année, par une instruction du type :

```
@{$POP[$t]} = CountAge;
```

Procédure SavSH

Cette fonction permet la sauvegarde d'une série temporelle de hachages, par exemple une série de taux par âges regroupés.

Exemple d'appel :

```
SavSH($classeur,"Taux","Taux par ages",@taux);
```

où `@taux` est un tableau comprenant un hachage (de clés identiques) pour chaque date

Procédure SavHH

Sauvegarde d'un hachage de hachages, par exemple un ensemble de taux par âges regroupés. NB : les sous-hachages n'ont pas besoin d'être de longueur identique, le programme gère le cas où la liste de leurs clés est variable.

Exemple d'appel :

```
SavHH($classeur,"Taux","Taux par ages regroupés",%taux);
```

VI.8 Fonctionnalités supplémentaires de la bibliothèque DefVarRetr

La presque totalité du contenu de cette bibliothèque a été examinée *supra* à la section III.3. Il s'agissait de l'ensemble des opérations de déclaration et d'initialisation de variables. Cette bibliothèque comprend par ailleurs deux fonctions de manipulation de données individuelles, utilisées notamment par la bibliothèque `OutilsComp`. Il s'agit des procédures `Clone` et `Delete` permettant d'ajouter des individus à la population simulée ou d'en retrancher. La première peut notamment être utile à la génération de cas-types. On les présente succinctement.

Fonction Clone

Cette fonction crée un individu additionnel à partir d'un individu existant. L'appel par défaut est :

```
Clone($i) ;
```

Cet appel copie l'ensemble des caractéristiques de l'individu `$i` dans l'individu d'identifiant zéro, à l'exclusion de la variable `$present` laissée à zéro. Cette fonction est utilisée par la procédure `TestLiq`. En effet, tester s'il y a liquidation nécessite d'examiner les conséquences pour le revenu de l'individu de départs en retraite à divers âges et il est préférable de tester ces effets sur une copie temporaire (un clone) plutôt que sur l'individu `$i` lui- même. Les données relatives à cet individu `$i` ne seront définitivement modifiées par `SimDir` que si l'individu liquide effectivement sa retraite.

La fonction renseigne également la variable scalaire `$id_clone` qui permet de savoir de qui l'individu 0 est le clone. Le fait de considérer le clone comme non-présent évite qu'il ne soit pris en compte dans les tabulations ou par d'autres procédures ne devant s'appliquer qu'aux individus simulés à part entière.

En général, il est déconseillé d'utiliser la fonction sous cette forme dans un programme appelant, car il risque d'y avoir conflit entre les affectations à l'individu zéro gérées par ce programme principal et celles qui sont gérées en interne par `TestLiq`.

On peut en revanche l'utiliser sous une forme utilisant un ou deux arguments optionnels supplémentaires optionnels. Le premier argument indique un numéro d'identifiant autre que zéro pour le clone. Par exemple :

```
Clone(10,Imax+1);
```

rajoute en fin de population un individu qui est le clone de l'individu 10.

Dans cet appel, il s'agira toujours d'un individu « non-présent ». On peut forcer cette présence par :

```
Clone(10,IMax+1,"present");
```

Avec cette syntaxe il est très facile de constituer une population de cas-types. Si on a défini l'ensemble des caractéristiques d'un individu 1, on peut par exemple avoir une boucle du type :

```
for $i (2..100)
{
    Clone(1,$i,"present");
    for $a (15..70)
    {
        $salaire_[$i][$a]=(0.5+$i/100)*$salaire_[1][$a]
    }
}
```

Cette boucle créera une population artificielle de 100 personnes à carrières parallèles à celle de l'individu 1, à des niveaux de salaire compris entre 0,5 et 1,5 fois celui de cet individu de référence. On pourra ensuite simuler la distribution des droits à retraite sur cette population.

Fonction Delete

Cette fonction a un seul argument qui est l'identifiant de l'individu et elle remet à **`undef`** l'ensemble des informations relatives à cet individu. L'appel est donc :

```
Delete($i) ;
```

Par défaut, `$i` est pris égal à zéro. C'est donc un appel simple sans argument qui sert à `TestLiq` pour désactiver le clone d'identifiant zéro après utilisation.

VII - Utilisations avancées (2) : création de nouveaux modules

Cette dernière section donne quelques indications à l'utilisateur ayant besoin d'enrichir les modules existants ou éventuellement de créer un nouveau module dédié à une thématique particulière.

On va présenter la structure générale d'un module (section VII.1) puis donner quelques exemples précis de sous-programmes Destinie existant pouvant servir de modèle pour de nouveaux modules (section VII.2).

VII.1 Structure générale d'un module

Un module est un fichier `.pm` qui définit un certain nombre d'objets qui peuvent être soit des variables, soit des sous-programmes. Ils sont ensuite utilisables par d'autres modules ou des programmes principaux `.pl`. Un module peut-lui même s'appuyer sur d'autres modules.

La structure-type est la suivante :

```
package NouveauModule;                    # Nom du module
use strict;

use DefVarRetr;                           # Appels autres librairies
use OutilsBase;

our @ISA    = qw (Exporter);              # Gestion des exportations
our @EXPORT = qw (fonction1 fonction2
                  @varglob $varglob);

our (@varglob,$varglob);            # Definition des variables globales
my  (@varloc,$varloc);              # Definition de variables locales

@varglob= … ;                       # Initialisations des variables
$varloc=… ;                         #     globales ou locales

sub fonction1                       # Codes des différentes fonctions
{
…
}

sub fonction2
{
…
}

1;                                  # Fin du module
```

On voit que le module débute par la déclaration de son nom, qui doit être identique au nom du fichier `.pm` qui le contient. On applique ensuite, de préférence, la directive de compilation `use strict` qui interdit d'utilisation de variables non pré-déclarées, puis on définit les bibliothèques auxquelles le module doit avoir accès, exactement comme dans les programmes principaux `.pl` vus jusqu'ici. A priori, un module Destinie a besoin a minima de `DefVarRetr` et `OutilsBase`.

Les deux instructions suivantes gèrent l'exportation des symboles créés par le module vers les unités appelantes. La seconde est celle que l'utilisateur doit compléter en fonction de la liste de variables et sous-programme qu'il veut mettre à disposition. Cette liste est ici passée sous la

forme qw (nom1 nom2…) qui est une notation plus compacte que ("nom1","nom2"…)dans laquelle qw est l'abréviation de « quoted words ».

Parmi ces variables exportées, celles qui sont utilisées dans le corps du module doivent être explicitement redéclarées dans une instruction our, les autres variables locales au module étant pour leur part déclarées par un my.

Le module renseigne ensuite celles de ces variables qui ont besoin d'être initialisées. Dans le cas où le module déclarerait de nouveaux paramètres dépendant du temps, on peut notamment imaginer qu'ils soient lus dans un fichier Excel sur le modèle des paramètres lus par DefVarRetr dans ParamSociaux.xls. Dans ce cas, le package aura dû prévoir un use OutilsExcel et on aura, à cet endroit du package, une séquence d'instructions du type :

```
$param        = ClassIn($RepOutils."ParamNewPack.xls");
@Param11      = GetSer($param,"feuil1", 1);
@Param12      = GetSer($param,"feuil1", 2);
@Param21      = GetSer($param,"feuil2", 1);
```

dans laquelle $RepOutils est une variable Destinie calculée par DefVarRetr donnant le chemin d'accès au répertoire Outils.

Après cela, le package consiste dans la suite des codes-source des différents sous-programmes, introduits par le mot clé sub.

Le texte du module se termine enfin, obligatoirement, par une valeur de retour égale à 1.

VII.2 Création de fonctions : principes et exemples

La partie la plus conséquente de la programmation d'un module correspond à l'écriture des fonctions. Il n'est pas possible de proposer un guide systématique, mais on présente les quelques principes généraux et des exemples de quelques fonctions prélevées dans les librairies présentées à la section précédente.

VII.2.1 Principes de la programmation des fonctions en Perl

Le code source d'un sous-programme ou fonction prend la forme suivante :

```
sub NomFonction ($$)     #1. Nom de la fonction et paramètres
{
   my ($i,$a,$loc) ;     #2. Déclaration des variables locales
   $i=shift(@_) ;        #3. Initialisation des variables locales à
   $a= shift(@_) ;       #   partir du tableau @_ des arguments d'appel
   $loc=… ;              #4. Calculs impliquant les variables locales
   $var2_[$i][$a]=… ;    #   ($i,$a,$loc) et globales ($var1, @var2)
   return $loc ;         #5. Retour éventuel d'une variable locale
}
```

Le début du code des fonctions est toujours de cette forme, avec l'instruction sub (subroutine) suivie du *nom* du sous programme et de la déclaration -facultative- du nombre et du type des *arguments* de cette fonction. Dans l'exemple fourni, les arguments sont deux scalaires, mais il pourrait s'agir aussi d'adresses de tableaux (\@) ou de hachages (\#), de noms logiques de fichiers, ou d'adresses d'éléments de code (&). Si des arguments sont facultatifs, ils doivent apparaître en dernier et figurer après un point-virgule.

Par exemple :

```
sub fonction($\@;$)
```

indique que la fonction a trois arguments dont les deux premiers, obligatoires sont un scalaire et une adresse de tableau et le dernier, facultatif, est un scalaire.

On notera qu'on ne mentionne pas le cas où l'argument serait un tableau (@) plutôt qu'une adresse de tableau (\@). La raison est que l'ensemble des arguments passés à la fonction vont eux-mêmes être rangés dans le tableau par défaut `@_`. Une fonction qui attendrait deux tableaux complets `@tab1` et `@tab2` les rangerait d'office bout-à-bout dans le tableau `@_` sans possibilité de les différencier à nouveau, sauf à passer des nombres d'éléments comme arguments supplémentaires. C'est la raison pour laquelle on préfère en général le passage par adresse `\@` qui, de plus, est plus économe en place mémoire et temps de calcul. Ce passage par adresse n'impose pas que les adresses soient passées sous les formes `\@tab1` et `\@tab2`. Perl reconnaît les formes simples `@tab1` et `@tab2` comme correspondant aux adresses des deux tableaux considérés dès lors qu'il sait que ce sont des adresses qui sont attendues par le sous programme.

Le corps du sous-programme est ensuite compris entre accolades.

La première instruction consiste à déclarer les variables qui seront propres à la procédure, avec une instruction `my`.

Les instructions suivantes consistent à dépiler le contenu du tableau des arguments d'appel, en général à l'aide de l'instruction `shift`. Lorsque la liste d'arguments d'appel est de longueur variable, l'instruction est en général utilisée sous forme conditionnelle, i.e.

```
if (@_) {$param=shift(@_)}
```

qui dit d'affecter à la variable `$param` l'argument suivant de la liste, s'il existe. L'instruction `if` appliquée à un tableau revient en effet à tester que le tableau est non vide.

Le reste du sous-programme effectue les opérations demandées. Si elles portent sur des variables globales extérieures au sous-programme, on les modifie directement. Si la variable globale qu'on veut manipuler est un tableau dont le sous programme ne connaît que l'adresse, le plus pratique est d'utiliser l'opérateur de déréférencement. Par exemple, si un programme doit travailler sur la valeur de la variable globale `@x` pour l'individu `$i`, on peut avoir les instructions types sont :

```
sub fonction ($\@)
{
   my ($i,$refvar)=@_ ;
   $$refvar[$i]=…
}
```

où la syntaxe `$$refvar[$i]` indique à Perl de travaille sur la position `$i` du tableau dont l'adresse est `$refvar`. Dans ce cas, l'appel `fonction(157,@x)` travaillera directement sur `$x[157]`. Cette forme d'écriture est celle qui est généralement utilisée dans toutes les procédures Destinie travaillant sur données individuelles.

Dans l'exemple présenté on suppose par ailleurs que, en sus de modifier des variables globales, le programme retourne aussi la valeur finale de sa variable locale `$loc`. Dans ce cas, elle va être récupérée en appelant la fonction sous la forme, par exemple :

```
$resultat = fonction (15,22) ;
```

mais le retour pourrait aussi bien consister en un tableau complet ou toute autre forme d'objet Perl.

On va maintenant donner les quelques exemples empruntés aux bibliothèques Destinie 1llustrant ces possibilités.

VII.2.2 Un cas élémentaire : l'exemple de la fonction Duree

On commence par l'exemple de la fonction `Duree`, exemple assez simple qui se caractérise néanmoins par le fait que sa liste d'arguments est de longueur totalement indéfinie. La fonction

doit en effet pouvoir calculer la durée passée par l'individu `$i` jusqu'à l'âge `$a` dans un ou plusieurs états vis-à-vis du marché du travail, sans présager du nombre de ces états. Il faut donc pouvoir écrire aussi bien

```
$duree_cho=Duree($i,$a,$CodeCho) ;
$duree_pri=Duree($i,$a,$CodeNC,$CodeCad);
$duree_sal=Duree($i,$a,$CodeNC,$CodeCad,@CodesFP);
$duree_act=Duree($i,$a,@CodesAct);
```

Le texte du sous programme qui permet ces appels diversifiés est donné ci-dessous.

```
sub Duree
{
   my ($i,$a,$duree);
   $i=shift(@_);
   $a=shift(@_);
   $duree=0;
   for (15..$a)
   {
     if  (In($statut_[$i][$_],@_)) {$duree++}
   }
   return $duree;
}
```

On voit dans cet exemple que les deux premiers paramètres `$i` et `$a` sont prélevés d'office dans la liste d'arguments `@_`. Après cela, le sous-programme se borne à laisser cette liste d'arguments en l'état, et à calculer le nombre d'années pour lesquelles la variable `$statut_[$i][$a]` appartient à cet ensemble `@_`, à l'aide de la fonction `In` de `OutilsBase`. La fonction retourne le scalaire `$duree`.

VII.2.3 Utilisation de paramètres légaux dépendant de la date : l'exemple de la fonction CSGRet

Le calcul de la CSG acquittée sur la pension de retraite se fait pour l'individu à l'âge courant. Il dépend de la valeur courante du taux de CSG et de son seuil d'exonération tous deux stockés dans des séries temporelles lues par `DefVarRetr` dans le fichier `ParamSociaux.`

Pour que ceci soit possible, il faut que le programme connaisse la date à laquelle se fait l'évaluation. Celle-ci n'est pas visible depuis le sous-programme car il s'agit d'une variable de boucle qui est forcément locale au programme `.pl` appelant. Pour résoudre ce problème, on aurait pu imaginer que la date `$t` soit aussi passée comme argument au sous-programme, mais on a préféré éviter la multiplication des arguments d'appel. Pour ce faire, le principe général de tous les programmes de même type est plutôt d'évaluer `$t` sur la base de la date de naissance et de l'âge courant de l'individu considéré.

