



Institut des
Politiques Publiques

GUIDE METHODOLOGIQUE IPP – JUIN 2013

Le modèle de micro-simulation TaxIPP – Life La bible de l'utilisateur

Alexis Eidelman





L'Institut des politiques publiques (IPP) est développé dans le cadre d'un partenariat scientifique entre PSE-Ecole d'économie de Paris (PSE) et le Centre de Recherche en Economie et Statistique (CREST). L'IPP vise à promouvoir l'analyse et l'évaluation quantitatives des politiques publiques en s'appuyant sur les méthodes les plus récentes de la recherche en économie.

www.ipp.eu



z

RÉSUMÉ

Cette note méthodologique décrit le fonctionnement de Taxipp, le modèle de micro-simulation de l'ensemble du système socio-fiscale français sur l'ensemble du cycle de vie développé à l'Institut des politiques publiques (IPP). Le modèle Taxipp est un modèle de micro-simulation dynamique non spécifique qui simule pour un échantillon représentatif de la population française les prestations dans leur définition la plus large et les prélèvements, pour l'instant en mettant de côté les taxes sur les entreprises. Des réformes de la législation peuvent aussi être appliquées.

SOMMAIRE

1	Avant-Propos sur ce document	5
1.1	La bible de TaxIPP-Life	5
1.2	Message aux codeurs	6
2	Introduction	8
2.1	L'ambition de TaxIPP-Life	8
2.1.1	Un modèle de microsimulation généraliste	8
2.1.2	Mesurer les niveaux de vie	9
2.1.3	Mesurer la redistribution	9
2.2	Historique	9
2.2.1	Contexte lors de la création	10
2.2.2	Frise chronologique	10
3	La simulation : la technique	12
3.0.3	Alignement	13
3.0.4	Pondérations	15
3.0.5	Formation des unions	19
3.0.6	Le choix du pas temporel	23
3.0.7	L'individualisation des transferts	28
3.0.8	Choix du langage	30
4	La simulation : manuel de l'utilisateur	32
4.1	Le manuel de Liam2	33
4.2	Le manuel de Til-Liam	33
4.2.1	Options de la console YAML	33
4.2.2	Gestion des périodes	37
4.2.3	Points plus ou moins secondaires	38
5	La simulation : le modèle	39
5.1	L'organisation générale	39
5.1.1	4 entités	39

5.1.2	Pourquoi un registre ?	39
5.2	Description des étapes	40
5.2.1	Les étapes de la simulation dynamique	40
6	La simulation : les données	46
6.1	Les carrières passées	46
6.1.1	Intérêt de l'enquête patrimoine	47
6.1.2	Limites de l'enquête	47
6.1.3	Fermeture de l'échantillon	48
	Annexes : dictionnaires des variables	52
A.	Fichiers sources	52
B.	Fichiers simulation	52
C.	Paramètres	52
D.	Le matching Patrimoine-EIC	52
	Résultats	54
7	Installation	55
7.1	Environnement de l'ordinateur	55
7.1.1	Installer Python	56
7.1.2	Installer R	57
7.1.3	Interface Python et R	58
7.1.4	Et \LaTeX ? Et Git?	58
7.2	Les fichiers de TaxIPP-Life	58
7.2.1	Où trouver les sources de TaxIPP-Life	58
7.2.2	Chemin d'installation	59
7.3	Ma première simulation	60
	Bibliographie	60
	Glossaire	61

CHAPITRE 1

AVANT-PROPOS SUR CE DOCUMENT

1.1 La bible de TaxIPP-Life

Ce document s'attache à décrire le modèle TaxIPP-Life. Pour un modèle de microsimulation, on doit avoir plusieurs niveau de description correspondant à différent type de lecteurs. On peut définir en particulier trois grandes catégories, ceux qui veulent savoir comment le modèle marche et ceux qui veulent savoir comment l'utiliser pour faire des simulations et ceux qui veulent y participer à sa construction. Les premiers veulent savoir comment le modèle réalise ses simulations, quelles sont les hypothèses qui ont été faites et se faire une idée générale du modèle. Les deuxièmes veulent pouvoir installer les programmes comprendre ce qu'ils doivent changer en fonction de leur besoin. Enfin, les derniers doivent savoir quels sont les choix de programmation qui ont été retenus (au moins dans les grandes lignes) et avoir toutes les informations nécessaires pour pouvoir améliorer le code.

En général pour parler à ces différents publics, on sépare la documentation en un document technique et un guide de l'utilisateur. Ici, les deux documents sont rassemblés. Ce document se veut donc exhaustif pour essayer de limiter l'éparpillement des informations. On fera attention à bien permettre plusieurs niveau

de lecture. Certains chapitre entier seront entièrement destiné à un seul public. Pour la description du modèle étape par étape, on utilisera une structure présentant d'abord la simulation, puis expliquant le code, le lecteur en fonction de ce qui l'intéresse pourra donc passer rapidement sur certaines parties sans problèmes. L'intérêt de cela est aussi de permettre de pouvoir entrer dans chaque partie du programme directement et de s'y consacrer plus ou moins en fonction de son sujet d'intérêt.

Si on voulait tout de même publier un guide de l'utilisateur ou un document technique, on pourrait sélectionner les informations de ce document. Toutefois, comme on veut que ce présent document reste la bible associée au modèle Taxipp, tout ce que l'on ajoute à ces publications doit se retrouver ici.

1.2 Message aux codeurs

Je m'adresse ici particulièrement et uniquement à ceux qui vont être amené à changer le code. Si vous continuer à lire, c'est donc que vous aller modifier le modèle. L'améliorer ! La microsimulation est un domaine à part en économie. Par rapport aux études classiques, le point important est qu'il s'agit d'un projet d'ampleur qui ne doit pas répondre à une seule question prédéfinie à l'avance. Taxipp est complètement dans cet état d'esprit et a été écrit précisément pour être ouvert et permettre différentes études. Un bon point c'est qu'on ne part pas de rien. En revanche, il y a une contrepartie c'est que chacun doit veiller à ce qu'il faut veiller à ce que les programmes soient propres, à faire en sorte dans la mesure du possible que rien de ce que l'on fait ne limite ce que les autres peuvent ou pourront faire. L'investissement que chacun doit faire pour les autres se traduit aussi par la mise à jour de ce document. D'expérience, il y a beaucoup de perte possible dans un tel modèle. La règle est simple, si c'est pas ou mal documenté ce que vous avez fait

ne sert à rien. Les gens qui prendront votre suite, s'ils savent ce que vous avez fait et pourquoi vous l'avez fait, pourront s'appuyer dessus, ne pas défaire minutieusement ce que vous avez codé pour se rendre compte bien après qu'en fait c'était bien¹. Enfin, il faut bien le dire, lire un code avec une documentation pas à jour est plus destructif qu'autre chose.

J'espère vous avoir convaincu à la nécessité de travailler proprement et de consacrer ce temps qui a l'air ingrat mais qui laisse votre nom à la postérité, d'ailleurs commencer par mettre votre nom dans l'historique !