Ceci est fait en ligne `#1` du corps de sous programme ci-dessous, en n'omettant pas de retrancher 1900 pour respecter le fait que les séries temporelles sont toutes décalées à cette date origine. Ceci ayant été fait, le sous programme accède directement aux paramètres `$SeuilExoCSg[$t]` et `$TauxCSGRet[$t]` (lignes #2 et #3).

```
sub CSGRet ($)
{
   my ($i,$t,$cot);
   $i=shift(@_);
   $t=$anaiss[$i]+$age[$i]-1900;                                  #1

   if ($pension[$i]>$SeuilExoCSG[$t])                             #2
   {
      $cot=$TauxCSGRet[$t]*$pension[$i]                           #3
   }
```

```
    else
    {
       $cot=0
    }
    return $cot;
}
```

VII.2.4 Fonction dont un argument est une série temporelle complète : l'exemple de la fonction SalMoy

La fonction `SalMoy` calcule le salaire moyen sur les `$n` dernières années d'activité antérieures à l'âge `$a`, ou par défaut sur l'ensemble de la carrière jusqu'à l'âge `$a`. Elle peut le faire en net ou en brut. Par défaut, les salaires passés sont revalorisés sur les prix, mais l'utilisateur peut choisir une autre règle de revalorisation. Elle a été conçue pour que les appels suivants soient tous possibles :

```
$moyenne=SalMoy($i,"brut",10,@PlafondSS) ;
$moyenne=SalMoy($i,"brut",10) ;
$moyenne=SalMoy($i,"brut",@PlafondSS);
$moyenne=SalMoy($i,"brut");
```

Ceci suppose que le programme admette que le troisième ou le quatrième argument de la fonction puissent être alternativement un nombre ou une adresse de tableau. Le code du programme est le suivant :

```
sub SalMoy
{
   my ($buf,$i,$option,$n,$refrevalo,$t,$a,$num,$denom);
   $i =shift(@_);
   $option =shift(@_);
   $refrevalo=\@Prix;                                              #1
   $n=99;                                                          #2
   while (@_)
   {
      $buf=shift(@_);                                              #3
      if (ref($buf)) {$refrevalo=$buf} else {$n=$buf}              #4
   }

   $a     = $age[$i];
   $t     = ($anaiss[$i]+$a)%1900;
   $num   = 0;
   $denom = 0;
   if ($option =~ /net/i)                                          #5
   {
      while (($denom<$n) && ($a>15) && (($anaiss[$i]+$a)>1920))
      {
         $a--;                                                     #6
         if (In($statut_[$i][$a],@CodesOcc))
         {
            $denom++ ;
            $num+=SalNet($i,$a)
               *$$refrevalo[$t]/$$refrevalo[($anaiss[$i]+$a)%1900] #7
         }
      }
   }
   else
   {
      while (($denom<$n) && ($a>15) && (($anaiss[$i]+$a)>1920))
      {
         $a--;
```

```
            if (In($statut_[$i][$a],@CodesOcc))
            {
               $denom++ ;
               $num+=$salaire_[$i][$a]*
                 $$refrevalo[$t]/$$refrevalo[($anaiss[$i]+$a)%1900]     #8
            }
         }
      }
      if ($denom>0)
      {
         return Arr($num/$denom,0)
      }
}
```

On voit, que dans ce sous-programme, la variable servant de base aux revalorisations de salaires passées est repérée par son adresse `$refrevalo`, qui, par défaut, est l'adresse de la variable globale `@Prix` définie et initialisée par `DefVarRetr` (instruction `#1`). La longueur de carrière sur laquelle va être calculé le salaire moyen est par ailleurs initialisée à 99 (instruction #2).

Le programme commence par prélever d'office les paramètres `$i` et `$option` dans les deux premières positions du tableau `@_`. Ensuite, il ne sait pas immédiatement si le paramètre suivant va être un nombre ou une référence de tableau. Il le stocke donc temporairement dans la variable `$buf`, et il utilise la fonction **`ref`** pour détecter s'il s'agit ou non d'une adresse. Selon le cas, il affecte sa valeur à `$n` ou `$refrevalo` (lignes `#3` et `#4`).

Ceci ayant été fait, on utilise la variable de revalorisation dans des boucles sur `$a` par des appels de type `$$refrevalo[$anaiss[$i]+$a%1900]` (lignes 7 et 8).

Dans le même programme, on notera les instructions conditionnelles de type **`if`** `($option =~ /net/i)` (ligne `#5`) qui permettent que le paramètre `$option` ait été éventuellement passé avec des majuscules et des caractères additionnels : plus précisément, l'instruction `=~` est un opérateur testant si la chaine indiquée à sa droite est présente *à l'intérieur* de la chaîne donnée à sa gauche, l'option `/i` neutralisant le contrôle de la casse des caractères.

On notera enfin le balayage en arrière de la carrière des individus à l'aide de l'opérateur `$a--` (ligne #6).

VII.2.5 Fonction dont un argument est un intervalle de valeurs : l'exemple de la fonction NbEnf

La fonction `NbEnf` est un autre exemple de fonction dans laquelle le même argument peut avoir des contenus différents. Par défaut, elle calcule le nombre d'enfants actuellement survivants de l'individu `$i`, mais elle peut aussi calculer l'ensemble des enfants qu'il a eu (option `"tous"`) ou le nombre d'enfants actuellement vivants et dans un intervalle d'âge donné. On peut ainsi avoir des appels :

```
$descendance_finale[$i]=NbEnf($i,"tous");
```

où :

```
$enfants_03[$i] = NbEnf($i,[0..3]);
```

Dans le deuxième cas, la notation `[0..3]` ne renvoie pas au contenu d'un tableau, comme le ferait la syntaxe `(0..3)` mais à l'adresse d'un tableau anonyme dont le contenu est `(0..3)`.

Dans le texte du programme ci-dessous, on voit que ceci est géré en traitant d'abord à part les cas de l'option par défaut et de l'option `"tous"`. Si on n'est dans aucun de ces deux cas, on en déduit que le paramètre qui a été passé est une référence de tableau, et on transfère le tableau anonyme vers lequel pointe cette adresse dans le tableau non anonyme local `@bornes` (instruction #1), après quoi on reprend le balayage de la liste d'enfants de l'individu `$i` en se restreignant à ceux dont l'âge tombe à l'intérieur du tableau `@bornes` (ligne #2).

Le même procédé est utilisé pour gérer les tranches de tableaux dans les sous-programmes tels que `Coupe`, `Stab` ou `Mtab` de `OutilsTab`.

```
sub NbEnf
{
   my ($i,$e,$option,@bornes,$nenf);
   $i=shift(@_);
   $option="en vie";
   if (@_) {$option=shift(@_)};
   $nenf=undef;
   if ($statut[$i]>0)
   {
      $nenf=0;
      if ($option =~ /en vie/i)
      {
         for $e (1..6) {if ($enf[$i][$e]  > 0) {$nenf++}}
      }
      elsif ($option =~ /tous/i)
      {
         for $e (1..6) {if ($enf[$i][$e] != 0) {$nenf++}}
      }
      else
      {
         @bornes=@{$option};                                        #1
         for $e (1..6)
         {
            if (($enf[$i][$e]>0) &&
                (In($age[$enf[$i][$e]],@bornes))) {$nenf++}         #2
         }
      }
   }
   return $nenf;
}
```

VII.2.6 Fonction dont un argument est un bloc d'instructions : l'exemple de la fonction Svar

Le dernier exemple proposé illustre la possibilité de transmettre un argument consistant en un fragment de code Perl. On a vu que cette possibilité est utilisée systématiquement pour les procédures de tabulation. L'exemple présenté est celui de la fonction `SVar`, qui accepte des syntaxes de type :

```
$somme_sal=SVar(@salaire);
```

où :

```
$somme_sal=SVar(@salaire,sub{($age[$_]>40)
                  && (In($statut[$_],@CodesFP))});
```

Dans cette syntaxe, le premier argument est une adresse de tableau de type `\@`, récupérée dans la variable locale `$var` (ligne #2). Dans les deux exemples ci-dessus, l'élément

`$salaire[$i]` sera donc visible dans le sous-programme sous la forme `$$var[$i]` (ligne #4).

Dans le deuxième exemple d'appel, la séquence `sub{…}` définit pour sa part un mini sous-programme anonyme se résumant à l'instruction `($age[$_]>40) && (In($statut[$_],@CodesFP)`. On préspécifie à `SVar` que c'est ce type d'argument qu'il doit attendre (argument & dans la liste des arguments d'appel de la ligne #1) et l'adresse de sous-programme anonyme est affectée, dans le sous-programme, au scalaire local `$cond` (ligne #3).

Ceci ayant été fait, l'instruction `if (&$cond)` équivaut à `if ($age[$_]>40) && (In($statut[$_],@CodesFP)` et, à l'intérieur d'une boucle sur `$_`, fournit le filtrage souhaité (ligne #5). Si aucun argument n'a été passé, `$cond` pointe vers le sous-programme dont le retour est simplement égal à un, et tous les individus sont pris en compte.

```
sub SVar (\@;&)                                                    #1
{
   my ($var,$cond,$sum);
   $var  = shift(@_);                                              #2
   if (@_) {$cond=shift(@_)} else {$cond=sub {1}};                 #3
   $sum     = 0;
   if ($VarRef eq undef)                                           #4
   {
      print "*** Erreur SVar : variable \$VarRef non définie \n";
      return undef;
   }
   else
   {
      for (1..IMax)
      {
          $sum+=$$var[$_] if (($$VarRef[$_]>$ValRef) && (&$cond))  #5
      }
   }
   return Arr($WSVar*$sum,2);
}
```

On relèvera que les filtrages font aussi intervenir la condition `($$VarRef[$_]>$ValRef)`. Elle sert à épargner à l'utilisateur le filtrage sur les individus effectivement en vie à la date courante. Pour ce faire, on a vu que Destinie suppose qu'il existe une variable de référence des comptages dont l'adresse est stockée dans `$VarRef`. Il suppose également qu'il y a une valeur seuil `$Valref` de cette variable de référence au-delà de laquelle les individus sont à compter comme présents. `$VarRef` et `$ValRef` sont initialisées par défaut dans `DefVarRetr` par la commande `UseVRef(@present,0)`. De cette manière, la condition `if (VarRef[$_]>$ValRef)` équivaut à `if ($present[$_]>0)`. Le cas échéant, on pourrait avoir d'autres préfiltrages en réappelant `UseVRef` avec d'autres arguments.

Références

Albouy, V., Davezies, L. et Debrand, T. (2009) « Health Expenditure Models: a Comparison of five Specifications Using Panel Data", *Document de travail Insee/Dese* G2010/02.

Bardaji, J., Sédillot, B. et Walraët, E. (2003) « Un outil de prospective des retraites : le modèle de microsimulation Destinie », *Économie et Prévision*, n°160-161, pp. 193-213.

Blanchet, D., et Chanut, J.M. (1998) «Projeter les retraites à long terme. Résultats d'un modèle de microsimulation», *Économie et Statistique*, 315: 95-106.

Blanchet, D., Buffeteau, S., Crenner, E. et Le Minez, S. (2010) « Le modèle de microsimulation Destinie 2 : principales caractéristiques et résultats illustratifs », *Document de travail Insee/Dese* n°G2010/13.

Bozio, A. (2008) *Réformes des retraites : estimations sur données françaises*, Thèse EHESS.

Buffeteau, S. et Le Minez, S. (2010) « Le modèle de microsimulation Destinie 2 : le générateur de biographies », *Document de travail Insee-Dese* à paraître.

Division «Redistribution et Politiques Sociales» (1999) «Le modèle de microsimulation dynamique DESTINIE», *Document de travail Insee/Dese* n°G99/13

Duée, M. et Rebillard, C. (2004) « La dépendance des personnes âgées : une projection à long terme », *Document de travail Insee/Dese* n°G2004/02.

O'Donoghue, C., Lennon, J. et Hynes, S. (2009) « The Life-Cycle Income Analysis Model (LIAM): A Study of a Flexible Dynamic Microsimulation Modelling Computing Framework », Working Paper CeRP n°85/09

Statistique Canada (2010) *Guide du concepteur de ModGen* (accessible à http://www.statcan.gc.ca/microsimulation/pdf/dev-guide-fra.pdf)

Annexe 1 : Liste des variables globales par ordre alphabétique

Variable	Contenu	Type de variable	Mode de calcul ou d'initialisation
$age[$i]	Age courant	Variable individuelle permanente	Relec (OutilsRetr)
$AgeAnnDecFP	Age d'annulation de la décote fonction publique.	Paramètre législatif	UseLeg (OutilsRetr)
$AgeEligPR[$t]	Age d'accès à la préretraite	Paramètre législatif	Lue par DefVarRetr (feuille ParamPreret de ParamSociaux.xls)
$ageliq[$i]	Age de la liquidation	Variable individuelle permanente	Liq (OutilsDroits)
$AgeMaxFP	Age de liquidation maximal FP	Paramètre législatif	UseLeg (OutilsRetr)
$AgeMaxRG	Age de liquidation maximal RG	Paramètre législatif	UseLeg (OutilsRetr)
$AgeMinFPA	Age minimum d'ouverture des droits, FPA	Paramètre législatif	UseLeg (OutilsRetr)
$AgeMinFPS	Age minimum d'ouverture des droits, FPS	Paramètre législatif	UseLeg (OutilsRetr)
$AgeMinRG	Age minimum d'ouverture des droits, RG	Paramètre législatif	UseLeg (OutilsRetr)
$anaiss[$i]	Année de naissance	Variable individuelle permanente	Relec (OutilsRetr)
$AnneeDepart	Millésime associé au fichier de données initiales	Option ou paramètre de simulation	UseBios (OutilsRetr)
$AnneeDepartCN	Année de démarrage du système en comptes notionels	Option ou paramètre de simulation	UseOptCN (OutilsRetr)
$BioEmp	Fichier en entrée contenant les biographies professionnelles	Nom de fichier ou de repertoire	UseBios (OutilsRetr)
$BioFam	Fichier en entrée contenant les biographies familiales	Nom de fichier ou de repertoire	UseBios (OutilsRetr)
$BMAF[$t]	Base mensuelle des allocations familiales	Paramètre législatif	Lue par DefVarRetr (feuille ParamFam de ParamSociaux.xls)
$CibleLiq	Nom du tableau de stockage cibles d'age de départ à retraite	Option ou paramètre de simulation	UseOpt (OutilsRetr)
$CodeAvpf	Code inactif avec AVPF	Code ou label	Initialisation dans DefVarRetr
$CodeCad	Code cadre du privé	Code ou label	Initialisation dans DefVarRetr
$CodeCel	Code célibataire	Code ou label	Initialisation dans DefVarRetr
$CodeCho	Code chômeur	Code ou label	Initialisation dans DefVarRetr
$CodeFem	Code femme	Code ou label	Initialisation dans DefVarRetr
$CodeFP	Code fonction publique	Code ou label	Initialisation dans DefVarRetr
$CodeFPA	Code fonction publique catégorie active	Code ou label	Initialisation dans DefVarRetr
$CodeFPS	Code fonction publique catégorie sédentaire	Code ou label	Initialisation dans DefVarRetr
$CodeHom	Code homme	Code ou label	Initialisation dans DefVarRetr
$CodeIna	Code inactif	Code ou label	Initialisation dans DefVarRetr
$CodeInd	Code indépendant	Code ou label	Initialisation dans DefVarRetr
$CodeMar	Code marié (en couple)	Code ou label	Initialisation dans DefVarRetr
$CodeNC	Code non cadre du privé	Code ou label	Initialisation dans DefVarRetr
$CodePR	Code préretraité	Code ou label	Initialisation dans DefVarRetr
$CodesAct[$_]	Liste des codes actifs	Code ou label	Initialisation dans DefVarRetr

Variable	Contenu	Type de variable	Mode de calcul ou d'initialisation
$CodeSco	Code élève, étudiant	Code ou label	Initialisation dans DefVarRetr
$CodesFP[$_]	Liste des codes fonction publique	Code ou label	Initialisation dans DefVarRetr
$CodesOcc[$_]	Listes des codes actifs occupés	Code ou label	Initialisation dans DefVarRetr
$CodeVeu	Code veuf	Code ou label	Initialisation dans DefVarRetr
$CoefMdaFP[$t]	Coefficient MDA de la FP	Paramètre législatif	Lue par DefVarRetr (feuille ParamRetrBase de ParamSociaux.xls)
$CoefMdaRG[$t]	Coefficient MDA du RG	Paramètre législatif	Lue par DefVarRetr (feuille ParamRetrBase de ParamSociaux.xls)
$conjoint[$i]	Identifiant du conjoint	Variable individuelle permanente	Relec (OutilsRetr)
$deces[$i]	Indicatrice de décès l'année en cours	Variable individuelle permanente	Relec (OutilsRetr)
$DecoteFP	Décote fonction publique	Paramètre législatif	UseLeg (OutilsRetr)
$DecoteRG	Coefficient de décote du régime général (par année d'écart au taux plein)	Paramètre législatif	UseLeg (OutilsRetr)
$duree_avpf	Durée passée en AVPF	Variable individuelle temporaire	SimDir(OutilsRetr), Liq et DurBase (OutilsDroits)
$duree_cho	Nombre d'années au chômage	Variable individuelle temporaire	SimDir(OutilsRetr), Liq et DurBase (OutilsDroits)
$duree_emp	Nombre d'années en emploi	Variable individuelle temporaire	SimDir(OutilsRetr), Liq et DurBase (OutilsDroits)
$duree_fp	Durée cotisée dans les régimes du secteur public	Variable individuelle temporaire	SimDir(OutilsRetr), Liq et DurBase (OutilsDroits)
$duree_fp_maj	Durée FP incluant MDA éventuelle (selon options)	Variable individuelle temporaire	Simdir (OutilsRetr), Liq et DurMajo (OutilsDroits)
$duree_fpa	Durée cotisée dans les régimes du secteur public, catégorie active	Variable individuelle temporaire	SimDir(OutilsRetr), Liq et DurBase (OutilsDroits)
$duree_fps	Durée cotisée dans les régimes du secteur public, catégorie sédentaire	Variable individuelle temporaire	SimDir(OutilsRetr), Liq et DurBase (OutilsDroits)
$duree_in	Durée cotisée comme indépendant	Variable individuelle temporaire	SimDir(OutilsRetr), Liq et DurBase (OutilsDroits)
$duree_in_maj	Durée cotisée comme indépendant incluant MDA éventuelle (selon options)	Variable individuelle temporaire	Simdir (OutilsRetr), Liq et DurMajo (OutilsDroits)
$duree_PR	Nombre d'années en préretraite	Variable individuelle temporaire	SimDir(OutilsRetr), Liq et DurBase (OutilsDroits)
$duree_rg	Durée cotisée au régime général	Variable individuelle temporaire	SimDir(OutilsRetr), Liq et DurBase (OutilsDroits)
$duree_rg_maj	Durée RG incluant AVPF et MDA (selon options)	Variable individuelle temporaire	Simdir (OutilsRetr), Liq et DurMajo (OutilsDroits)
$duree_tot	Durée cotisée totale	Variable individuelle temporaire	SimDir(OutilsRetr), Liq et DurBase (OutilsDroits)
$duree_tot_maj	Durée cotisée totale incluant AVPF et MDA (selon options)	Variable individuelle temporaire	Simdir (OutilsRetr), Liq et DurMajo (OutilsDroits)
$DureeCalcSAM	Nombre d'années pour calcul du SAM dans le RG	Paramètre législatif	UseLeg (OutilsRetr)
$DureeCibFP	Durée cible de la fonction publique	Paramètre législatif	UseLeg (OutilsRetr)
$DureeCibRG	Durée cible régime général	Paramètre législatif	UseLeg (OutilsRetr)
$DureeProratRG	Durée de proratisation du régime général	Paramètre législatif	UseLeg (OutilsRetr)
$enf[$e][$i]	Identifiant du e-ième enfant	Variable individuelle permanente	Relec (OutilsRetr)
$findet[$i]	Age de fin d'études	Variable individuelle permanente	Relec (OutilsRetr)
$gamma[$i]	Aversion pour le risque	Variable individuelle permanente	Relec (OutilsRetr)

Variable	Contenu	Type de variable	Mode de calcul ou d'initialisation
$id_clone	Identifiant de l'individu de départ lorsqu'on travaille sur un clone	Variable individuelle temporaire	Clone (DefVarRetr)
$indic_mc[$i]	Indicatrice d'octroi du minimum contributif	Variable individuelle permanente	SimDir (OutilsRetr), Liq et Revalo (OutilsDroits)
$indic_mg[$i]	Indicatrice d'octroi du minimum garanti de la FP	Variable individuelle permanente	SimDir (OutilsRetr), Liq et Revalo (OutilsDroits)
$k[$i]	Préférence pour le loisir/pénibilité du travail	Variable individuelle permanente	Relec (OutilsRetr)
$Label{".."}[$_]	Labels des codes sexe, situation matrimoniale ou statut d'emploi	Code ou label	Initialisation dans DefVarRetr
$LegRetroMax	Législation en vigueur pour la rétropolation des retraites	Option ou paramètre de simulation	UseLegRetroMax (OutilsRetr)
$MajoPlafCF[$t]	Majoration du plafond ressources du CF et de l'APJE	Paramètre législatif	Lue par DefVarRetr (feuille ParamFam de ParamSociaux.xls)
$MajTauxRGMax	Majoration du taux maximal de la pension RG	Paramètre législatif	UseLeg (OutilsRetr)
$matri[$i]	Statut matrimonial	Variable individuelle permanente	Relec (OutilsRetr)
$MaxRevRG[$t]	Réversion maximale du RG	Paramètre législatif	Lue par DefVarRetr (feuille ParamRev de ParamSociaux.xls)
$mere[$i]	Identifiant de la mère	Variable individuelle permanente	Relec (OutilsRetr)
$migrant[$i]	Indique si l'individu est un migrant de l'année	Variable individuelle permanente	Relec (OutilsRetr)
$min_cont	Minimum contributif	Variable individuelle temporaire	SimDir (OutilsRetr), Liq et MinCont (OutilsDroits)
$min_garanti	Minimum garanti	Variable individuelle temporaire	SimDir (OutilsRetr), Liq et MinGaranti (OutilsDroits)
$min_vieil[$i]	Allocation du minimum vieillesse	Variable individuelle permanente	SimMinVieil (OutilsRetr)
$Mincont1[$t]	Minimum contributif	Paramètre législatif	Lue par DefVarRetr (feuille ParamRetrBase de ParamSociaux.xls)
$Mincont2[$t]	Majoration du minimum contributif	Paramètre législatif	Lue par DefVarRetr (feuille ParamRetrBase de ParamSociaux.xls)
$MinPR[$t]	Niveau minimum de la préretraite	Paramètre législatif	Lue par DefVarRetr (feuille ParamRetrBase de ParamSociaux.xls)
$MinRevRG[$t]	Réversion minimale du RG	Paramètre législatif	Lue par DefVarRetr (feuille ParamRev de ParamSociaux.xls)
$MinVieil1[$t]	Minimum vieillesse personne seule	Paramètre législatif	Lue par DefVarRetr (feuille ParamRetrBase de ParamSociaux.xls)
$MinVieil2[$t]	Minimum vieillesse pour un couple	Paramètre législatif	Lue par DefVarRetr (feuille ParamRetrBase de ParamSociaux.xls)
$Options	Chaine de caractères synthétisant différentes options de simulation	Option ou paramètre de simulation	UseOpt (OutilsRetr)
$OptionsCN	Chaine de caractères synthétisant les options spécifiques aux comptes notionnels	Option ou paramètre de simulation	UseOptCN (OutilsRetr)
$pension[$i]	Pension directe totale	Variable individuelle permanente	SimDir (OutilsRetr), Liq et Revalo (OutilsDroits)
$pension_ag[$i]	Pension agirc	Variable individuelle permanente	SimDir (OutilsRetr), Liq et Revalo (OutilsDroits)
$pension_ar[$i]	Pension arrco	Variable individuelle permanente	SimDir (OutilsRetr), Liq et Revalo (OutilsDroits)
$pension_cn_fp[$i]	Pension comptes notionnels secteur public	Variable individuelle permanente	SimDir (OutilsRetr), Liq et Revalo (OutilsDroits)
$pension_cn_pri[$i]	Pension comptes notionnels secteur privé	Variable individuelle permanente	SimDir (OutilsRetr), Liq et Revalo (OutilsDroits)
$pension_fp[$i]	Pension fonction publique	Variable individuelle permanente	SimDir (OutilsRetr), Liq et Revalo (OutilsDroits)
$pension_in[$i]	Pension d'indépendant	Variable individuelle permanente	SimDir (OutilsRetr), Liq et Revalo (OutilsDroits)
$pension_rg[$i]	Pension régime général	Variable individuelle permanente	SimDir (OutilsRetr), Liq et Revalo (OutilsDroits)

Variable	Contenu	Type de variable	Mode de calcul ou d'initialisation
$pere[$i]	Identifiant du père	Variable individuelle permanente	Relec (OutilsRetr)
$PlafARS1[$t]	Plafond de ressource de l'ARS (1er enfant)	Paramètre législatif	Lue par DefVarRetr (feuille ParamFam de ParamSociaux.xls)
$PlafARS2[$t]	Plafond de ressource de l'ARS (2eme enfant)	Paramètre législatif	Lue par DefVarRetr (feuille ParamFam de ParamSociaux.xls)
$PlafARS3[$t]	Plafond de ressource de l'ARS (3eme enfant)	Paramètre législatif	Lue par DefVarRetr (feuille ParamFam de ParamSociaux.xls)
$PlafARS4[$t]	Plafond de ressource de l'ARS (4eme enfant)	Paramètre législatif	Lue par DefVarRetr (feuille ParamFam de ParamSociaux.xls)
$PlafARS5[$t]	Plafond de ressource de l'ARS (enfant suppl.)	Paramètre législatif	Lue par DefVarRetr (feuille ParamFam de ParamSociaux.xls)
$PlafCF1[$t]	Plafond ressources du CF et de l'APJE (1er enfant)	Paramètre législatif	Lue par DefVarRetr (feuille ParamFam de ParamSociaux.xls)
$PlafCF2[$t]	Plafond ressources du CF et de l'APJE (2eme enfant)	Paramètre législatif	Lue par DefVarRetr (feuille ParamFam de ParamSociaux.xls)
$PlafCF3[$t]	Plafond ressources du CF et de l'APJE (3eme enfant)	Paramètre législatif	Lue par DefVarRetr (feuille ParamFam de ParamSociaux.xls)
$PlafCF4[$t]	Plafond ressources du CF et de l'APJE (4eme enfant)	Paramètre législatif	Lue par DefVarRetr (feuille ParamFam de ParamSociaux.xls)
$PlafCF5[$t]	Plafond ressources du CF et de l'APJE (enfant suppl.)	Paramètre législatif	Lue par DefVarRetr (feuille ParamFam de ParamSociaux.xls)
$PlafondSS[$t]	Plafond de la sécurité sociale (annuel brut, nominal)	Paramètre législatif	Lue par DefVarRetr (feuille ParamGene de ParamSociaux.xls)
$PlafRevRG[$t]	Plafond de ressources reversions RG et In	Paramètre législatif	Lue par DefVarRetr (feuille ParamRev de ParamSociaux.xls)
$PointFP[$t]	Valeur du point fonction publique	Paramètre législatif	Lue par DefVarRetr (feuille ParamGene de ParamSociaux.xls)
$points_agirc	Cumul de points AGIRC	Variable individuelle temporaire	Simdir (OutilsRetr), Liq et Points (OutilsDroits)
$points_arrco	Cumul de points ARRCO	Variable individuelle temporaire	Simdir (OutilsRetr), Liq et Points (OutilsDroits)
$points_cn_fp[$i]	Cumul points comptes notionnels du secteur public	Variable individuelle temporaire	Simdir (OutilsRetr), Liq et Points (OutilsDroits)
$points_cn_pri[$i]	Cumul points compte notionnel du secteur privé	Variable individuelle temporaire	Simdir (OutilsRetr), Liq et Points (OutilsDroits)
$present[$i]	Indicatrice de présence dans la population à la date courante	Variable individuelle permanente	Relec (OutilsRetr)
$Prix[$t]	Indice de prix (2005=1)	Paramètre législatif	Lue par DefVarRetr (feuille ParamGene de ParamSociaux.xls)
$quotient[$s][$t][$a]	Quotients de mortalité par sexe, date et âge	Paramètres démographiques	Lue par DefVarRetr (fichier ParamMortal.xls)
$RendementCN[$t]	Rendement effectif du système en comptes notionnels	Paramètre législatif	A renseigner par l'utilisateur
$RendementCNPrev[$t]	Rendement prévisionnel du système en comptes notionnels	Paramètre législatif	A renseigner par l'utilisateur
$RepBios	Repertoire regroupant la collection de fichiers Bios	Nom de fichier ou de repertoire	Reconstitution automatique par DefVarRetr
$RepMain	Repertoire racine de l'ensemble des programmes Destinie	Nom de fichier ou de repertoire	Reconstitution automatique par DefVarRetr
$RepOutils	Repertoire contenant les bibliothèques Destinie	Nom de fichier ou de repertoire	Reconstitution automatique par DefVarRetr
$RepTrav	Repertoire ou reside le programme appelant	Nom de fichier ou de repertoire	Reconstitution automatique par DefVarRetr
$rev[$i]	Réversion totale (somme des cinq précédentes)	Variable individuelle permanente	SimDer (OutilsRetr), Revalo (OutilsDroits)
$rev_ag[$i]	Reversion agirc	Variable individuelle permanente	SimDer (OutilsRetr), Revalo (OutilsDroits)
$rev_ar[$i]	Reversion arrco	Variable individuelle permanente	SimDer (OutilsRetr), Revalo (OutilsDroits)
$rev_cn_fp[$i]	Reversions comptes notionels secteur public	Variable individuelle permanente	SimDer (OutilsRetr), Revalo (OutilsDroits)
$rev_cn_pri[$i]	Reversions comptes notionels secteur privé	Variable individuelle permanente	SimDer (OutilsRetr), Revalo (OutilsDroits)