1. Ceci dit, il faut veiller à ne pas écraser ce que les prédécesseurs ont fait, si on peut facilement revenir à l'état précédent c'est mieux. Ca sert toujours pour comprendre les évolutions du modèle.

CHAPITRE 2

INTRODUCTION

2.1 L'ambition de TaxIPP-Life

2.1.1 Un modèle de microsimulation généraliste

L'objectif du modèle Taxipp est l'étude du cycle de vie dans toutes ses dimensions. Il se distingue en cela de la plupart des autres modèles de microsimulation dynamique qui se concentrent sur un domaine particulier : l'éducation, les retraites, la dépendance ou l'activité féminine par exemple.

La généralité désirée du modèle pousse à n'avoir aucune impasse sur les différentes étapes marquantes de la vie économique de la population. Taxipp est ainsi, à ma connaissance, le seul modèle de microsimulation intégrant les héritages et droit de succession.

Prendre en compte le plus possible de dimension permet aussi en théorie d'étudier des phénomènes plus complexes, d'un second ordre. On peut penser à l'influence de l'assurance chômage sur les décisions d'éducation, ou celle de l'allongement de la vie, sur l'immobilier, les loyers et les allocations logements mais aussi à bon nombre d'autres problématiques.

Cet objectif généraliste permet à chaque partie de profiter de l'expertise du reste du modèle. La structure du modèle doit permettre à ce qu'un travail sur les choix de départ en retraite profite aux études qui étudie la consommation des senior par exemple. L'investissement qui peut être fait sur les parties sensibles et clés du modèle comme le marché du travail ou la démographie est utile pour toutes les études et le spécialiste de la santé n'aura pas besoin de se spécialiser dans le marché du travail plus qu'il ne le souhaite.

2.1.2 Mesurer les niveaux de vie

Taxipp part du principe que les études en coupe sont par définition limitée. Tout le monde est conscient que le revenu d'une année ne peut-être qu'un proxy du niveau de vie. Un revenu sur cycle de vie ne sera aussi qu'un proxy mais forcément de bien meilleur qualité. On peut en effet faire trois grand reproche au revenu en coupe. Celui-ci fluctue d'une année sur l'autre, il n'a presque pas de sens dans les années d'inactivité, en particulier pendant les études et un revenu en coupe est toujours exposé au fait qu'il ne tient pas compte du patrimoine.

A COMPLETER avec document JEM + rapport Alexis

2.1.3 Mesurer la redistribution

A COMPLETER avec document JEM + rapport Alexis

2.2 Historique

2.2.1 Contexte lors de la création

On ne peut pas parler de l'historique de Taxipp sans parler des deux piliers sur lesquels s'est appuyé le modèle et qui étaient là avant lui : Liam2 et OpenFisca. Le premier est bon pour la partie dynamique de la simulation, le second est bon pour la partie législation. Il faut remercier ces deux projets, et le bon goût qu'ils ont eu d'être OpenSource. Il faut aussi remercier les personnes qui en étaient responsable de ces modèles à la naissance de Taxipp, respectivement Gaëtan de Menten et Mahdi Ben Jelloul. De nombreuses interactions ont permis de profiter de leur expérience et de reprendre ces travaux pour la construction de Taxipp. Ces échanges ont été très importants pour Taxipp et on ose espérer que Liam2 et OpenFisca ont aussi profité du regard extérieur sur leur travaux. Il serait évidemment bon que les trois modèles continuent de se développer, et le plus collectivement possible.

Le modèle Taxipp est aussi héritier du modèle Destinie de l'Insee. Les documents publiés sur ce modèle ainsi que les échanges avec son équipe de développement a permis d'avoir rapidement de bonnes équations de simulation. En particulier, Didier Blanchet, parce qu'il avait commencé une version en R de Destinie, a permis à Taxipp de s'appuyer dès le début sur un code lisible et simple avec les grandes lignes d'un modèle de microsimulation dynamique.

2.2.2 Frise chronologique

Le projet Taxipp a débuté en septembre 2012 à l'initiative d'Alexis Eidelman qui a passé un an pour asseoir les bases du projet. Avec Béatrice Boutchenik, une première étape a été l'appariement de l'enquête patrimoine avec les DADS¹.

1. On pourra lire son mémoire de master pour plus d'informations

?	Personne sur le projet			Réalisation	Production
?	Alexis	Béatrice	?	?	?
Sept 2012	X	X	?		?
Dec 2012	X	?	?	Matching Patrimoine EIR-EIC	?
Mai 2013	X	?	?	Connection OpenFisca - Liam	Présentation JEM
Juillet 2013	X	?	?	?	Présentation IFS
Aout 2013	X	?	?	?	Documentation
?	?	?	?	?	?

CHAPITRE 3

LA SIMULATION : LA TECHNIQUE

Un certain nombre de questions doivent se poser lorsque l'on fait de la microsimulation dynamique et nous les abordons ici. Ce chapitre se concentre donc essentiellement sur le moteur de la simulation, et il n'est pas nécessaire de maîtriser les points ci-dessous pour utiliser le modèle. On prendra en revanche soin de lire la seconde section de ce chapitre qui explique comment utiliser Taxipp.

On essaie de garder la même structure pour chaque question afin de faciliter la lecture. On situe d'abord le problème puis on explique la solution retenue par Taxipp. Cette solution est décrite par son principe de base, puis sa réalisation technique. On discute ensuite de ce choix et on donne des pistes d'améliorations potentielles.

Les lecteurs sont aussi invités à s'économiser le plus possible la lecture de cette partie sur les réalisations techniques, à ne lire que les parties qui les intéressent et le cas échéant à passer directement au chapitre suivant concernant la simulation proprement dite.

3.0.3 Alignement

3.0.3.1 Difficulté

Un exemple vaudra mieux qu'un long discours. Si on a une population de 100 individus et que l'on veut simuler chaque année 10 naissances et 10 décès, on ne voit pas la difficulté. Mais, il s'agit en général de simuler la probabilité de décès de chaque individu. La première idée est de tirer un nombre aléatoire u_i dans une loi uniforme entre 0 et 1 et de la comparer pour chaque individu à la probabilité individuelle (ici dans cet exemple très simple $\frac{1}{10}$) et de sélectionner les cas où $u_i < P_i$. Mais en faisant cela, on aura en moyenne dix événements mais pas à chaque tirage. On peut s'éloigner de réalité statistique, on peut aussi avoir un effet en cascade et ne plus avoir une population constante.

On est alors tenté de sélectionner les plus hauts scores. Par exemple, si nos 100 individus sont âgés sont tous nés successivement avec un an d'écart (ils ont donc entre 1 et 100 ans) et que la probabilité de mourir croît avec l'âge alors on sélectionnera les 10 plus âgés. Cela pose encore un problème car en agissant de la sorte les probabilités d'événement ne sont pas du tout proportionnelles aux probabilités individuelles. Dans notre exemple un individu de 40 ans aura une probabilité nulle de mourir tant qu'il y aura 10 personnes de plus de 40 ans dans l'échantillon. Au final, on aura une simulation sans aucun décès à cette âge donné contrairement à la réalité.