Variable	Contenu	Type de variable	Mode de calcul ou d'initialisation
$rev_fp[$i]	Réversion fonction publique	Variable individuelle permanente	SimDer (OutilsRetr), Revalo (OutilsDroits)
$rev_in[$i]	Réversion d'indépendant	Variable individuelle permanente	SimDer (OutilsRetr), Revalo (OutilsDroits)
$rev_rg[$i]	Réversion régime général	Variable individuelle permanente	SimDer (OutilsRetr), Revalo (OutilsDroits)
$RevaloRG[$t]	Coeff revalo nominale des pensions RG en cours de service	Paramètre législatif	Lue par DefVarRetr (feuille ParamRetrBase de ParamSociaux.xls)
$RevaloFP[$t]	Coeff revalo nominale des pensions FP en cours de service	Paramètre législatif	Lue par DefVarRetr (feuille ParamRetrBase de ParamSociaux.xls)
$RevaloCN[$t]	Revalorisation pension comptes notionnels	Paramètre législatif	A renseigner par l'utilisateur
$RevaloSPC[$t]	Coeff de revalorisation des salaires portés au comptes	Paramètre législatif	Lue par DefVarRetr (feuille ParamRetrBase de ParamSociaux.xls)
$salaire[$i]	Salaire courant (équivaut à $salaire_[$i][$age[$i]])	Variable individuelle permanente	Relec (OutilsRetr)
$salaire_[$i][$a]	Salaire par âge	Variable individuelle permanente	Relec (OutilsRetr)
$SalRefAGIRC[$t]	Salaire de référence AGIRC (nominal)	Paramètre législatif	Lue par DefVarRetr (feuille ParamComp de ParamSociaux.xls)
$SalRefARRCO[$t]	Salaire de référence ARRCO (nominal)	Paramètre législatif	Lue par DefVarRetr (feuille ParamComp de ParamSociaux.xls)
$sam_in	SAM pour le calcul de la pension d'indépendant	Variable individuelle temporaire	Simdir (OutilsRetr), Liq et SalBase (OutilsDroits)
$sam_rg	SAM servant au calcul de la pension RG	Variable individuelle temporaire	Simdir (OutilsRetr), Liq et SalBase (OutilsDroits)
$seuil[$i]	Seuil de déclenchement du report du départ en retraite	Variable individuelle permanente	Relec (OutilsRetr)
$SeuilExoCSG[$t]	Seuil d'exonération de la CSG sur les retraites	Paramètre législatif	Lue par DefVarRetr (feuille ParamAutres de ParamSociaux.xls)
$sexe[$i]	Sexe	Variable individuelle permanente	Relec (OutilsRetr)
$SMIC[$t]	Le SMIC	Paramètre législatif	Lue par DefVarRetr (feuille ParamGene de ParamSociaux.xls)
$SousRepBios	Sous-repertoire de $RepBios effectivement utilisé	Nom de fichier ou de repertoire	Reconstitution automatique par DefVarRetr
$sr_fp	Salaire de référence pour le calcul de la pension FP	Variable individuelle temporaire	Simdir (OutilsRetr), Liq et SalBase (OutilsDroits)
$statut[$i]	Statut courant (équivaut à $statut_[$i][$age[$i]])	Variable individuelle permanente	Relec (OutilsRetr)
$statut_[$i][$a]	Statut d'emploi par âge	Variable individuelle permanente	Relec (OutilsRetr)
$SurcoteFP	Surcote fonction publique	Paramètre législatif	UseLeg (OutilsRetr)
$SurcoteRG1	Coefficient de surcote du régime général, 1ere année	Paramètre législatif	UseLeg (OutilsRetr)
$SurcoteRG2	Coefficient de surcote du régime général, 2eme année	Paramètre législatif	UseLeg (OutilsRetr)
$SurcoteRG3	Coefficient de surcote du régime général, 3eme année	Paramètre législatif	UseLeg (OutilsRetr)
$survie[$s][$t][$a]	Survies par sexe, date et âge	Paramètres démographiques	Lue par DefVarRetr (fichier ParamMortal.xls)
$taux[$i]	Préférence pour le présent	Variable individuelle permanente	Relec (OutilsRetr)
$taux_prim[$i]	Taux de prime dans la fonction publique	Variable individuelle permanente	Relec (OutilsRetr)
$TauxAGFF[$t]	Taux AGFF (depuis 2001)	Paramètre législatif	Lue par DefVarRetr (feuille ParamAutres de ParamSociaux.xls)
$TauxAGIRC_B[$t]	Taux AGIRC tranche B	Paramètre législatif	Lue par DefVarRetr (feuille ParamComp de ParamSociaux.xls)
$TauxAGIRC_C[$t]	Taux AGIRC tranche C	Paramètre législatif	Lue par DefVarRetr (feuille ParamComp de ParamSociaux.xls)
$TauxAGIRC_SB[$t]	Taux contractuel salarié AGIRC tranche B	Paramètre législatif	Lue par DefVarRetr (feuille ParamComp de ParamSociaux.xls)
$TauxAGIRC_SC[$t]	Taux contractuel salarié AGIRC tranche C	Paramètre législatif	Lue par DefVarRetr (feuille ParamComp de ParamSociaux.xls)
$TauxAppAGIRC[$t]	Taux Appel AGIRC	Paramètre législatif	Lue par DefVarRetr (feuille ParamComp de ParamSociaux.xls)

Variable	Contenu	Type de variable	Mode de calcul ou d'initialisation
$TauxAppARRCO[$t]	Taux appel ARRCO	Paramètre législatif	Lue par DefVarRetr (feuille ParamComp de ParamSociaux.xls)
$TauxARRCO_1[$t]	Taux contractuel ARRCO tranche 1	Paramètre législatif	Lue par DefVarRetr (feuille ParamComp de ParamSociaux.xls)
$TauxARRCO_2[$t]	Taux contractuel ARRCO tranche 2	Paramètre législatif	Lue par DefVarRetr (feuille ParamComp de ParamSociaux.xls)
$TauxARRCO_S1[$t]	Taux contractuel salarié ARRCO tranche 1	Paramètre législatif	Lue par DefVarRetr (feuille ParamComp de ParamSociaux.xls)
$TauxARRCO_S2[$t]	Taux contractuel salarié ARRCO tranche 2	Paramètre législatif	Lue par DefVarRetr (feuille ParamComp de ParamSociaux.xls)
$TauxASSEDIC[$t]	Taux assurance chômage	Paramètre législatif	Lue par DefVarRetr (feuille ParamAutres de ParamSociaux.xls)
$TauxCotCN[$t]	Taux de cotisation aux comptes notionnels	Paramètre législatif	A renseigner par l'utilisateur
$TauxCotFP[$t]	Taux de cotisation salarié du secteur public	Paramètre législatif	Lue par DefVarRetr (feuille ParamRetrBase de ParamSociaux.xls)
$TauxCSGRet[$t]	Taux CSG sur retraites	Paramètre législatif	Lue par DefVarRetr (feuille ParamAutres de ParamSociaux.xls)
$TauxCSGSal[$t]	Taux CSG sur salaires	Paramètre législatif	Lue par DefVarRetr (feuille ParamAutres de ParamSociaux.xls)
$TauxEmpRGSP[$t]	Taux employeur régime général sous plafond	Paramètre législatif	Lue par DefVarRetr (feuille ParamRetrBase de ParamSociaux.xls)
$TauxMalSP[$t]	Taux de cotisation maladie sous plafond (jusqu'en 1979)	Paramètre législatif	Lue par DefVarRetr (feuille ParamAutres de ParamSociaux.xls)
$TauxMalTot[$t]	Taux de cotisation maladie déplafonnée (depuis 1967)	Paramètre législatif	Lue par DefVarRetr (feuille ParamAutres de ParamSociaux.xls)
$TauxPleinRG	Taux plein du régime général	Paramètre législatif	UseLeg (OutilsRetr)
$TauxPleinRG	Taux plein du régime général	Paramètre législatif	UseLeg (OutilsRetr)
$TauxPR1[$t]	Taux préretraite tranche 1	Paramètre législatif	Lue par DefVarRetr (feuille ParamPreret de ParamSociaux.xls)
$TauxPR2[$t]	Taux préretraite tranche 2	Paramètre législatif	Lue par DefVarRetr (feuille ParamPreret de ParamSociaux.xls)
$TauxPR3[$t]	Taux préretraite tranche 3	Paramètre législatif	Lue par DefVarRetr (feuille ParamPreret de ParamSociaux.xls)
$TauxRevAGIRC[$t]	Taux de réversion AGIRC	Paramètre législatif	Lue par DefVarRetr (feuille ParamRev de ParamSociaux.xls)
$TauxRevARRCO[$t]	Taux de réversion ARRCO	Paramètre législatif	Lue par DefVarRetr (feuille ParamRev de ParamSociaux.xls)
$TauxRevFP[$t]	Taux de réversion de la FP	Paramètre législatif	Lue par DefVarRetr (feuille ParamRev de ParamSociaux.xls)
$TauxRevInd[$t]	Taux de réversion des régimes d'indépendants	Paramètre législatif	Lue par DefVarRetr (feuille ParamRev de ParamSociaux.xls)
$TauxRevRG[$t]	Taux de réversion du RG	Paramètre législatif	Lue par DefVarRetr (feuille ParamRev de ParamSociaux.xls)
$TauxRGMax	Taux maximal de la pension RG	Paramètre législatif	UseLeg (OutilsRetr)
$TauxRGMax	Taux maximal de la pension RG	Paramètre législatif	UseLeg (OutilsRetr)
$TauxRGSalTot[$t]	Taux RG sur l'ensemble du salaire	Paramètre législatif	Lue par DefVarRetr (feuille ParamRetrBase de ParamSociaux.xls)
$TauxSalRGSP[$t]	Taux salarié régime général sous plafond	Paramètre législatif	Lue par DefVarRetr (feuille ParamRetrBase de ParamSociaux.xls)
$TauxSondage	Taux de sondage de l'échantillon traité	Option ou paramètre de simulation	UseBios (OutilsRetr)
$ValPtAGIRC[$t]	Valeur du point AGIRC (nominal)	Paramètre législatif	Lue par DefVarRetr (feuille ParamComp de ParamSociaux.xls)
$ValPtARRCO[$t]	Valeur du point ARRCO (nominal)	Paramètre législatif	Lue par DefVarRetr (feuille ParamComp de ParamSociaux.xls)

Annexe 2 : Variables globales classées par type et mode de calcul ou d'initialisation

Type de variable	Mode de calcul ou d'initialisation	Variable	Contenu
Codes ou labels	Initialisation dans DefVarRetr	$CodeAvpf	Code inactif avec AVPF
		$CodeCad	Code cadre du privé
		$CodeCel	Code célibataire
		$CodeCho	Code chômeur
		$CodeFem	Code femme
		$CodeFP	Code fonction publique
		$CodeFPA	Code fonction publique catégorie active
		$CodeFPS	Code fonction publique catégorie sédentaire
		$CodeHom	Code homme
		$CodeIna	Code inactif
		$CodeInd	Code indépendant
		$CodeMar	Code marié (en couple)
		$CodeNC	Code non cadre du privé
		$CodePR	Code préretraité
		$CodesAct[$_]	Liste des codes actifs
		$CodeSco	Code élève, étudiant
		$CodesFP[$_]	Liste des codes fonction publique
		$CodesOcc[$_]	Listes des codes actifs occupés
		$CodeVeu	Code veuf
		$Label{"..."}[$_]	Labels des codes sexe, situation matrimoniale ou statut d'emploi
Noms de fichiers ou de répertoires	Reconstitution automatique par DefVarRetr	$RepBios	Repertoire regroupant la collection de fichiers Bios
		$RepMain	Repertoire racine de l'ensemble des programmes Destinie
		$RepOutils	Repertoire contenant les bibliothèques Destinie
		$RepTrav	Repertoire ou reside le programme appelant
		$SousRepBios	Sous-repertoire de $RepBios effectivement utilisé
	UseBios (OutilsRetr)	$BioEmp	Fichier en entrée contenant les biographies professionnelles
		$BioFam	Fichier en entrée contenant les biographies familiales
Options ou paramètres de simulation	UseBios (OutilsRetr)	$AnneeDepart	Millésime associé au fichier de données initiales
		$TauxSondage	Taux de sondage de l'échantillon traité
Options ou paramètres de simulation	UseLegRetroMax (OutilsRetr)	$LegRetroMax	Législation en vigueur pour la rétropolation des retraites
	UseOpt (OutilsRetr)	$CibleLiq	Nom du tableau de stockage cibles d'age de départ à retraite
		$Options	Chaine de caractères synthétisant différentes options de simulation
	UseOptCN (OutilsRetr)	$AnneeDepartCN	Année de démarrage du système en comptes notionels

Type de variable	Mode de calcul ou d'initialisation	Variable	Contenu
		$OptionsCN	Chaine de caractères synthétisant les options spécifiques aux comptes notionnels
Paramètres législatifs	A renseigner par l'utilisateur	$RendementCN[$t]	Rendement effectif du système en comptes notionnels
		$RendementCNPrev[$t]	Rendement prévisionnel du système en comptes notionnels
		$RevaloCN[$t]	Revalorisation pension comptes notionnels
		$TauxCotCN[$t]	Taux de cotisation aux comptes notionnels
	Lus par DefVarRetr (feuille ParamAutres de ParamSociaux.xls)	$SeuilExoCSG[$t]	Seuil d'exonération de la CSG sur les retraites
		$TauxAGFF[$t]	Taux AGFF (depuis 2001)
		$TauxASSEDIC[$t]	Taux assurance chômage
		$TauxCSGRet[$t]	Taux CSG sur retraites
		$TauxCSGSal[$t]	Taux CSG sur salaires
		$TauxMalSP[$t]	Taux de cotisation maladie sous plafond (jusqu'en 1979)
		$TauxMalTot[$t]	Taux de cotisation maladie déplafonnée (depuis 1967)
	Lus par DefVarRetr (feuille ParamComp de ParamSociaux.xls)	$SalRefAGIRC[$t]	Salaire de référence AGIRC (nominal)
		$SalRefARRCO[$t]	Salaire de référence ARRCO (nominal)
		$TauxAGIRC_B[$t]	Taux AGIRC tranche B
		$TauxAGIRC_C[$t]	Taux AGIRC tranche C
		$TauxAGIRC_SB[$t]	Taux contractuel salarié AGIRC tranche B
		$TauxAGIRC_SC[$t]	Taux contractuel salarié AGIRC tranche C
		$TauxAppAGIRC[$t]	Taux Appel AGIRC
		$TauxAppARRCO[$t]	Taux appel ARRCO
		$TauxARRCO_1[$t]	Taux contractuel ARRCO tranche 1
		$TauxARRCO_2[$t]	Taux contractuel ARRCO tranche 2
		$TauxARRCO_S1[$t]	Taux contractuel salarié ARRCO tranche 1
		$TauxARRCO_S2[$t]	Taux contractuel salarié ARRCO tranche 2
		$ValPtAGIRC[$t]	Valeur du point AGIRC (nominal)
		$ValPtARRCO[$t]	Valeur du point ARRCO (nominal)
Paramètres législatifs	Lus par DefVarRetr (feuille ParamFam de ParamSociaux.xls)	$BMAF[$t]	Base mensuelle des allocations familiales
		$MajoPlafCF[$t]	Majoration du plafond ressources du CF et de l'APJE
		$PlafARS1[$t]	Plafond de ressource de l'ARS (1er enfant)
		$PlafARS2[$t]	Plafond de ressource de l'ARS (2eme enfant)
		$PlafARS3[$t]	Plafond de ressource de l'ARS (3eme enfant)
		$PlafARS4[$t]	Plafond de ressource de l'ARS (4eme enfant)
		$PlafARS5[$t]	Plafond de ressource de l'ARS (enfant suppl.)

Type de variable	Mode de calcul ou d'initialisation	Variable	Contenu
		$PlafCF1[$t]	Plafond ressources du CF et de l'APJE (1er enfant)
		$PlafCF2[$t]	Plafond ressources du CF et de l'APJE (2eme enfant)
		$PlafCF3[$t]	Plafond ressources du CF et de l'APJE (3eme enfant)
		$PlafCF4[$t]	Plafond ressources du CF et de l'APJE (4eme enfant)
		$PlafCF5[$t]	Plafond ressources du CF et de l'APJE (enfant suppl.)
	Lus par DefVarRetr (feuille ParamGene de ParamSociaux.xls)	$PlafondSS[$t]	Plafond de la sécurité sociale (annuel brut, nominal)
		$PointFP[$t]	Valeur du point fonction publique
		$Prix[$t]	Indice de prix (2005=1)
		$SMIC[$t]	Le SMIC
	Lus par DefVarRetr (feuille ParamPreret de ParamSociaux.xls)	$AgeEligPR[$t]	Age d'accès à la préretraite
		$TauxPR1[$t]	Taux préretraite tranche 1
		$TauxPR2[$t]	Taux préretraite tranche 2
		$TauxPR3[$t]	Taux préretraite tranche 3
	Lus par DefVarRetr (feuille ParamRetrBase de ParamSociaux.xls)	$CoefMdaFP[$t]	Coefficient MDA de la FP
		$CoefMdaRG[$t]	Coefficient MDA du RG
		$Mincont1[$t]	Minimum contributif
		$Mincont2[$t]	Majoration du minimum contributif
		$MinPR[$t]	Niveau minimum de la préretraite
		$MinVieil1[$t]	Minimum vieillesse personne seule
		$MinVieil2[$t]	Minimum vieillesse pour un couple
		$RevaloRG[$t]	Coeff revalo nominale des pensions RG en cours de service
		$RevaloFP[$t]	Coeff revalo nominale des pensions FP en cours de service
		$RevaloSPC[$t]	Coeff de revalorisation des salaires portés au comptes
		$TauxCotFP[$t]	Taux de cotisation salarié du secteur public
		$TauxEmpRGSP[$t]	Taux employeur régime général sous plafond
Paramètres législatifs	Lus par DefVarRetr (feuille ParamRetrBase de ParamSociaux.xls)	$TauxRGSalTot[$t]	Taux RG sur l'ensemble du salaire
		$TauxSalRGSP[$t]	Taux salarié régime général sous plafond
	Lus par DefVarRetr (feuille ParamRev de ParamSociaux.xls)	$MaxRevRG[$t]	Réversion maximale du RG
		$MinRevRG[$t]	Réversion minimale du RG
		$PlafRevRG[$t]	Plafond de ressources reversions RG et In
		$TauxRevAGIRC[$t]	Taux de réversion AGIRC
		$TauxRevARRCO[$t]	Taux de réversion ARRCO
		$TauxRevFP[$t]	Taux de réversion de la FP
		$TauxRevInd[$t]	Taux de réversion des régimes d'indépendants

Type de variable	Mode de calcul ou d'initialisation	Variable	Contenu
		$TauxRevRG[$t]	Taux de réversion du RG
	UseLeg (OutilsRetr)	$AgeAnnDecFP	Age d'annulation de la décote fonction publique.
		$AgeMaxFP	Age de liquidation maximal FP
		$AgeMaxRG	Age de liquidation maximal RG
		$AgeMinFPA	Age minimum d'ouverture des droits, FPA
		$AgeMinFPS	Age minimum d'ouverture des droits, FPS
		$AgeMinRG	Age minimum d'ouverture des droits, RG
		$DecoteFP	Décote fonction publique
		$DecoteRG	Coefficient de décote du régime général
		$DureeCalcSAM	Nombre d'années pour calcul du SAM dans le RG
		$DureeCibFP	Durée cible de la fonction publique
		$DureeCibRG	Durée cible régime général
		$DureeProratRG	Durée de proratisation du régime général
		$MajTauxRGMax	Majoration du taux maximal de la pension RG
		$SurcoteFP	Surcote fonction publique
		$SurcoteRG1	Coefficient de surcote du régime général, 1ere année
		$SurcoteRG2	Coefficient de surcote du régime général, 2eme année
		$SurcoteRG3	Coefficient de surcote du régime général, 3eme année
		$TauxPleinRG	Taux plein du régime général
		$TauxPleinRG	Taux plein du régime général
		$TauxRGMax	Taux maximal de la pension RG
		$TauxRGMax	Taux maximal de la pension RG
Paramètres démographiques	Lus par DefVarRetr (fichier ParamMortal.xls)	$quotient[$s][$t][$a]	Quotients de mortalité par sexe, date et âge
		$survie[$s][$t][$a]	Survies par sexe, date et âge
Variables individuelles permanentes	Liq (OutilsDroits)	$ageliq[$i]	Age de la liquidation
	Relec (OutilsRetr)	$age[$i]	Age courant
		$anaiss[$i]	Année de naissance
		$conjoint[$i]	Identifiant du conjoint
		$deces[$i]	Indicatrice de décès l'année en cours
		$enf[$e][$i]	Identifiant du e-ième enfant
		$findet[$i]	Age de fin d'études
		$gamma[$i]	Aversion pour le risque
		$k[$i]	Préférence pour le loisir/pénibilité du travail
		$matri[$i]	Statut matrimonial

Type de variable	Mode de calcul ou d'initialisation	Variable	Contenu
		$mere[$i]	Identifiant de la mère
		$migrant[$i]	Indique si l'individu est un migrant de l'année
		$pere[$i]	Identifiant du père
		$present[$i]	Indicatrice de présence dans la population à la date courante
		$salaire[$i]	Salaire courant (équivaut à $salaire_[$i][$age[$i]])
		$salaire_[$i][$a]	Salaire par âge
		$seuil[$i]	Seuil de déclenchement du report du départ en retraite
		$sexe[$i]	Sexe
		$statut[$i]	Statut courant (équivaut à $statut_[$i][$age[$i]])
		$statut_[$i][$a]	Statut d'emploi par âge
		$taux[$i]	Préférence pour le présent
		$taux_prim[$i]	Taux de prime dans la fonction publique
	SimDer (OutilsRetr), Revalo (OutilsDroits)	$rev[$i]	Réversion totale (somme des cinq précédentes)
		$rev_ag[$i]	Reversion agirc
		$rev_ar[$i]	Reversion arrco
		$rev_cn_fp[$i]	Reversions comptes notionels secteur public
		$rev_cn_pri[$i]	Reversions comptes notionels secteur privé
		$rev_fp[$i]	Réversion fonction publique
Variables individuelles permanentes	SimDer (OutilsRetr), Revalo (OutilsDroits)	$rev_in[$i]	Réversion d'indépendant
		$rev_rg[$i]	Réversion régime général
	SimDir (OutilsRetr), Liq et Revalo (OutilsDroits)	$indic_mc[$i]	Indicatrice d'octroi du minimum contributif
		$indic_mg[$i]	Indicatrice d'octroi du minimum garanti de la FP
		$pension[$i]	Pension directe totale
		$pension_ag[$i]	Pension agirc
		$pension_ar[$i]	Pension arrco
		$pension_cn_fp[$i]	Pension comptes notionnels secteur public
		$pension_cn_pri[$i]	Pension comptes notionnels secteur privé
		$pension_fp[$i]	Pension fonction publique
		$pension_in[$i]	Pension d'indépendant
		$pension_rg[$i]	Pension régime général
	SimMinVieil (OutilsRetr)	$min_vieil[$i]	Allocation du minimum vieillesse
Variables individuelles temporaires	Clone (DefVarRetr)	$id_clone	Identifiant de l'individu de départ lorsqu'on travaille sur un clone
	Simdir (OutilsRetr), Liq et DurMajo (OutilsDroits)	$duree_fp_maj	Durée FP incluant MDA éventuelle (selon options)

Type de variable	Mode de calcul ou d'initialisation	Variable	Contenu
		$duree_in_maj	Durée cotisée comme indépendant incluant MDA éventuelle (selon options)
		$duree_rg_maj	Durée RG incluant AVPF et MDA (selon options)
		$duree_tot_maj	Durée cotisée totale incluant AVPF et MDA (selon options)
	SimDir (OutilsRetr), Liq et MinCont (OutilsDroits)	$min_cont	Minimum contributif
	SimDir (OutilsRetr), Liq et MinGaranti (OutilsDroits)	$min_garanti	Minimum garanti
	Simdir (OutilsRetr), Liq et Points (OutilsDroits)	$points_agirc	Cumul de points AGIRC
		$points_arrco	Cumul de points ARRCO
		$points_cn_fp[$i]	Cumul points comptes notionnels du secteur public
		$points_cn_pri[$i]	Cumul points compte notionnel du secteur privé
	Simdir (OutilsRetr), Liq et SalBase (OutilsDroits)	$sam_in	SAM pour le calcul de la pension d'indépendant
		$sam_rg	SAM servant au calcul de la pension RG
		$sr_fp	Salaire de référence pour le calcul de la pension FP
	SimDir(OutilsRetr), Liq et DurBase (OutilsDroits)	$duree_avpf	Durée passée en AVPF
		$duree_cho	Nombre d'années au chômage
		$duree_emp	Nombre d'années en emploi
Variables individuelles temporaires	SimDir(OutilsRetr), Liq et DurBase (OutilsDroits)	$duree_fp	Durée cotisée dans les régimes du secteur public
		$duree_fpa	Durée cotisée dans les régimes du secteur public, catégorie active
		$duree_fps	Durée cotisée dans les régimes du secteur public, catégorie sédentaire
		$duree_in	Durée cotisée comme indépendant
		$duree_PR	Nombre d'années en préretraite
		$duree_rg	Durée cotisée au régime général
		$duree_tot	Durée cotisée totale

Annexe 3 : liste des procédures, classées par bibliothèque et fonction

Note de lecture :
- Le tableau indique les types d'arguments des différents programmes.
- Les noms sont indicatifs du type d'information attendu ou retourné : par exemple `$i ou $j` pour des numéros d'individus, `$a` pour un âge, `$t` pour une date, `$val` pour les autres types de scalaires, `@var` pour un vecteur de données individuelles, `@liste` pour une liste d'indentifiants, `@tab` pour les autres types de tableaux, `%hach` pour un hachage….
- Lorsque le programme attend un argument de type sous programme (**sub**`{ }`), il s'agit systématiquement d'un filtre, i.e. d'une condition sur les caractéristiques de l'individu d'indice `$_`, comme dans `count(`**sub**`{In($statut[$_],@CodesAct)})`.
- Un argument présenté comme une chaine de caractère segmentée par des `/` correspond à une série d'options à passer, en format libre, au sein d'une chaine de caractère unique, comme dans `UseOpt("TP - NoMG et NoBonif.")`.

Nom de la fonction	Objectif	Types d'arguments		Type de retour
		Obligatoires	Optionnels	
DefVarRetr				
`Clone`	Duplication d'un individu	`$i`	`$j, "present"`	
`Delete`	Suppresssion d'un individu	`$i`		
OutilsBase				
Initialisations				
`Init`	Mise à zéro d'une variable individuelle	`@var`		
`UseVRef`	Définition de la variable de référence	`@var,$valeur`		
Opérations élémentaires				
`In`	Test appartenance à un ensemble de valeurs	`$valeur,$val_1,$val_2,$val_3…`		`$booleen`
`Out`	Test de non appartenance	`$valeur,$val_1,$val_2,$val_3…`		`$booleen`
`Min`	Minimum d'un ensemble quelconque de valeurs	`$val_1,$val_2,$val_3…`		`$valeur`
`Max`	Maximum d'un ensemble quelconque de valeurs	`$val_1,$val_2,$val_3…`		`$valeur`
`IMin`	Indice du premier élément non vide d'un tableau	`$val_1,$val_2,$val_3…`		`$entier`
`IMax`	Indice du dernier élément d'un tableau	`$val_1,$val_2,$val_3…`		`$entier`
`Arr`	Arrondi d'un scalaire à un nombre donné de décimales	`$valeur_in,$n_dec`		`$valeur_out`
`ArrAlea`	Arrondi aléatoire	`$valeur_in,$n_dec`		`$valeur_out`
`Pointage`	Affichage de l'heure	`$texte`	`FICHIER`	
Valeurs de fonctions-type				
`Esc`	Valeur en x d'une fonction en escalier	`$x,$x0,$y0,$x1,$y1…`		`$valeur`
`Affn`	Valeur en x d'une fonction affine par morceaux	`$x,$x0,$y0,$x1,$y1…`		`$valeur`
`Sigm`	Valeur en x d'une fonction sigmoide	`$x,$x0,$y0,$x1,$y1…`		`$valeur`
`Part`	Part de x comprise entre deux seuils	`$x,$plancher,$plafond`		`$valeur`
`MinMax`	Plancher si x<plancher, plafond si x>plafond, x	`$x,$plancher,$plafond`		`$valeur`

Nom de la fonction	Objectif	Types d'arguments		Type de retour
		Obligatoires	Optionnels	
	entre les deux			
Redr	Transformation monotone d'une variable x sur support [0,1]	$proba		$proba
Tirages de variables aléatoires				
Alea	Tirage pseudo-aleatoire dans loi [0,1]			$valeur
Booleen	Tirage booleen	$proba		$booleen
MultiNom	Tirage selon une loi multinomiale	$proba_1,$proba_2,$proba_3…		$valeur
Logist	Tirage selon loi logistique	$moyenne,$ec		$valeur
Pareto	Tirage selon une loi de Pareto	$seuil,$forme		$valeur
LogLogist	Tirage selon une loi log-logistique (définie par deux quelconques de ses quantiles	$seuil_1,$val_1,$seuil_2,$val_2		$valeur
Création et manipulation de variables individuelles et de listes d'identifiants				
Sel	Création liste d'identifiants selon critère déterministe	**sub**{filtre}		@liste
Tirage	Creation liste d'identifiants à partir d'une proba d'inclusion	**sub**{filtre}		
Trunc	Troncation d'une liste à ses n premiers éléments	@liste,$nombre		
Excl	Exclusion d'éléments d'une liste	@liste,**sub**{filtre}		@liste
EltAlea	Tirage d'un élément au hasard dans une liste	@liste		$element
TriAsc	Tri ascendant d'une liste selon valeurs d'une variable	@liste_in,@var		@liste_out
TriDesc	Tri descendant d'une liste selon valeurs d'une variable	@liste_in,@var		@liste_out
TriAlea	Permutation aléatoire d'une liste	@liste_in		@liste_out
TriSect	Permutation d'une liste par section	@liste_in		@liste_out
OutilsComp				
Uinst	Utilité instantanée selon statut courant	$i		$valeur
Uact	Utilité actualisée sur durée de vie restante selon âge de départ anticipé	$i,$age,$ageliq		$valeur
SSW	Somme actualisée des droits en fonction de l'âge de départ anticipé	$i,$age,$ageliq		$valeur
Accrual	Calcule le gain en richesse SS actualisée si report d'un an de la liquidation	$i,$ageliq		$valeur
TestLiq	Simule décision de liquidation à l'âge courant (sept options de comportement)	$i		$booleen
AgeOpt	Calcule âge optimal selon critères SSW ou UAct.	$i		$valeur,$valeur_maximande

OutilsDroits				
Calculs de variables intermédiaires non indicées				
Duree	Calcul d'une durée dans un statut	$i,@liste_statuts		$valeur
DurBase	Calcul des durées de base (durées cotisations et années en AVPF)	$i		$valeur
DurMajo	Calcul des durées majorées des droits familiaux	$i		
AllocChom	Calcul des allocations chômage pour le SAM	$i		
SalBase	Calcul des SAM et du salaire de référence FP	$i		
Points	Calcul des décomptes de points ARRCO et AGIRC	$i		
MinCont	Calcul du minimum contributif	$i		
MinGaranti	Calcul du minimum garanti	$i		
Tests de conditions d'âge				
AgeTrim	Retourne un age trimestrialisé	$i		$valeur
AgeMin	Indique si l'individu a atteint l'âge minimum d'ouverture des droits	$i		$booleen
AgeMax	Indique si l'individu a dépassé l'âge de mise à la retraite d'office	$i		$booleen
TauxPlein	Indique si l'individu a atteint les conditions du taux plein	$i		$booleen
Calculs retraites				
Liq	Calcul des droits complets à la liquidation	$i		
Revalo	Revalorisation de l'ensemble des droits (y.c. réversion)	$i		
Indicateurs dérivés				
CotRet	Calcule les cotisations retraite sur salaires	$i	$a	$valeur
CotAut	Calcule la somme des autres cotisations sur salaires	$i	$a	$valeur
CSGSal	Calcule la CSG/CRDS sur salaires	$i	$a	$valeur
CSGRet	Calcule la CSG/CRDS sur pensions	$i		$valeur
SalNet	Calcule le salaire net	$i	$a	$valeur
PNet	Calcule la pension nette	$i		$valeur
SalMoy	Calcule la moyenne des n derniers salaires	$i,"brut/net"	$n,@serie_revalo	$valeur
SalCum	Calcule la somme des n derniers salaires	$i,"brut/net"	$n,@serie_revalo	$valeur
Pente	Calcule un indice de pente de la carrière	$i,"brut/net"	$n,@serie_revalo	$valeur
TauxRemp	Calcule un taux de remplacement rapporté aux n derniers salaires	$i,"brut/net"	$n,@serie_revalo	$valeur
TauxAnn	Calcule un taux d'annuité apparent	$i,"brut/net"	$n,@serie_revalo	$valeur