3.0.3.2 La solution de TaxIPP-Life

Taxipp utilise la méthode déjà implémentée dans `liam2`. Depuis juin 2013, il est aussi possible d'utiliser une méthode dite `sidewalk`. Cela est implémenté dans le code `alignement.py` de `liam`. Cette méthode est celle utilisée par Destinie.

Principe Pour des informations sur ces méthodes et sur l'alignement en général, il est vivement conseillé de se reporter au survey de Li et O'Donoghue.

Réalisation Simple application du code de Liam2.

3.0.3.3 Limites et idées d'amélioration

On aimerait pouvoir tester encore d'autres méthodes d'alignement.

Pour la sidewalk method, il faut avoir en tête qu'on sélectionne en théorie en fonction de la somme des probabilité, on peut sélectionner moins mais pas plus. Il n'y a pas trop de bidouillage possible dans ce cas même si Li et O'Donoghue donne tout de même des pistes. Ici, on se contente de ce la somme des probabilités puisque c'est ce que Destinie fait.

3.0.4 Pondérations

3.0.4.1 Difficulté

Une différence entre de la microsimulation dynamique et de la microsimulation statique est que les liens entre individus peuvent évoluer lors de la simulation. Si les individus au départ de l'échantillon ont des pondérations cela pose un problème dans les liens. En particulier, le problème se pose pour les mariages. Comment simuler un marché du mariage si les individus n'ont pas tous la même pondération ? On ne peut pas unir un individu représentant 5 000 personnes avec un individu en représentant 10 000 par exemple. Combien aurait-on de telles union ? Quel est la pondération du couple ? Combien de nouveau nés sont représentés par une naissance dans ce couple ? Il n'y a pas de sens à faire cela.

3.0.4.2 La solution de TaxIPP-Life

Principe Il y n'y a guère que deux options possible. Ou bien lors du matching, si on associe les 5 000 personnes aux 10 000, séparée les 10 000 en deux groupes de 5 000 (représenter par deux individus dans la base) et marier un des groupes laissant l'autre célibataire. L'autre option est de dupliquer dès le départ les individus pour avoir une pondération de 1 et autant de lignes que l'on a d'individu dans la population étudiée. En fait, on n'a besoin que d'une pondération uniforme qui doit être le plus grand commun dénominateur des pondérations en théorie. C'est cette dernière option que l'on retient dans TaxIPP-Life.

La première a pourtant l'air plus attractive car elle ne demande pas d'agrandir l'échantillon avant la simulation et permet donc de tourner plus vite. Cependant, elle complique beaucoup le code, en particulier pour des étapes d'optimisation. On n'exclut toutefois pas d'y venir un jour.

D'y venir ou plutôt d'y revenir. Initialement, les choses étaient codées comme cela dans Liam2. Eux-même, on fait le choix d'abandonner partant en effet du fait que le code devenait trop complexe. De plus, ils remettaient en cause la vraie efficacité comparée puisque selon eux, après quelques périodes, les pondérations devenaient 1 pour tout le monde ¹.

Il est plus simple pour l'instant de ne pas trop s'éloigner de Liam2. De plus, et c'est un point important, la duplication de l'échantillon a l'avantage de permettre au même individu de la base initiale d'avoir plusieurs trajectoire professionnelle par exemple. On obtient ainsi de la variabilité tout à fait appréciable au niveau de chaque individu dans le modèle.

Réalisation La duplication des individus se fait au niveau de la préparation des données. La première tentative avait été d'utiliser une étape de Liam2 directement pour cette duplication. Finalement, elle se fait en R, ce qui dans ce cas précis est plus simple et plus cohérent.

La duplication doit en effet obligatoirement précéder l'étape de fermeture de l'échantillon ² pour que les liens parents enfants respectent les pondérations. Comme on veut toutefois pouvoir travailler sur un petit échantillon pour avoir des résultats rapidement, l'étape de duplication est une étape relativement explicite du fichier `run_all.R`.

Dupliquer les ménages ou les individus en fonction du poids n'est pas en soit compliqué. En revanche, il faut être minutieux sur le liens entre les différentes

1. Il faut en effet garder à l'esprit que si un individu est "splité" alors il faut aussi spliter toutes les personnes qui lui sont liés : ses enfants par exemple. Après les enfants, et les parents, les conjoints de ceux-ci, puis les parents et enfants des conjoints, etc. On comprend que la chaîne des relations peut s'avérer assez longue.

2. (Mettre une référence en latex vers la section ici). On trouvera aussi dans cette section des chiffres montrant que cela est indispensable

lignes des tables.

TODO : Documenter le code

3.0.4.3 Limites

Les programmes sont donc en place. En revanche, on a passé dans le paragraphe précédent sur le fait que pour l'instant, la pondération uniforme utilisée n'était pas ni 1, ni le PGCD, ni même la pondération minimale mais 200. Pourquoi cela ? Il faut savoir que dans l'enquête Patrimoine 2010, qui contient un sur-échantillonnage des hauts patrimoines, la plus petite pondération est de 6. Même en prenant 6 comme approximation grossière du PGCD, cela nous mène en théorie à une base de plus de 10 millions de lignes dans la base pour représenter la population française. C'est beaucoup. Le nombre de 200 est donc un arbitrage entre résoudre le problème des pondérations et avoir un échantillon de taille acceptable pour que la simulation tourne en un temps raisonnable. Si la partie dynamique ne pose pas trop de problème pour l'instant la simulation de la législation ne supporte pas de trop grande table. Pour l'instant, on travaille tout de même à partir de

On fait donc pour l'instant semblant que tout le monde représente 200 personnes. Toute personne dont la pondération initiale est inférieure à 399 a donc un poids de 200. C'est une sérieuse limite.

Plus important, quel que soit la pondération finale choisie, les pondérations sont réelles et le nombre de duplication entier. Il y a un arrondi à faire (sauf si on veut dupliquer pour avoir une pondération par exemple de un millièmme mais là ça va faire beaucoup). Une pondération de 1,4 devra être ou bien une ligne de la base ou deux. Il se peut donc que la somme des pondérations finale de soit pas la somme des pondérations initiale. Il faut donc garder un facteur multiplicatif pour la production de chiffres finale. Ce n'est pas fait pour l'instant.

3.0.4.4 Idées d'amélioration

La principale amélioration serait de permettre l'utilisation de tailles plus grande. Pour cela un travail sur OpenFisca est en cours pour alléger les tables et pour permettre de faire des calcul sur des bases plus grandes. En fait, depuis juin 2013, il est possible de calculer la législation par morceau (chunk en anglais) et même si ce n'est pas encore tout à fait au point le problème n'est plus technique encourageant. La dernière limite se situe au niveau de R qui semble ne pas aimer les trop grosses tables. Mais comme cette étape ne concerne que l'initialisation, il y a certainement possibilité de contourner ce problème.

On peut aussi citer l'idée de la technique employée par le modèle DESTINIE de l'Insee qui permet potentiellement d'utiliser un petit échantillon avec pondération uniforme mais conservant la représentativité de l'ensemble de la population. Après l'étape de duplication, un nombre de ménage est sélectionnés. Si on effectue cette étape après la fermeture de l'échantillon (ce qui est souhaitable pour optimiser les chances d'avoir des liens parents-enfants représentatif), il faut aller chercher les ménages rattachés. Sinon, il faut effectuer ensuite l'étape de fermeture. On fait ainsi une sorte de bootstrap tout en respectant travaillant avec une pondération uniforme, ce qui peut être intéressant.

3.0.5 Formation des unions

3.0.5.1 Difficulté

Par rapport aux autres étapes, la formation des unions à ceci de particulier qu'il s'agit d'un marché et non pas d'une imputation individuelle. Le conjoint imputé à Alain ne dépend pas que d'Alain mais aussi des personnes célibataires en même temps qu'Alain.

Nous nous intéressons ici qu'à l'étape de formation des unions et pas de la sélection des candidats potentiels au mariage.

Il faut noter que l'on peut avoir deux approches différentes. Comme souvent en microsimulation dynamique, on peut chercher ou bien à modéliser les comportements ou bien à reproduire les observations statistiques. Les deux ne sont bien sûr jamais décorrélés. Dans le cas des unions, on peut vouloir ou bien simuler une probabilité de rencontre et de mariage correspondant à ce qui semble être la réalité des rencontres ou bien voir le problème sous un angle de microsimulateur qui serait : A une période de ma simulation donnée, je sais que je dois simuler des unions, comment je vais associer mes personnes célibataires pour reproduire des unions acceptable par exemple en terme de différence d'âge et de diplôme.

La solution de  entre pour l'instant résolument dans le second cadre.

Une fois répondu à cette première question, la mise en application est encore périlleuse. En effet, les temps de calcul des solutions optimales invite à les laisser sur le côté. Il y a donc un arbitrage à faire entre temps de calcul et précision.

3.0.5.2 La solution de TaxIPP-Life

Principe Nous définissons une valeur, un score, de chaque couple potentiel en fonction des caractéristiques des deux conjoints. L'objectif est ensuite de minimiser cette distance globale pour obtenir le meilleur matching. Une méthode brutale est d'étudier toutes les combinaisons possibles. Il est bien sûr impensable de l'appliquer puisque cela demande un temps de calcul en $n!$ si n est le nombre de mariage. Heureusement, il existe une méthode plus simple, dite méthode hongroise (ou de Munkres-Kuhn-Tucker). Elle est tout de même assez coûteuse en temps de calcul, et inapplicable dans notre cadre quand la taille de la base augmente. Elle a aussi le désavantage d'être avant tout pensée pour des matrices carrées, avec autant de candidats des deux côtés du mariage. Pour nous, on aimerait que ce soit l'algorithme d'optimisation qui nous dise les candidats à l'union qui vont rester sur le carreau.

La méthode appliquée ici n'est pas du tout raffinée. Il s'agit simplement de faire un matching par le tri. On balaie une des deux parties, que l'on appellera la population receveuse. Pour chaque receveur, on cherche parmi les donneurs le meilleur matching. Si plusieurs candidats ont la valeur minimale, on en sélectionne un aléatoirement. On retire la personne sélectionnée du bassin de donneur et on réitère pour le receveur suivant.

Réalisation La réalisation se fait simplement en utilisant une fonctionnalité de `liam2`. Notons que l'ordre dans lequel on balaye la population receveuse est important. La première personne de la liste aura en effet le matching qui de son point de vue est optimal alors que la dernière aura celui ou ceux qui restent. Et peut-être même personne.

Pour l'instant, on ne définit pas d'ordre. Pourtant, traditionnellement on essaie de trouver les cas les plus difficile à matcher pour avoir du choix dans le conjoint et ne pas se retrouver avec des couples vraiment trop excentrique. Comme on vient

de le laisser entendre, cela implique que ces personnes difficile à matcher le seront toujours.

3.0.5.3 Limites

Le matching n'a évidemment pour l'instant rien d'optimal. En terme d'output d'abord. En terme de calcul, l'étape de matching demeure l'étape la plus gourmande en temps.

3.0.5.4 Idées d'amélioration

On pourrait changer l'ordre en fonction du nombre de donneurs et du nombre de receveurs. Si on a moins de receveurs, on sait que l'on va tous les appairer, autant commencer par les plus durs. Si on a plus de receveurs, on sait que certains ne vont pas être matchés, autant laisser les cas compliqués de côté. En fait, la bonne façon de faire est probablement d'inverser les positions de donneurs et de receveur en fonction de la taille de leur population. Si la population A a moins d'individu que la population B alors c'est elle qui doit être la receveuse. On classe par difficulté A permettant ainsi à ces éléments extravagants d'être appairer avec le meilleur match possible. Les individus difficile à matcher de B seront ceux que l'on va laisser.

Une idée pour améliorer le temps de calcul de cette étape et aussi introduire un peu de variabilité serait pour chaque receveur de tirer aléatoirement un nombre k de receveurs et de faire la sélection parmi ces k prétendant. On passe de $\frac{n(n-1)}{2}$ opérations (en ordre de grandeur et quand on a autant de receveurs que de donneurs) à $n \times k$ (là aussi en première approximation car il peut y avoir moins de k candidat à la fin), ce qui est intéressant.

Il s'agit pour l'instant de reproduire des éléments statistiques, on pourrait ima-

giner simuler un véritable marché du mariage (voir les travaux de Chiappori par exemple sur le sujet). Comme le problème peut avoir une forme linéaire, ce n'est peut-être pas si coûteux en temps de calcul. Cela reste un objectif de moyen terme, un beau sujet pour une étude particulière.

3.0.6 Le choix du pas temporel

3.0.6.1 Difficulté

Le choix d'un pas de simulation relève d'un arbitrage entre précision des calculs d'un côté et simplicité et temps de simulation de l'autre.

D'abord précisons, ce qui peut influencer le choix de ce pas temporel. La période se caractérise en fait par la durée minimale de l'événement simulée. Par exemple, dans une simulation annuelle, Pour un élément de simulation donné, par exemple le mariage, une simulation annuelle ne dit pas que tous les mariages ont lieu le premier janvier puisqu'on peut simuler une date de mariage. En revanche, une simulation annuelle fait que le mariage durera au moins un an. Ce n'est pas tout à fait vrai puisque le mariage puisque cela dépend de l'étape de divorce ou d'autres étapes qui peuvent aussi modifier le statut. Ce qui est vrai c'est qu'il n'y aura pas d'autre mariage simulé pendant un an. On peut à partir de ce petit exemple retenir l'idée qu'il faut associer en théorie une période à chaque variable d'état. On peut en théorie simuler l'emploi de façon mensuelle et le statut marital de façon annuelle par exemple.

3.0.6.2 La solution de TaxIPP-Life

Taxipp permet de choisir plusieurs fréquences de simulation. On laisse donc libre l'utilisateur de choisir la précision qu'il veut. De plus, dans la simulation la plus fine qui soit, pour l'instant le mois, on permet de ne faire certaines étapes de la simulation sur des bases plus longues, annuelle ou trimestrielle par exemple.