OutilsExcel				
Procédure de gestion générale de classeurs Excel				
ClassIn	Ouvre un classeur en lecture	"nom fichier"		$classeur
ClassOut	Ouvre un nouveau classeur en écriture	"nom fichier"		$classeur
CloseOut	Ferme un classeur ouvert en écriture	$classeur		
ListClass	Liste des classeurs d'un repertoire et leur contenu	"nom repertoire"		@liste_classeurs
ListFeuil	Liste des feuilles d'un classeur ouvert en entree	$classeur		@liste_feuilles
Lecture de données				
GetAP	Lecture d'un tableau par age et période	$classeur,"feuille"		@tab
GetSer	Lecture d'une série temporelle	$classeur,"feuille",$colonne		@tab
GetPro	Lecture d'un profil par age	$classeur,"feuille",$ligne		@tab
GetH	Lecture d'un hachage simple	$classeur,"feuille",$ligne		%tab
GetHH	Lecture d'un double hachage	$classeur,"feuille",$ligne		%tab
Ecriture de données				
SavAP	Sauvegarde d'un tableau par age et période	$classeur,"feuille","titre",@tab		Création feuille Excel
SavPro	Sauvegarde d'un ensemble de profils par age	$classeur, "feuille","titre","nomvar1",@var1	"nomvar2",@var2, "nomvar3",@var3…	Création feuille Excel
SavSer	Sauvegarde d'un ensemble de série temporelle	$classeur, "feuille","titre","nomvar1",@var1	"nomvar2",@var3, "nomvar3",@var3…	Création feuille Excel
SavHS	Sauvegarde d'un hachage de series temporelles	$classeur,"feuille","titre",%hach		Création feuille Excel
SavSH	Sauvegarde d'une série temporelle de hachages	$classeur,"feuille","titre",@tab		Création feuille Excel
SavHH	Sauvegarde d'un hachage de hachages	$classeur,"feuille","titre",%hach		Création feuille Excel
OutilsMen				
Fonctions démographiques				
ACharg	Indique si l'individu est un enfant à charge	$i		$booleen
PersRef	Retourne la personne de référence du ménage d'appartenance	$i		$j
EcartAge	Calcule l'écart d'âge entre l'individu et son conjoint	$i		$valeur
NbEnf	Nombre d'enfants	$i	[a0..a1] ou "tous"	$valeur
NbEnfC	Nombre d'enfants à charge	$i	[a0..a1]	$valeur
TailMen	Taille du ménage	$i		$valeur
ListEnf	Liste des identifiants des enfants	$i	[a0..a1] ou "tous"	@liste
ListEnfC	Liste des identifiants des enfants à charge	$i	[a0..a1]	@liste
ListMen	Liste des identifiants des membres du ménage	$i		@liste
Fonctions de calcul du niveau de vie				
NbUC	Nombre d'unités de consommation	$i		$valeur
RevMen	Revenu global du ménage	$i		$valeur
NVMen	Niveau de vie du ménage	$i		$valeur

OutilsRetr				
Programmes précisant les hypothèses de simulation des retraites				
UseBios	Définition des fichiers dans lesquels sont récupérées les biographies	"repertoire", $annee_depart, $taux_sondage		
UseOpt	Choix des options générales de simulation	"Exo/pmin/TP/Uinst/Uact/SSW/Accrual /NoMDA/NoBonif/NoAVFP/ /NoMC/NoMG NoAssim/CN"	@cible (pour options Exo et pmin)	
UseOptCN	Choix des options comptes notionnels, s'il y a lieu	$t_0,"RG/base/etendu/global /progressif/immediat"		
UseLeg	Choix de la législation en projection	$date,$generation		
UseLegRetroMax	Choix de la législation pour les reconstitutions initiales	$date		
UseConv	Choix des coefficients de conversion si comptes notionnels	$a_deb,$a_fin,"EV",$t ou $a_deb,$a_fin,$val_deb…,$val_fin		
Programme de récupération des résultats de l'étape Bios				
Relec	Relecture des données initiales ou projetées	$t		
Programmes d'initialisation ou de projection des pensions				
SimDir	Initialisation ou projection des droits directs	$i		
SimDer	Initialisation ou projection des droits dérivés	$i		
SimMinVieil	Initialisation ou projection des droits directs	$i		
Programmes d'affichage des données individuelles (surtout utilisés par l'explorateur)				
ShowBio	Affichage de la carrière complète	$i	"option"	
ShowPens	Affichage des différents éléments de la pension pour un départ à age donné	$i	"option"	
OutilsTab				
Initialisation				
UseW	Définit poids pour calculs des effectifs et des masses	$poids_ind,$poids_fin		
Tabulations générales simples				
SVar	Somme des valeurs d'une variable (avec possibilité de filtrage)	@var,**sub**{filtre}		$valeur
MVar	Moyenne des valeurs d'une variable (avec possibilité de filtrage)	@var,**sub**{filtre}		$valeur
QVar	Quantiles (au choix) d'une variable (avec possibilité de filtrage)	@var,**sub**{filtre}		$valeur
EVar	Ecart-type d'une variable	@var,**sub**{filtre}		$valeur
Count	Comptage nombre d'identifiants vérifiant filtre	**sub**{filtre}		$valeur
Taux	Calcul d'un taux	**sub**{filtre 1}	**sub**{filtre 2}	$valeur
Ratio	Calcul d'un ratio	**sub**{filtre 1}	**sub**{filtre 2}	$valeur
Distrib	Distribution d'une variable discrète	@var	**sub**{filtre}	%tab
Raccourcis pour tabulations spécifiques				
SSal	Somme des salaires	"option"	**sub** {filtre}	$valeur
MSal	Salaire moyen	"option"	**sub** {filtre}	$valeur
SPens	Somme des pensions	"option"	**sub** {filtre}	$valeur

MPens	Pension moyenne	"option"	**sub** {filtre}	$valeur
Tabulations par âge				
UseGrAge	Definit les classes d'âge	$agedeb,$agefin	$pas	
GrAge	Calcule le groupe d'âge auquel appartient l'individu $i	$i		$age ou "mm-nn"
MAge	Moyenne par âge des valeurs d'une variable	@var,**sub**{filtre}		@tab ou %tab
CountAge	Comptage nombre d'identifiants par âge vérifiant condition de filtrage	**sub**{filtre}		@tab ou %tab
TauxAge	Calcul d'un taux par âge	**sub**{filtre1}	**sub**{filtre2}	@tab ou %tab
Tabulations par génération				
UseGrGen	Définit les classes de générations	$genedeb,$genefin	$pas	
GrGen	Calcule le groupe de génération auquel appartient l'individu $i	$i		$gene ou "mm-nn"
MGen	Moyenne par génération des valeurs d'une variable	@var,**sub**{filtre}		@tab ou %tab
CountGen	Comptage nombre d'identifiants par génération vérifiant condition de filtrage	**sub**{filtre}		@tab ou %tab
TauxGen	Calcul d'un taux par génération	**sub**{filtre1}	**sub**{filtre2}	@tab ou %tab
Autres manipulations de tableaux				
Coupe	Extraction d'une partie d'un tableau à double indice (y.c. coupes diagonales)	@tab_in,[n..m]		@tab_out
STab	Somme des valeurs des cases d'un tableau à une dimension, ou de produits de cases de deux tableaux ou plus	@tab1,[n1..m1]	@tab2,[n2..m2], @tab3,[n3..m3]…	$valeur
MTab	Moyenne des valeurs	@tab1,[n1..m1]	@tab2,[n2..m2], @tab3,[n3..m3]…	$valeur
SAct	Idem STab avec actualisation	$taux,@tab1,[n1..m1]	@tab2,[n2..m2], @tab3,[n3..m3]…	$valeur
TRI	Calcul d'un taux de rendement interne associé à un tableau simple	@tab_in,[n..m]		$valeur

G 9001 J. FAYOLLE et M. FLEURBAEY
Accumulation, profitabilité et endettement des entreprises

G 9002 H. ROUSSE
Détection et effets de la multicolinéarité dans les modèles linéaires ordinaires - Un prolongement de la réflexion de BELSLEY, KUH et WELSCH

G 9003 P. RALLE et J. TOUJAS-BERNATE
Indexation des salaires : la rupture de 1983

G 9004 D. GUELLEC et P. RALLE
Compétitivité, croissance et innovation de produit

G 9005 P. RALLE et J. TOUJAS-BERNATE
Les conséquences de la désindexation. Analyse dans une maquette prix-salaires

G 9101 Equipe AMADEUS
Le modèle AMADEUS - Première partie - Présentation générale

G 9102 J.L. BRILLET
Le modèle AMADEUS - Deuxième partie - Propriétés variantielles

G 9103 D. GUELLEC et P. RALLE
Endogenous growth and product innovation

G 9104 H. ROUSSE
Le modèle AMADEUS - Troisième partie - Le commerce extérieur et l'environnement international

G 9105 H. ROUSSE
Effets de demande et d'offre dans les résultats du commerce extérieur manufacturé de la France au cours des deux dernières décennies

G 9106 B. CREPON
Innovation, taille et concentration : causalités et dynamiques

G 9107 B. AMABLE et D. GUELLEC
Un panorama des théories de la croissance endogène

G 9108 M. GLAUDE et M. MOUTARDIER
Une évaluation du coût direct de l'enfant de 1979 à 1989

G 9109 P. RALLE et alii
France - Allemagne : performances économiques comparées

G 9110 J.L. BRILLET
Micro-DMS ***NON PARU***

G 9111 A. MAGNIER
Effets accélérateur et multiplicateur en France depuis 1970 : quelques résultats empiriques

G 9112 B. CREPON et G. DUREAU
Investissement en recherche-développement : analyse de causalités dans un modèle d'accélérateur généralisé

G 9113 J.L. BRILLET, H. ERKEL-ROUSSE, J. TOUJAS-BERNATE
"France-Allemagne Couplées" - Deux économies vues par une maquette macro-économétrique

G 9201 W.J. ADAMS, B. CREPON, D. ENCAOUA
Choix technologiques et stratégies de dissuasion d'entrée

G 9202 J. OLIVEIRA-MARTINS,
J. TOUJAS-BERNATE
Macro-economic import functions with imperfect competition - An application to the E.C. Trade

G 9203 I. STAPIC
Les échanges internationaux de services de la France dans le cadre des négociations multilatérales du GATT
Juin 1992 (1ère version)
Novembre 1992 (version finale)

G 9204 P. SEVESTRE
L'économétrie sur données individuelles-temporelles. Une note introductive

G 9205 H. ERKEL-ROUSSE
Le commerce extérieur et l'environnement international dans le modèle AMADEUS (réestimation 1992)

G 9206 N. GREENAN et D. GUELLEC
Coordination within the firm and endogenous growth

G 9207 A. MAGNIER et J. TOUJAS-BERNATE
Technology and trade : empirical evidences for the major five industrialized countries

G 9208 B. CREPON, E. DUGUET, D. ENCAOUA et P. MOHNEN
Cooperative, non cooperative R & D and optimal patent life

G 9209 B. CREPON et E. DUGUET
Research and development, competition and innovation : an application of pseudo maximum likelihood methods to Poisson models with heterogeneity

G 9301 J. TOUJAS-BERNATE
Commerce international et concurrence imparfaite : développements récents et implications pour la politique commerciale

G 9302 Ch. CASES
Durées de chômage et comportements d'offre de travail : une revue de la littérature

G 9303 H. ERKEL-ROUSSE
Union économique et monétaire : le débat économique

G 9304 N. GREENAN - D. GUELLEC /
G. BROUSSAUDIER - L. MIOTTI
Innovation organisationnelle, dynamisme technologique et performances des entreprises

G 9305 P. JAILLARD
Le traité de Maastricht : présentation juridique et historique

G 9306 J.L. BRILLET
Micro-DMS : présentation et propriétés

G 9307 J.L. BRILLET
Micro-DMS - variantes : les tableaux

G 9308 S. JACOBZONE
Les grands réseaux publics français dans une perspective européenne

G 9309 L. BLOCH - B. CŒURE
Profitabilité de l'investissement productif et transmission des chocs financiers

G 9310 J. BOURDIEU - B. COLIN-SEDILLOT
Les théories sur la structure optimal du capital : quelques points de repère

G 9311 J. BOURDIEU - B. COLIN-SEDILLOT
Les décisions de financement des entreprises françaises : une évaluation empirique des théories de la structure optimale du capital

G 9312 L. BLOCH - B. CŒURÉ
Q de Tobin marginal et transmission des chocs financiers

G 9313 Equipes Amadeus (INSEE), Banque de France, Métric (DP)
Présentation des propriétés des principaux modèles macroéconomiques du Service Public

G 9314 B. CREPON - E. DUGUET
Research & Development, competition and innovation

G 9315 B. DORMONT
Quelle est l'influence du coût du travail sur l'emploi ?

G 9316 D. BLANCHET - C. BROUSSE
Deux études sur l'âge de la retraite

G 9317 D. BLANCHET
Répartition du travail dans une population hétérogène : deux notes

G 9318 D. EYSSARTIER - N. PONTY
AMADEUS - an annual macro-economic model for the medium and long term

G 9319 G. CETTE - Ph. CUNÉO - D. EYSSARTIER - J. GAUTIÉ
Les effets sur l'emploi d'un abaissement du coût du travail des jeunes

G 9401 D. BLANCHET
Les structures par âge importent-elles ?

G 9402 J. GAUTIÉ
Le chômage des jeunes en France : problème de formation ou phénomène de file d'attente ? Quelques éléments du débat

G 9403 P. QUIRION
Les déchets en France : éléments statistiques et économiques

G 9404 D. LADIRAY - M. GRUN-REHOMME
Lissage par moyennes mobiles - Le problème des extrémités de série

G 9405 V. MAILLARD
Théorie et pratique de la correction des effets de jours ouvrables

G 9406 F. ROSENWALD
La décision d'investir

G 9407 S. JACOBZONE
Les apports de l'économie industrielle pour définir la stratégie économique de l'hôpital public

G 9408 L. BLOCH, J. BOURDIEU,
B. COLIN-SEDILLOT, G. LONGUEVILLE
Du défaut de paiement au dépôt de bilan : les banquiers face aux PME en difficulté

G 9409 D. EYSSARTIER, P. MAIRE
Impacts macro-économiques de mesures d'aide au logement - quelques éléments d'évaluation

G 9410 F. ROSENWALD
Suivi conjoncturel de l'investissement

G 9411 C. DEFEUILLEY - Ph. QUIRION
Les déchets d'emballages ménagers : une analyse économique des politiques française et allemande

G 9412 J. BOURDIEU - B. CŒURÉ -
B. COLIN-SEDILLOT
Investissement, incertitude et irréversibilité
Quelques développements récents de la théorie de l'investissement

G 9413 B. DORMONT - M. PAUCHET
L'évaluation de l'élasticité emploi-salaire dépend-elle des structures de qualification ?

G 9414 I. KABLA
Le Choix de breveter une invention

G 9501 J. BOURDIEU - B. CŒURÉ - B. SEDILLOT
Irreversible Investment and Uncertainty :
When is there a Value of Waiting ?