Principe Liam2 fonctionne avec des périodes qui sont en théorie des entiers et sans signification particulière (ni des années ni des mois). On est donc libre de

choisir ce que l'on veut tant que l'on numérote les périodes de 1 à n . Mais on ne tient pas forcément à avoir ce système de numérotation puisqu'il n'est pas très significatif et surtout, puisque la signification change en fonction de la simulation, on ne peut pas utiliser la période à l'intérieur de la simulation, par exemple, on ne peut pas avoir *naissance : period* ou quoique ce soit utilisant explicitement *period*. Pour permettre l'utilisation de différent pas temporels, on s'impose le format suivant : **period = yyyyymm**. Ce choix répond à la contrainte de travailler avec des entiers. Il permet de lire directement l'année ce qui est important. Si on ne travaille qu'avec des années, on part du principe qu'on est en janvier yyyy01. La différence entre l'année 1234, qui s'écrit 123401 et janvier de cette même année est donc uniquement que lorsque 123401 désigne l'année 123402 n'existe pas. Ce n'est peut-être pas le plus judicieux et 123400 pourrait être envisagé sous réserve de répondre à quelques détails techniques. On essaiera aussi de revenir à 1234 plutôt que 123400.³

Quoi qu'il arrive, on code toujours à partir de l'intervalle de temps atomistique : le mois. Les autres périodes de temps sont définies comme étant des multiples de ce mois. On peut donc avoir des périodes qui durent 2 mois (bimensual), 3 mois (quarter), 4 mois (triannual) et 6 mois (semester) ou 12 mois(year).

Par rapport à *liam2*, on introduit dans la console *yaml*, un paramètre *time_scale* qui est le pas de la simulation, par défaut l'année. Les processus sont effectués à chaque période sauf s'ils ont une périodicité propre plus grande que le pas de la simulation. Par exemple, si le paramètre est 'year' par exemple pour le mariage, est que le pas de la simulation est mensuel alors le process ne sera effectué qu'un fois toute les douzes périodes. L'étape est réalisée à chaque fois dès que l'on entre dans la période concerné, donc dans le cas présent en janvier. Avec un processus semestriel, on l'effectue au premier mois de chaque semestre, au mois 01 et 07 en

3. On notera cependant que cela empêche d'avoir de fonctionner année par année mais en changeant au mois de juillet par exemple.

l'occurrence. On laisse le lecteur généraliser gaiement.

Réalisation Par rapport à Liam2, on ajoute le pas temporel au paramètre de sa fonction simulation. Notez que le nombre de périodes est toujours le nombre de simulations, si on veut faire tourner deux ans mois par mois, on doit donc mettre 24 mois. On peut imaginer ajouter un jour un paramètre et avoir "period : 5, year" pour multiplier automatiquement par le nombre qui va bien quand le pas temporel est le mois ou le semestre. Pratique si on ne maîtrise pas la table des 12 jusqu'à 50...

On regardera dans simulation.py, comment sont calculées les périodes et comment on travaille sur la périodicité. Rien de bien sorcier.

Ensuite, un traitement particulier est fait dans alignement pour aller chercher la valeur correspondant au pas temporel du processus. Si on a des données au mois, et qu'on veut simuler à l'année, il faut sommer les 12 mois de l'année. Inversement, si on a des données à l'année et que l'on veut un mois, on prend la valeur annuelle et on la divise par 12 (ou le nombre de sous-périodes). Tout ceci est fait dans une fonction `kill_axis` de `alignment.py`.

On pourrait éventuellement supposer une autre répartition que uniforme sur l'année. Par exemple, en faisant une sorte d'interpolation avec les valeurs des années précédente et suivante pour que, dans une phase de croissance par exemple, on garde une progression sur l'année. Toutefois, si on a envie de faire ça, le plus simple est peut-être de changer le fichier csv même si je n'aime pas trop qu'on en fasse pas la différence entre les données sources pures et les données modifiées qui découle d'une hypothèse, aussi intelligente cette dernière soit-elle.

Attention, il faut faire un peu attention dans ce processus. Ce n'est pas la même chose de dire que (prenons des chiffres ronds) que 40% de décès sur une population de 10

000, ce n'est pas la même chose que 10% sur 10 000 puis 10% sur 9000, puis 10% sur 8 100, etc. Dans un cas, on finit avec une population de 6 000 personnes dans l'autre une population de 7 500 personnes. Même si ce cas n'est problématique que lorsque l'événement modifie la base de tirage, il faut faire attention à cela. Dans ces cas tirer, un nombre d'événements plutôt qu'une proportion est plus sûr.

Si on aligne, justement, il faut écrire un programme qui permet d'aller chercher automatiquement les valeurs de calage correspondant à la période à partir d'un fichier excel.

Enfin, toujours sur l'alignement, en cas d'utilisation de la méthode "sidewalk", ou d'un nombre défini, on ajoute une option qui donne la période à laquelle l'équation, ou le nombre d'événement donné, correspond. C'est le même principe que pour les données de calage externe. L'intérêt est de pouvoir garder la même équation quel que soit le pas de la simulation.

3.0.6.3 Limites

La limite est donc le mois pour l'instant. Passer à la semaine ne devrait pas poser de problème majeur. Enfin, par rapport aux mois, les semaines ont le désavantage de ne pas être en nombre ni entier, ni constant dans les années.

Pour la partie législation, il faut noter que la simulation est de toute façon annuelle, on aimerait toutefois faire un peu mieux.

3.0.6.4 Idées d'amélioration

Pour les simulations faites sur une périodicité plus grande que le mois, elles sont faites en début de période, mais en cas d'événement, il faut faudrait imputer un mois. Si on ne le fait pas, alors tous les changements ont lieu en janvier par

exemple. Du coup, si la situation sur le marché de l'emploi dépend du statut conjugal, alors on observera un pic en janvier mécanique. Cela n'est pas grave en soit mais on peut être sûr qu'un jour quelqu'un perdra ça de vue, et s'en étonnera. Par exemple, dans une étude sur l'emploi saisonnier ou même en essayant d'aligner les résultats sur des données trimestrielles. Il est donc sage de simuler dans ces cas un mois d'événement.

En collaboration avec OpenFisca, il faut aussi calculer les transferts en fonction du mois. Le salaire brut serait ainsi mensuel, le RSA trimestrielle, les allocations en général mensuel. Il faut certainement penser un peu les choses pour ne pas avoir à faire tourner 12 fois l'intégralité de la législation.

3.0.7 L'individualisation des transferts

3.0.7.1 Difficulté

Dans une étude sur le cycle de vie, l'unité d'étude est résolument l'individu. Mais bon nombre de dispositifs sont familialisés, l'impôt sur le revenu ou les prestations familiale par exemple. Il faut alors comprendre quelle est la part du transferts attribuable à chacun.

3.0.7.2 La solution de TaxIPP-Life

Principe La méthode est pour l'instant un peu *ad-hoc*. L'impôt sur le revenu est partagé entre les deux membres du couple. Les prestations familiales sont partagées entre les enfants à charge. Le reste, minima et allocations logements, sont répartis entre tous les membres du ménage. A chaque fois la répartition se fait à part égale.

Réalisation L'individualisation se fait à l'aide d'une fonction qui se trouve dans le dossier `til/src/`. Elle est appelée dans `of_on_liam.py` avant le merge avec le registre.

Au début, on mettait les résultats brut de OpenFisca dans les bases (on faisait un merge avec `simul.h5`). L'individualisation était alors faite dans une étape registre. On peut toujours revenir à cela ou bien en restaurant les programmes ou bien, en sommant sur l'entité⁴

3.0.7.3 Limites

Il y a plusieurs limites, on peut se dire que l'impôt sur le revenus par exemple, devrait être réparti en fonction de revenus. Les allocations familiales dépendant de

4. Je préfère cette option qui est plus souple si on veut tester une option avec mariage et une sans par exemple. A ce moment là, si dans une première période on simule le séparer, dans une seconde le non séparé alors c'est mieux de faire la somme au niveau de foy pour la comparaison finale et l'optimisation du conjoint.

l'âge de l'enfant, il peut être assez aisé des les répartir plus justement.

Essayer de répartir selon une règle unique les transferts alors que ceux-ci dépendent du couple semble assez arbitraire. On pourrait avoir envie d'affecter plutôt une valeur dépendant de l'apport marginal de l'individu. Combien est-ce que le ménage paie d'impôt en plus parce que je suis là. Combien est-ce qu'il recevrait d'allocation logement en moins sans moi ?

3.0.7.4 Idées d'amélioration

En dehors d'amélioration dans la méthode utilisée jusque là, une idée est de changer complètement de méthode. Comme le sous-entend la section précédente, utiliser une valeur marginale est tentant. Malheureusement, on peut montrer que cette méthode présente des défauts importants. Un document de travail de l'Insee montre que la valeur de Shapley est la solution pour garder cette idée, répartir les transferts de manière juste et avec de bonnes propriétés. Seul souci, elle doit étudier toutes les configurations du ménage, pour k personnes, il faut étudier $2^k - 1$ configurations. Cela dit, il existe certainement des moyens de ne faire tourner ces points que sur un petit nombre de transferts. Par exemple, certains éléments sont individuels et n'ont pas besoin d'être recalculés. On peut aussi avoir des raccourcis pour les individus qui sans revenus, pour eux l'influence sur les prestations et même sur l'impôt sur le revenu n'est pas compliqué à implémenter. Il faut tout de même être prudent car les différents éléments de la législation sont imbriqués les uns dans les autres...

3.0.8 Choix du langage

3.0.8.1 Difficulté

La question est évidente pour coder un modèle de microsimulation dynamique il faut s'appuyer sur un langage de programmation. Blanchet 2013, étudie cette question et milite pour l'utilisation de R. Ceci étant, il part du principe qu'un logiciel statistique est un avantage parce qu'il est en général simple et connu des potentiels développeur de modèle qui sont plus souvent des statisticiens que des programmeurs.

En dehors de la simplicité d'un langage, son coût mais surtout sa rapidité de calcul doivent être des paramètres déterminant.

3.0.8.2 La solution de TaxIPP-Life

Principe Le moteur de Taxipp est écrit en langage Python. Une interface simplifier (langage YAML) permet pour un certain type d'utilisateur de rester à un niveau très simple. Plus simple que pour les langages statistiques. Ces deux niveaux de langage a aussi l'avantage de pouvoir bien séparer ce qui relève du général, du moteur et ce qui relève de la modélisation proprement dite.

Réalisation

3.0.8.3 Limites

3.0.8.4 Mise à jour

Il y a deux types de mises à jour. Les packages de Python évolue et il est bon de se tenir au courant. En particulier pour pandas qui évolue vite et qui est central.

Ensuite, les nouvelles version de Liam2 demande un peu de travail. En effet, d'une part Liam2 n'est pas sur GitHub (même s'ils y pensent). D'autre part et surtout, les modifications apportées aux programmes pour l'utilisation de Taxipp ne

sont pas toujours intégrées dans le code de liam2. Ainsi lors d'une mise à jour, il y a potentiellement un conflit entre le nouveau fichier de liam et le fichier de liam2 amélioré à la sauce Taxipp . Il faut donc balayer les modifications pour à la fois tenir compte des améliorations de liam2 (et entre en cohérence avec leur documentation) et à la fois ne pas perdre ce qui a été fait en vue de faire tourner Taxipp .

CHAPITRE 4

LA SIMULATION : MANUEL DE L'UTILISATEUR

Le modèle Taxipp s'appuie sur le modèle de simulation dynamique Liam2. De ce fait, le manuel de l'utilisateur est essentiellement le manuel d'utilisation de Liam2 que l'on peut trouver sur <http://liam2.plan.be>.

Ce n'est toutefois pas complètement suffisant. En effet, les programmes de liam2 utilisé par Taxipp sont modifiés. Contrairement à OpenFisca, pour lequel les modifications sont pour l'instant, directement intégrées au modèle, les modifications effectuées par Taxipp, ne sont pas toujours reprises par Liam2 si bien qu'il y a un décalage entre Liam2 et ses versions successives et ce qu'on peut appeler Til-Liam qui est la version de liam2 adaptée à Taxipp.

On dresse ici la liste des modifications par rapport à la version "officielle" de Liam2. Même si le titre du chapitre laisse penser qu'on ne traitera que de l'utilisation des fichiers YAML, il y aura aussi des points de programmation un peu plus techniques. En effet, il est important pour le codeur de savoir où se situent les différences dans les codes entre Liam2 et Til-Liam. Cela permet de voir ce à quoi il faut

faire attention lors des mises à jour de Liam2. Cela permet au codeur de Taxipp d'identifier ces points qui sont parfois moins généraux et codé un peu différemment de Liam2. On peut aussi se dire, entre nous, que la base de Liam2 est pour l'instant a priori de meilleure qualité que les additions qui ont été faites. En effet, ces patchs sont probablement parfois un peu ad-hoc et peut-être pas complètement optimisés.

4.1 Le manuel de Liam2

On le redit, pour utiliser Taxipp, il faut lire le manuel de Liam2 : <http://liam2.plan.be>.

4.2 Le manuel de Til-Liam

Dans un premier temps, on présente les modifications par rapport au manuel de Liam2 qui peuvent modifier l'écriture des fichiers YAML. Il s'agit en général d'options rajoutées ou bien dans la partie de définition de entities ou bien dans la définition de la simulation.

Un bon nombre des modifications concernent, comme vous allez le voir, le travail sur les périodes. Là où Liam2 est pensé pour travailler avec des périodes (des années) Taxipp souhaite pouvoir choisir des pas de simulations. On s'attardera spécifiquement sur les périodes pour expliquer les choix faits par Til-Liam.

Enfin, une petite partie décrira les modifications plus esthétiques qu'autre chose.

4.2.1 Options de la console YAML

On sépare les modifications entre celle qui modifient l'écriture des entities (y compris leur process) et celle qui modifie la partie simulation.