G 9502 L. BLOCH - B. CŒURÉ
Imperfections du marché du crédit, investissement des entreprises et cycle économique

G 9503 D. GOUX - E. MAURIN
Les transformations de la demande de travail par qualification en France
Une étude sur la période 1970-1993

G 9504 N. GREENAN
Technologie, changement organisationnel, qualifications et emploi : une étude empirique sur l'industrie manufacturière

G 9505 D. GOUX - E. MAURIN
Persistance des hiérarchies sectorielles de salaires: un réexamen sur données françaises

G 9505 Bis D. GOUX - E. MAURIN
Persistence of inter-industry wages differentials: a reexamination on matched worker-firm panel data

G 9506 S. JACOBZONE
Les liens entre RMI et chômage, une mise en perspective
NON PARU - article sorti dans Economie et Prévision n°122 (1996) - pages 95 à 113

G 9507 G. CETTE - S. MAHFOUZ
Le partage primaire du revenu
Constat descriptif sur longue période

G 9601 Banque de France - CEPREMAP - Direction de la Prévision - Erasme - INSEE - OFCE
Structures et propriétés de cinq modèles macroéconomiques français

G 9602 Rapport d'activité de la DESE de l'année 1995

G 9603 J. BOURDIEU - A. DRAZNIEKS
L'octroi de crédit aux PME : une analyse à partir d'informations bancaires

G 9604 A. TOPIOL-BENSAÏD
Les implantations japonaises en France

G 9605 P. GENIER - S. JACOBZONE
Comportements de prévention, consommation d'alcool et tabagie : peut-on parler d'une gestion globale du capital santé ?
Une modélisation microéconométrique empirique

G 9606 C. DOZ - F. LENGLART
Factor analysis and unobserved component models: an application to the study of French business surveys

G 9607 N. GREENAN - D. GUELLEC
La théorie coopérative de la firme

G 9608 N. GREENAN - D. GUELLEC
Technological innovation and employment reallocation

G 9609 Ph. COUR - F. RUPPRECHT
L'intégration asymétrique au sein du continent américain : un essai de modélisation

G 9610 S. DUCHENE - G. FORGEOT - A. JACQUOT
Analyse des évolutions récentes de la productivité apparente du travail

G 9611 X. BONNET - S. MAHFOUZ
The influence of different specifications of wages-prices spirals on the measure of the NAIRU : the case of France

G 9612 PH. COUR - E. DUBOIS, S. MAHFOUZ, J. PISANI-FERRY
The cost of fiscal retrenchment revisited: how strong is the evidence ?

G 9613 A. JACQUOT
Les flexions des taux d'activité sont-elles seulement conjoncturelles ?

G 9614 ZHANG Yingxiang - SONG Xueqing
Lexique macroéconomique Français-Chinois

G 9701 J.L. SCHNEIDER
La taxe professionnelle : éléments de cadrage économique

G 9702 J.L. SCHNEIDER
Transition et stabilité politique d'un système redistributif

G 9703 D. GOUX - E. MAURIN
Train or Pay: Does it Reduce Inequalities to Encourage Firms to Train their Workers?

G 9704 P. GENIER
Deux contributions sur dépendance et équité

G 9705 E. DUGUET - N. IUNG
R & D Investment, Patent Life and Patent Value
An Econometric Analysis at the Firm Level

G 9706 M. HOUDEBINE - A. TOPIOL-BENSAÏD
Les entreprises internationales en France : une analyse à partir de données individuelles

G 9707 M. HOUDEBINE
Polarisation des activités et spécialisation des départements en France

G 9708 E. DUGUET - N. GREENAN
Le biais technologique : une analyse sur données individuelles

G 9709 J.L. BRILLET
Analyzing a small French ECM Model

G 9710 J.L. BRILLET
Formalizing the transition process : scenarios for capital accumulation

G 9711 G. FORGEOT - J. GAUTIÉ
Insertion professionnelle des jeunes et processus de déclassement

G 9712 E. DUBOIS
High Real Interest Rates: the Consequence of a Saving Investment Disequilibrium or of an insufficient Credibility of Monetary Authorities?

G 9713 Bilan des activités de la Direction des Etudes et Synthèses Economiques - 1996

G 9714 F. LEQUILLER
Does the French Consumer Price Index Overstate Inflation?

G 9715 X. BONNET
Peut-on mettre en évidence les rigidités à la baisse des salaires nominaux ?
Une étude sur quelques grands pays de l'OCDE

G 9716 N. IUNG - F. RUPPRECHT
Productivité de la recherche et rendements d'échelle dans le secteur pharmaceutique français

G 9717 E. DUGUET - I. KABLA
Appropriation strategy and the motivations to use the patent system in France - An econometric analysis at the firm level

G 9718 L.P. PELÉ - P. RALLE
Âge de la retraite : les aspects incitatifs du régime général

G 9719 ZHANG Yingxiang - SONG Xueqing
Lexique macroéconomique français-chinois, chinois-français

G 9720 M. HOUDEBINE - J.L. SCHNEIDER
Mesurer l'influence de la fiscalité sur la localisation des entreprises

G 9721 A. MOUROUGANE
Crédibilité, indépendance et politique monétaire
Une revue de la littérature

G 9722 P. AUGERAUD - L. BRIOT
Les données comptables d'entreprises
Le système intermédiaire d'entreprises
Passage des données individuelles aux données sectorielles

G 9723 P. AUGERAUD - J.E. CHAPRON
Using Business Accounts for Compiling National Accounts: the French Experience

G 9724 P. AUGERAUD
Les comptes d'entreprise par activités - Le passage aux comptes - De la comptabilité d'entreprise à la comptabilité nationale - *A paraître*

G 9801 H. MICHAUDON - C. PRIGENT
Présentation du modèle AMADEUS

G 9802 J. ACCARDO
Une étude de comptabilité générationnelle pour la France en 1996

G 9803 X. BONNET - S. DUCHÊNE
Apports et limites de la modélisation « Real Business Cycles »

G 9804 C. BARLET - C. DUGUET - D. ENCAOUA - J. PRADEL
The Commercial Sucess of Innovations
An econometric analysis at the firm level in French manufacturing

G 9805 P. CAHUC - Ch. GIANELLA - D. GOUX - A. ZILBERBERG
Equalizing Wage Differences and Bargaining Power - Evidence form a Panel of French Firms

G 9806 J. ACCARDO - M. JLASSI
La productivité globale des facteurs entre 1975 et 1996

G 9807 Bilan des activités de la Direction des Etudes et Synthèses Economiques - 1997

G 9808 A. MOUROUGANE
Can a Conservative Governor Conduct an Accomodative Monetary Policy ?

G 9809 X. BONNET - E. DUBOIS - L. FAUVET
Asymétrie des inflations relatives et menus costs : tests sur l'inflation française

G 9810 E. DUGUET - N. IUNG
Sales and Advertising with Spillovers at the firm level: Estimation of a Dynamic Structural Model on Panel Data

G 9811 J.P. BERTHIER
Congestion urbaine : un modèle de trafic de pointe à courbe débit-vitesse et demande élastique

G 9812 C. PRIGENT
La part des salaires dans la valeur ajoutée : une approche macroéconomique

G 9813 A.Th. AERTS
L'évolution de la part des salaires dans la valeur ajoutée en France reflète-t-elle les évolutions individuelles sur la période 1979-1994 ?

G 9814 B. SALANIÉ
Guide pratique des séries non-stationnaires

G 9901 S. DUCHÊNE - A. JACQUOT
Une croissance plus riche en emplois depuis le début de la décennie ? Une analyse en comparaison internationale

G 9902 Ch. COLIN
Modélisation des carrières dans Destinie

G 9903 Ch. COLIN
Evolution de la dispersion des salaires : un essai de prospective par microsimulation

G 9904 B. CREPON - N. IUNG
Innovation, emploi et performances

G 9905 B. CREPON - Ch. GIANELLA
Wages inequalities in France 1969-1992
An application of quantile regression techniques

G 9906 C. BONNET - R. MAHIEU
Microsimulation techniques applied to intergenerational transfers - Pensions in a dynamic framework: the case of France

G 9907 F. ROSENWALD
L'impact des contraintes financières dans la décision d'investissement

G 9908 Bilan des activités de la DESE - 1998

G 9909 J.P. ZOYEM
Contrat d'insertion et sortie du RMI
Evaluation des effets d'une politique sociale

G 9910 Ch. COLIN - Fl. LEGROS - R. MAHIEU
Bilans contributifs comparés des régimes de retraite du secteur privé et de la fonction publique

G 9911 G. LAROQUE - B. SALANIÉ
Une décomposition du non-emploi en France

G 9912 B. SALANIÉ
Une maquette analytique de long terme du marché du travail

G 9912 Bis Ch. GIANELLA
Une estimation de l'élasticité de l'emploi peu qualifié à son coût

G 9913 Division « Redistribution et Politiques Sociales »
Le modèle de microsimulation dynamique DESTINIE

G 9914 E. DUGUET
Macro-commandes SAS pour l'économétrie des panels et des variables qualitatives

G 9915 R. DUHAUTOIS
Evolution des flux d'emplois en France entre 1990 et 1996 : une étude empirique à partir du fichier des bénéfices réels normaux (BRN)

G 9916 J.Y. FOURNIER
Extraction du cycle des affaires : la méthode de Baxter et King

G 9917 B. CRÉPON - R. DESPLATZ - J. MAIRESSE
Estimating price cost margins, scale economies and workers' bargaining power at the firm level

G 9918 Ch. GIANELLA - Ph. LAGARDE
Productivity of hours in the aggregate production function: an evaluation on a panel of French firms from the manufacturing sector

G 9919 S. AUDRIC - P. GIVORD - C. PROST
Evolution de l'emploi et des coûts par qualification entre 1982 et 1996

G 2000/01 R. MAHIEU
Les déterminants des dépenses de santé : une approche macroéconomique

G 2000/02 C. ALLARD-PRIGENT - H. GUILMEAU - A. QUINET
The real exchange rate as the relative price of nontrables in terms of tradables: theoretical investigation and empirical study on French data

G 2000/03 J.-Y. FOURNIER
L'approximation du filtre passe-bande proposée par Christiano et Fitzgerald

G 2000/04 Bilan des activités de la DESE - 1999

G 2000/05 B. CREPON - F. ROSENWALD
Investissement et contraintes de financement : le poids du cycle
Une estimation sur données françaises

G 2000/06 A. FLIPO
Les comportements matrimoniaux de fait

G 2000/07 R. MAHIEU - B. SÉDILLOT
Microsimulations of the retirement decision: a supply side approach

G 2000/08 C. AUDENIS - C. PROST
Déficit conjoncturel : une prise en compte des conjonctures passées

G 2000/09 R. MAHIEU - B. SÉDILLOT
Equivalent patrimonial de la rente et souscription de retraite complémentaire

G 2000/10 R. DUHAUTOIS
Ralentissement de l'investissement : petites ou grandes entreprises ? industrie ou tertiaire ?

G 2000/11 G. LAROQUE - B. SALANIÉ
Temps partiel féminin et incitations financières à l'emploi

G2000/12 Ch. GIANELLA
Local unemployment and wages

G2000/13 B. CREPON - Th. HECKEL
- Informatisation en France : une évaluation à partir de données individuelles

- Computerization in France: an evaluation based on individual company data

G2001/01 F. LEQUILLER
- La nouvelle économie et la mesure de la croissance du PIB
- The new economy and the measure ment of GDP growth

G2001/02 S. AUDRIC
La reprise de la croissance de l'emploi profite-t-elle aussi aux non-diplômés ?

G2001/03 I. BRAUN-LEMAIRE
Evolution et répartition du surplus de productivité

G2001/04 A. BEAUDU - Th. HECKEL
Le canal du crédit fonctionne-t-il en Europe ? Une étude de l'hétérogénéité des comportements d'investissement à partir de données de bilan agrégées

G2001/05 C. AUDENIS - P. BISCOURP - N. FOURCADE - O. LOISEL
Testing the augmented Solow growth model : An empirical reassessment using panel data

G2001/06 R. MAHIEU - B. SÉDILLOT
Départ à la retraite, irréversibilité et incertitude

G2001/07 Bilan des activités de la DESE - 2000

G2001/08 J. Ph. GAUDEMET
Les dispositifs d'acquisition à titre facultatif d'annuités viagères de retraite

G2001/09 B. CRÉPON - Ch. GIANELLA
Fiscalité, coût d'usage du capital et demande de facteurs : une analyse sur données individuelles

G2001/10 B. CRÉPON - R. DESPLATZ
Evaluation des effets des dispositifs d'allégements de charges sociales sur les bas salaires

G2001/11 J.-Y. FOURNIER
Comparaison des salaires des secteurs public et privé

G2001/12 J.-P. BERTHIER - C. JAULENT - R. CONVENEVOLE - S. PISANI
Une méthodologie de comparaison entre consommations intermédiaires de source fiscale et de comptabilité nationale

G2001/13 P. BISCOURP - Ch. GIANELLA
Substitution and complementarity between capital, skilled and less skilled workers: an analysis at the firm level in the French manufacturing industry

G2001/14 I. ROBERT-BOBEE
Modelling demographic behaviours in the French microsimulation model Destinie: An analysis of future change in completed fertility

G2001/15 J.-P. ZOYEM
Diagnostic sur la pauvreté et calendrier de revenus : le cas du "Panel européen des ménages »

G2001/16 J.-Y. FOURNIER - P. GIVORD
La réduction des taux d'activité aux âges extrêmes, une spécificité française ?

G2001/17 C. AUDENIS - P. BISCOURP - N. RIEDINGER
Existe-t-il une asymétrie dans la transmission du prix du brut aux prix des carburants ?

G2002/01 F. MAGNIEN - J.-L. TAVERNIER - D. THESMAR
Les statistiques internationales de PIB par habitant en standard de pouvoir d'achat : une analyse des résultats

G2002/02 Bilan des activités de la DESE - 2001

G2002/03 B. SÉDILLOT - E. WALRAET
La cessation d'activité au sein des couples : y a-t-il interdépendance des choix ?

G2002/04 G. BRILHAULT
- Rétropolation des séries de FBCF et calcul du capital fixe en SEC-95 dans les comptes nationaux français
- Retropolation of the investment series (GFCF) and estimation of fixed capital stocks on the ESA-95 basis for the French balance sheets

G2002/05 P. BISCOURP - B. CRÉPON - T. HECKEL - N. RIEDINGER
How do firms respond to cheaper computers? Microeconometric evidence for France based on a production function approach

G2002/06 C. AUDENIS - J. DEROYON - N. FOURCADE
L'impact des nouvelles technologies de l'information et de la communication sur l'économie française - un bouclage macro-économique

G2002/07 J. BARDAJI - B. SÉDILLOT - E. WALRAET
Évaluation de trois réformes du Régime Général d'assurance vieillesse à l'aide du modèle de microsimulation DESTINIE

G2002/08 J.-P. BERTHIER
Réflexions sur les différentes notions de volume dans les comptes nationaux : comptes aux prix d'une année fixe ou aux prix de l'année précédente, séries chaînées

G2002/09 F. HILD
Les soldes d'opinion résument-ils au mieux les réponses des entreprises aux enquêtes de conjoncture ?