4.2.1.1 Entities

Dans la définition des variables, on a un champ optionnel **default** qui permet d'initialiser directement lorsque la valeur n'est pas fournie. Par exemple quand `initialdata==False` ou dans la création d'une nouvelle ligne (fonction `new`). Ceci devrait être repris à terme par Liam2.

On ajoute deux nouvelles fonctions. Associées au fait que l'on a modifié la façon de travailler avec les périodes (voir plus bas). On a donc une fonction **add_time_scale(expr)** qui ajoute à l'expression *expr* qui doit être un entier je pense, le nombre de mois depuis la précédente période calculée. Et une fonction **add_time(expr)** qui ajoute à une expression qui doit avoir le format d'une date (yyyymm), la durée écoulée depuis la précédente période. On comprendra peut-être mieux tout ça dans la section suivante qui traite spécifiquement des périodes.

On a aussi deux fonctions **month** et **year** qui donne le mois et l'année. Ces deux fonction sont extrêmement simples et *year(period)* pourrait par exemple être remplacé par *period/100* mais l'intérêt est que si on change quelque chose dans la façon d'écrire les périodes, on n'aura pas besoin de revoir tout les fichier YAML.

La fonction **align** a été un peu modifiée. D'une part elle prend désormais un argument *periodicity_given* qui indique la période correspondant à l'alignement donné si celui-ci ne s'appuie pas sur des données Excel. On en a besoin parce que si on donne un nombre (ou une proportion) d'événement ou une probabilité (voir la méthode *sidewalk* ci-après), celle-ci est automatiquement multipliée ou divisée si on simule l'événement à une autre fréquence. Si on donne `need=1 200` et `year_given = 'year'`, alors si le processus est effectué sur une base mensuelle, on sélectionnera 100 événement à chaque mois. Pour l'instant 'year' est la valeur par défaut, mais il serait peut-être plus sage de demander systématiquement à l'utilisateur de

donner cette précision explicitement.

Toujours dans la fonction **align**, on a ajouté l'option *method*. En effet, en plus de sélectionner les scores les plus importants (dont on ne dira jamais assez que sans utiliser la fonction `logit_score` elle est difficilement acceptable), on permet désormais la sélection par la méthode *sidewalk*. C'est celle retenue par Destinie. Elle a l'avantage de n'exiger qu'une probabilité et de sélectionner un nombre correspondant à la somme des probabilités, sans entrer de marge d'alignement prédéfini. Il faut faire attention au fait que seule une probabilité peut être donnée comme score dans ce cas. Une erreur est levée quand on entre un nombre d'événement si celui-ci est supérieur à la somme des probabilités. On pourrait implémenter une solution (voir échange de mail entre Alexis Eidelman et Gaetan de Mertens). Sinon, on ne peut pas savoir a priori quand ce cas va se rencontrer.

4.2.1.2 Simulation

Comme Taxipp a besoin de faire tourner la législation, il est normal de pouvoir définir un processus **Legislation**. Pour cela, on crée une nouvelle classe de processus nommé `ExtProcess`. Ce processus, n'est pas associé à une entity mais bien à toute. Ce processus `Legislation` prend deux arguments, un argument *annee* qui est l'année de la législation que l'on veut faire tourner (et qui pourrait être *'period'* pour indiquer que l'on veut la législation de l'année en cours), l'autre qui est l'argument *ex_post* n'est plus utilisable désormais. Il permettait au départ de calculer la législation ou bien à chaque période, ou bien en fin de simulation. On verra plus bas que l'ajout d'un champ final similaire à `init` permet de faire ce choix sans argument particulier.

Final apparaît comme possibilité dans le champs `simulation`. Il s'agit comme `init`

de définir des processus que l'on ne fait tourner qu'à la dernière étape. La dernière période est calculé puis les `final_process` tourne. Je dis ça mais, même si c'est vraiment simple à écrire, ce n'est pas encore fait.

Comme on l'a déjà dit plus haut, des modifications importantes sont faite au niveau du pas de la simulation. On pourra regarder la partie technique plus haut (référence).

Dans la zone simulation, on ajoute un paramètre `time_scale` qui part défaut vaut 'year'. Ce paramètre peut prendre comme valeur les subdivisions de l'année jusqu'au mois : 'semester', 'triannual', 'quarter', 'bimensual' et 'month'. On comprend que l'on va simuler les périodes selon la valeur donnée.

On profite pour dire que le processus peuvent dans Liam2 être entré avec une *périodicité* (initiative de Taxipp pour cette option aussi). Cette périodicité peut être un entier k pour dire que l'on simulee une fois toute les k périodes. Toutefois ceci est désormais non recommandé. On préconise maintenant d'entrer là aussi une valeur de temps, par exemple 'semester'. Ainsi le processus sera effectué toutes les périodes en cas de `time_scale` annuelle ou semestrielle. Et une fois toutes les six périodes si le pas est mensuel. L'intérêt est que tout s'adapte à `time_scale` et qu'il n'y a donc que cette valeur à changer.

Disons enfin, que l'on a aussi un nouveau paramètre *start* pour les processus qui donne le mois de calcul. Le meilleur (et peut-être unique exemple) d'application concerne la législation, que l'on veut calculer annuellement mais au mois de décembre alors que part défaut, les changements annuels sont effectués en janvier.

Une option **retro** permet simplement de remonter le temps plutôt que de simuler vers le futur.

Enfin, toujours parce qu'on ne veut pas tout changer à chaque fois que l'on change le pas, on introduit un paramètre **init_period** qui prend la place de *start_period*. *start_period* est la valeur de la première période simulée tandis que **init_period** veut être la période des données entrées, celles sur lesquelles on fait tourner *init_process*. On doit entrer obligatoirement l'une ou l'autre des deux valeurs mais pas les deux. Pour la fonction *simulation*, c'est **init_period** qui est donné comme paramètre, si dans le fichier YAML, on donne **start_period**, on traduit simplement $\text{init_period} = \text{start_period} - \text{pas temporel}$

4.2.2 Gestion des périodes

Les éléments ci-dessus sont relativement explicite. Pour l'alignement par exemple, on sait où se trouvent les modifications.

4.2.2.1 Différents pas temporel

Comme on l'imagine l'essentiel du travail est dans le programme *simulation*. Un point important est que l'on ne donne plus le nom de la période dans le contexte. On préfère en effet donner la liste des périodes, *periods*, et le numéro de la période en cours, *period_idx*. Il y a donc bien sûr moyen de retrouver la *period* qui est *periods[period_idx]*, mais on peut en plus et sans soucis accéder aux périodes passées alors que dans la version officielle de *liam2*, on prend toujours *period-1* par exemple pour avoir la dernière période.

4.2.2.2 Alignement

On travaille sur *need* en fonction de la fréquence de la simulation et de la *periodicity_given*.

Le point délicat concerne la lecture des données d'un fichier Excel. En résumé, il s'agit de deviner quelle est la *periodicity_given* dans la table excel. Il suffit en

théorie de compter le nombre de valeur entrée dans l'année. C'est certainement la bonne vision à avoir et il faudrait peut-être modifier les programmes en ce sens pour plus de lisibilité.