G2002/10 I. ROBERT-BOBÉE
Les comportements démographiques dans le modèle de microsimulation Destinie - Une comparaison des estimations issues des enquêtes Jeunes et Carrières 1997 et Histoire Familiale 1999

G2002/11 J.-P. ZOYEM
La dynamique des bas revenus : une analyse des entrées-sorties de pauvreté

G2002/12 F. HILD
Prévisions d'inflation pour la France

G2002/13 M. LECLAIR
Réduction du temps de travail et tensions sur les facteurs de production

G2002/14 E. WALRAET - A. VINCENT
- Analyse de la redistribution intragénérationnelle dans le système de retraite des salariés du privé - Une approche par microsimulation
- Intragenerational distributional analysis in the french private sector pension scheme - A microsimulation approach

G2002/15 P. CHONE - D. LE BLANC - I. ROBERT-BOBEE
Offre de travail féminine et garde des jeunes enfants

G2002/16 F. MAUREL - S. GREGOIR
Les indices de compétitivité des pays : interprétation et limites

G2003/01 N. RIEDINGER - E.HAUVY
Le coût de dépollution atmosphérique pour les entreprises françaises : Une estimation à partir de données individuelles

G2003/02 P. BISCOURP et F. KRAMARZ
Création d'emplois, destruction d'emplois et internationalisation des entreprises industrielles françaises : une analyse sur la période 1986-1992

G2003/03 Bilan des activités de la DESE - 2002

G2003/04 P.-O. BEFFY - J. DEROYON - N. FOURCADE - S. GREGOIR - N. LAÏB - B. MONFORT
Évolutions démographiques et croissance : une projection macro-économique à l'horizon 2020

G2003/05 P. AUBERT
La situation des salariés de plus de cinquante ans dans le secteur privé

G2003/06 P. AUBERT - B. CRÉPON
Age, salaire et productivité
La productivité des salariés décline-t-elle en fin de carrière ?

G2003/07 H. BARON - P.O. BEFFY - N. FOURCADE - R. MAHIEU
Le ralentissement de la productivité du travail au cours des années 1990

G2003/08 P.-O. BEFFY - B. MONFORT
Patrimoine des ménages, dynamique d'allocation et comportement de consommation

G2003/09 P. BISCOURP - N. FOURCADE
Peut-on mettre en évidence l'existence de rigidités à la baisse des salaires à partir de données individulles ? Le cas de la France à la fin des années 90

G2003/10 M. LECLAIR - P. PETIT
Présence syndicale dans les firmes : quel impact sur les inégalités salariales entre les hommes et les femmes ?

G2003/11 P.-O. BEFFY - X. BONNET - M. DARRACQ-PARIES - B. MONFORT
MZE: a small macro-model for the euro area

G2004/01 P. AUBERT - M. LECLAIR
La compétitivité exprimée dans les enquêtes trimestrielles sur la situation et les perspectives dans l'industrie

G2004/02 M. DUÉE - C. REBILLARD
La dépendance des personnes âgées : une projection à long terme

G2004/03 S. RASPILLER - N. RIEDINGER
Régulation environnementale et choix de localisation des groupes français

G2004/04 A. NABOULET - S. RASPILLER
Les déterminants de la décision d'investir : une approche par les perceptions subjectives des firmes

G2004/05 N. RAGACHE
La déclaration des enfants par les couples non mariés est-elle fiscalement optimale ?

G2004/06 M. DUÉE
L'impact du chômage des parents sur le devenir scolaire des enfants

G2004/07 P. AUBERT - E. CAROLI - M. ROGER
New Techlologies, Workplace Organisation and the Age Structure of the Workforce: Firm-Level Evidence

G2004/08 E. DUGUET - C. LELARGE
Les brevets accroissent-ils les incitations privées à innover ? Un examen microéconométrique

G2004/09 S. RASPILLER - P. SILLARD
Affiliating versus Subcontracting: the Case of Multinationals

G2004/10 J. BOISSINOT - C. L'ANGEVIN - B. MONFORT
Public Debt Sustainability: Some Results on the French Case

G2004/11 S. ANANIAN - P. AUBERT
Travailleurs âgés, nouvelles technologies et changements organisationnels : un réexamen à partir de l'enquête « REPONSE »

G2004/12 X. BONNET - H. PONCET
Structures de revenus et propensions différentes à consommer - Vers une équation de consommation des ménages plus robuste en prévision pour la France

G2004/13 C. PICART
Évaluer la rentabilité des sociétés non financières

G2004/14 J. BARDAJI - B. SÉDILLOT - E. WALRAET
Les retraites du secteur public : projections à l'horizon 2040 à l'aide du modèle de microsimulation DESTINIE

G2005/01 S. BUFFETEAU - P. GODEFROY
Conditions de départ en retraite selon l'âge de fin d'études : analyse prospective pour les générations 1945 à1974

G2005/02 C. AFSA - S. BUFFETEAU
L'évolution de l'activité féminine en France : une approche par pseudo-panel

G2005/03 P. AUBERT - P. SILLARD
Délocalisations et réductions d'effectifs dans l'industrie française

G2005/04 M. LECLAIR - S. ROUX
Mesure et utilisation des emplois instables dans les entreprises

G2005/05 C. L'ANGEVIN - S. SERRAVALLE
Performances à l'exportation de la France et de l'Allemagne - Une analyse par secteur et destination géographique

G2005/06 Bilan des activités de la Direction des Études et Synthèses Économiques - 2004

G2005/07 S. RASPILLER
La concurrence fiscale : principaux enseignements de l'analyse économique

G2005/08 C. L'ANGEVIN - N. LAÏB
Éducation et croissance en France et dans un panel de 21 pays de l'OCDE

G2005/09 N. FERRARI
Prévoir l'investissement des entreprises
Un indicateur des révisions dans l'enquête de conjoncture sur les investissements dans l'industrie.

G2005/10 P.-O. BEFFY - C. L'ANGEVIN
Chômage et boucle prix-salaires :
apport d'un modèle « qualiifés/peu qualifiés »

G2005/11 B. HEITZ
A two-states Markov-switching model of inflation in France and the USA: credible target VS inflation spiral

G2005/12 O. BIAU - H. ERKEL-ROUSSE - N. FERRARI
Réponses individuelles aux enquêtes de conjoncture et prévision macroéconomiques :
Exemple de la prévision de la production manufacturière

G2005/13 P. AUBERT - D. BLANCHET - D. BLAU
The labour market after age 50: some elements of a Franco-American comparison

G2005/14 D. BLANCHET - T. DEBRAND -
P. DOURGNON - P. POLLET
L'enquête SHARE : présentation et premiers résultats de l'édition française

G2005/15 M. DUÉE
La modélisation des comportements démographiques dans le modèle de microsimulation DESTINIE

G2005/16 H. RAOUI - S. ROUX
Étude de simulation sur la participation versée aux salariés par les entreprises

G2006/01 C. BONNET - S. BUFFETEAU - P. GODEFROY
Disparités de retraite de droit direct entre hommes et femmes : quelles évolutions ?

G2006/02 C. PICART
Les gazelles en France

G2006/03 P. AUBERT - B. CRÉPON -P. ZAMORA
Le rendement apparent de la formation continue dans les entreprises : effets sur la productivité et les salaires

G2006/04 J.-F. OUVRARD - R. RATHELOT
Demographic change and unemployment:
what do macroeconometric models predict?

G2006/05 D. BLANCHET - J.-F. OUVRARD
Indicateurs d'engagements implicites des systèmes de retraite : chiffrages, propriétés analytiques et réactions à des chocs démographiques types

G2006/06 G. BIAU - O. BIAU - L. ROUVIERE
Nonparametric Forecasting of the Manufacturing Output Growth with Firm-level Survey Data

G2006/07 C. AFSA - P. GIVORD
Le rôle des conditions de travail dans les absences pour maladie

G2006/08 P. SILLARD - C. L'ANGEVIN - S. SERRAVALLE
Performances comparées à l'exportation de la France et de ses principaux partenaires
Une analyse structurelle sur 12 ans

G2006/09 X. BOUTIN - S. QUANTIN
Une méthodologie d'évaluation comptable du coût du capital des entreprises françaises : 1984-2002

G2006/10 C. AFSA
L'estimation d'un coût implicite de la pénibilité du travail chez les travailleurs âgés

G2006/11 C. LELARGE
Les entreprises (industrielles) françaises sont-elles à la frontière technologique ?

G2006/12 O. BIAU - N. FERRARI
Théorie de l'opinion
Faut-il pondérer les réponses individuelles ?

G2006/13 A. KOUBI - S. ROUX
Une réinterprétation de la relation entre productivité et inégalités salariales dans les entreprises

G2006/14 R. RATHELOT - P. SILLARD
The impact of local taxes on plants location decision

G2006/15 L. GONZALEZ - C. PICART
Diversification, recentrage et poids des activités de support dans les groupes (1993-2000)

G2007/01 D. SRAER
Allègements de cotisations patronales et dynamique salariale

G2007/02 V. ALBOUY - L. LEQUIEN
Les rendements non monétaires de l'éducation : le cas de la santé

G2007/03 D. BLANCHET - T. DEBRAND
Aspiration à la retraite, santé et satisfaction au travail : une comparaison européenne

G2007/04 M. BARLET - L. CRUSSON
Quel impact des variations du prix du pétrole sur la croissance française ?

G2007/05 C. PICART
Flux d'emploi et de main-d'œuvre en France : un réexamen

G2007/06 V. ALBOUY - C. TAVAN
Massification et démocratisation de l'enseignement supérieur en France

G2007/07 T. LE BARBANCHON
The Changing response to oil price shocks in France : a DSGE type approach

G2007/08 T. CHANEY - D. SRAER - D. THESMAR
Collateral Value and Corporate Investment
Evidence from the French Real Estate Market

G2007/09 J. BOISSINOT
Consumption over the Life Cycle: Facts for France

G2007/10 C. AFSA
Interpréter les variables de satisfaction : l'exemple de la durée du travail

G2007/11 R. RATHELOT - P. SILLARD
Zones Franches Urbaines : quels effets sur l'emploi salarié et les créations d'établissements ?

G2007/12 V. ALBOUY - B. CRÉPON
Aléa moral en santé : une évaluation dans le cadre du modèle causal de Rubin

G2008/01 C. PICART
Les PME françaises : rentables mais peu dynamiques

G2008/02 P. BISCOURP - X. BOUTIN - T. VERGÉ
The Effects of Retail Regulations on Prices
Evidence form the Loi Galland

G2008/03 Y. BARBESOL - A. BRIANT
Économies d'agglomération et productivité des entreprises : estimation sur données individuelles françaises

G2008/04 D. BLANCHET - F. LE GALLO
Les projections démographiques : principaux mécanismes et retour sur l'expérience française

G2008/05 D. BLANCHET - F. TOUTLEMONDE
Évolutions démographiques et déformation du cycle de vie active : quelles relations ?

G2008/06 M. BARLET - D. BLANCHET - L. CRUSSON
Internationalisation et flux d'emplois : que dit une approche comptable ?

G2008/07 C. LELARGE - D. SRAER - D. THESMAR
Entrepreurship and Credit Constraints - Evidence from a French Loan Guarantee Program

G2008/08 X. BOUTIN - L. JANIN
Are Prices Really Affected by Mergers?

G2008/09 M. BARLET - A. BRIANT - L. CRUSSON
Concentration géographique dans l'industrie manufacturière et dans les services en France : une approche par un indicateur en continu

G2008/10 M. BEFFY - É. COUDIN - R. RATHELOT
Who is confronted to insecure labor market histories? Some evidence based on the French labor market transition

G2008/11 M. ROGER - E. WALRAET
Social Security and Well-Being of the Elderly: the Case of France

G2008/12 C. AFSA
Analyser les composantes du bien-être et de son évolution
Une approche empirique sur données individuelles

G2008/13 M. BARLET - D. BLANCHET -
T. LE BARBANCHON
Microsimuler le marché du travail : un prototype

G2009/01 P.-A. PIONNIER
Le partage de la valeur ajoutée en France, 1949-2007

G2009/02 Laurent CLAVEL - Christelle MINODIER
A Monthly Indicator of the French Business Climate

G2009/03 H. ERKEL-ROUSSE - C. MINODIER
Do Business Tendency Surveys in Industry and Services Help in Forecasting GDP Growth?
A Real-Time Analysis on French Data

G2009/04 P. GIVORD - L. WILNER
Les contrats temporaires : trappe ou marchepied vers l'emploi stable ?

G2009/05 G. LALANNE - P.-A. PIONNIER - O. SIMON
Le partage des fruits de la croissance de 1950 à 2008 : une approche par les comptes de surplus

G2009/06 L. DAVEZIES - X. D'HAULTFOEUILLE
Faut-il pondérer ?… Ou l'éternelle question de l'économètre confronté à des données d'enquête

G2009/07 S. QUANTIN - S. RASPILLER - S. SERRAVALLE
Commerce intragroupe, fiscalité et prix de transferts : une analyse sur données françaises

G2009/08 M. CLERC - V. MARCUS
Élasticités-prix des consommations énergétiques des ménages

G2009/09 G. LALANNE - E. POULIQUEN - O. SIMON
Prix du pétrole et croisssance potentielle à long terme

G2009/10 D. BLANCHET - J. LE CACHEUX - V. MARCUS
Adjusted net savings and other approaches to sustaibability: some therotical backgroud

G2009/11 V. BELLAMY - G. CONSALES - M. FESSEAU -
S. LE LAIDIER - É. RAYNAUD
Une décomposition du compte des ménages de la comptabilité nationale par catégorie de ménage en 2003

G2009/12 J. BARDAJI - F. TALLET
Detecting Economic Regimes in France : a Qualitative Markov-Switching Indicator Using Mixed Frequency Data

G2009/13 R. AEBERHARDT - D. FOUGÈRE -
R. RATHELOT
Discrimination à l'embauche : comment exploiter les procédures de *testing* ?

G2009/14 Y. BARBESOL - P. GIVORD - S. QUANTIN
Partage de la valeur ajoutée, approche par données microéconomiques

G2009/15 I. BUONO - G. LALANNE
The Effect of the Uruguay round on the Intensive and Extensive Margins of Trade

G2010/01 C. MINODIER
Avantages comparés des séries des premières valeurs publiées et des séries des valeurs révisées - Un exercice de prévision en temps réel de la croissance trimestrielle du PIB en France

G2010/02 V. ALBOUY - L. DAVEZIES - T. DEBRAND
Health Expenditure Models: a Comparison of Five Specifications using Panel Data

G2010/03 C. KLEIN - O. SIMON
Le modèle MÉSANGE réestimé en base 2000
Tome 1 – Version avec volumes à prix constants

G2010/04 M.-É. CLERC - É. COUDIN
L'IPC, miroir de l'évolution du coût de la vie en France ? Ce qu'apporte l'analyse des courbes d'Engel

G2010/05 N. CECI-RENAUD - P.-A. CHEVALIER
Les seuils de 10, 20 et 50 salariés : impact sur la taille des entreprises françaises

G2010/06 R. AEBERHARDT - J. POUGET
National Origin Differences in Wages and Hierarchical Positions - Evidence on French Full-Time Male Workers from a matched Employer-Employee Dataset

G2010/07 S. BLASCO - P. GIVORD
Les trajectoires professionnelles en début de vie active : quel impact des contrats temporaires ?

G2010/08 P. GIVORD
Méthodes économétriques pour l'évaluation de politiques publiques

G2010/09 P.-Y. CABANNES - V. LAPÈGUE -
E. POULIQUEN - M. BEFFY - M. GAINI
Quelle croissance de moyen terme après la crise ?

G2010/10 I. BUONO - G. LALANNE
La réaction des entreprises françaises
à la baisse des tarifs douaniers étrangers

G2010/11 R. RATHELOT - P. SILLARD
L'apport des méthodes à noyaux pour mesurer la concentration géographique - Application à la concentration des immigrés en France de 1968 à 1999

G2010/12 M. BARATON - M. BEFFY - D. FOUGÈRE
Une évaluation de l'effet de la réforme de 2003 sur les départs en retraite - Le cas des enseignants du second degré public

G2010/13 D. BLANCHET - S. BUFFETEAU - E. CRENNER S. LE MINEZ
Le modèle de microsimulation Destinie 2 : principales caractéristiques et premiers résultats

G2010/14 D. BLANCHET - E. CRENNER
Le bloc retraites du modèle Destinie 2 : guide de l'utilisateur