4.2.2.3 Global

On prend la dernière valeur entrée. Pour l'instant, on ne peut avoir que des valeurs annuelles dans global, mais c'est facile à changer.

4.2.3 Points plus ou moins secondaires

Les **import** ont d'abord été implémentés dans Til-Liam, ils ont depuis été repris et amélioré par Liam2. Un bon exemple d'innovation réussie.

Pour commencer j'ai modifié l'affichage parce que je voulais quelque chose de plus condensé. Mais il serait peut-être mieux d'en faire une option. Pour commencer sur le modèle c'est peut-être bien d'avoir l'affichage original.

Le fichier import a été modifié pour pouvoir lire les données directement depuis R en utilisant la librairie pandas. C'est le format de référence désormais utilisé dans  .

On ajoute à la fin de la console le temps moyen par période.

CHAPITRE 5

LA SIMULATION : LE MODÈLE

5.1 L'organisation générale

5.1.1 4 entités

Le principe d'une entité est qu'elle ne

5.1.2 Pourquoi un registre ?

L'idée d'avoir un registre permet est une double bonne idée. D'une part, on peut garder les informations sur les personnes décédées et cela est particulièrement important pour les successions au second ordre ¹

Deux premières idées : créer un champ petits enfants (avec donc 4 variables un par grand parent-potentiel, mais alors il faut donc avoir éventuellement des arrière-grands-parents, on ne sait jamais. Etc. L'autre option plus sage est de conserver la variables père après le décès des enfants. MAIS pour remonter aux petits-enfants, et faire `invl_pere.inv_lpere`, il faut que la ligne du père existe encore. Cette méthode ne marche pas spontanément.

1. En effet, si un des enfants héritiers de quelqu'un est déjà lui-même décédé, ses enfants à lui deviennent bénéficiaires de l'héritage. Sans registre, comme l'enfant est décédé et donc absent de la base, on ne peut accéder à l'identité des petits-enfants

5.2 Description des étapes

5.2.1 Les étapes de la simulation dynamique

Avant d'entamer cette section, il faut préciser que TaxIPP-Life est encore un projet jeune en développement. Les étapes présentées ici peuvent toutes être améliorées, certaines sont tellement rudimentaires qu'elles n'ont d'autre mérite que d'exister. La description de ces étapes permet toutefois de donner une première idée des fonctionnalités et possibilités de TaxIPP-Life. L'écriture sous forme de modules indépendants permet à tout moment d'améliorer chacune de ces étapes.

5.2.1.1 Démographie

L'âge est bien sûr incrémenté à chaque période.

Naissance Toutes les femmes en couple âgées de 16 à 50 ans peuvent se voir imputer un enfant². Un certain nombre de ces femmes est sélectionné en s'assurant de vérifier les projections de naissance en fonction de l'âge de la mère. Le sexe de l'enfant est choisi aléatoirement.

Le fait de ne pas avoir de probabilité différente pour les femmes selon leurs caractéristiques autre que leur âge est problématique. Le nombre d'enfants ainsi que le niveau d'études devrait intervenir³.

Décès La probabilité de décès ne dépend que de l'âge et du sexe. Le nombre de décès prédit par les projections démographiques de l'Insee est ainsi reproduit. Si une personne décédée vit seule avec des enfants, ces enfants sont attribués à l'autre

2. On élimine les femmes qui ont 8 enfants ou plus ou dont le conjoint est dans cette situation. On pourrait toutefois supprimer sans difficulté cette condition.

3. On ne peut pas pour autant dire que le tirage obtenu est complètement indépendant du niveau d'études et du nombre d'enfants dans la mesure où ceux-ci interviennent dans les équations de mise en couple et de divorce, pré-requis pour se voir imputer des enfants dans TaxIPP-Life.

parent si celui-ci est encore en vie. Si ce n'est pas le cas, ils sont affectés à un ménage spécial sans adulte avec tous les autres enfants dans ce cas.

Union Pour l'instant, dans le modèle, l'union entre deux personnes correspond à la fois à un emménagement et à une union légale. Sans qu'il n'y ait de contre-indication technique à le faire, il n'y a donc pas de création de concubinage. Il en existe toutefois dans la base initiale. Reprenant les équations utilisée par DESTINIE à partir du travail de ? on impute une probabilité de se mettre en couple. Cela est fait d'une part pour les premières unions et d'autre part pour les personnes ayant déjà été en couple séparément selon le sexe, en fonction de l'âge depuis la fin des études, du fait d'avoir un enfant, et de la durée depuis la précédente séparation le cas échéant. En utilisant une méthode d'alignement de Liam2 (voir ? pour une définition et une revue des différents type d'alignements), un tiers de ces personnes est sélectionné⁴. Ensuite, un matching est effectué pour associer les individus en fonction de leur âge, de leur différence d'âge et de leur niveau de diplôme comparé. Encore une fois, ce matching, crucial pour étudier bon nombre de questions de redistribution, par exemple, l'impact de l'endogamie, mériterait d'être affiné. En cas d'union, les personnes rattachées aux nouveaux conjoints (enfants le plus souvent) emménagent avec eux. Idem pour les déclarations fiscales.

Séparation Les séparations se traduisent à la fois par un changement de logement d'un des deux conjoints et par une rupture du contrat (et donc le passage à des déclarations fiscales séparées). La probabilité de rupture dépend du nombre d'enfant du couple, de son ancienneté et de la différence d'âge entre ses membres. Ceci pourrait être régi par des probabilités mais pour l'instant c'est l'homme qui déménage (sauf lorsque le couple habitait chez ses parents à lui) et la femme qui a une nouvelle déclaration⁵. Les enfants et personnes rattachées ne changent de

4. Un calage plus pertinent ne serait pas une mauvaise chose.

5. Dans le cas de couple homosexuel, le choix est fait aléatoirement

logement que s'ils sont liés à la personne qui part sans être liés à la personne qui reste, que ce soit pour le logement ou pour la déclaration fiscale.

5.2.1.2 Logement

Le travail sur le logement est extrêmement rudimentaire. Les déménagements ne se font que lors des unions et séparations et lorsqu'une personne de plus de 24 ans n'est ni personne de référence ni en couple (i.e. vit avec ses parents). Le travail sur les loyers n'a pas encore été fait mais ne devrait pas poser de difficulté majeure.

On pourrait à terme utiliser réellement un parc de logement dont on simulerait la variation chaque année. Lors des déménagements, les ménages choisiraient l'un des logements en fonction de la taille du ménage et du logement ainsi qu'en fonction des revenus et du loyer. On simulerait ainsi des déménagements en fonction d'un surpeuplement ou sous-peuplement ainsi qu'en cas d'évolution dans les revenus. L'idée serait de reproduire des tensions sur le marché du logement.

On impute aussi initialement pour chaque logement un propriétaire. Du travail est encore nécessaire à cette étape mais indubitablement savoir identifier les propriétaires et les locataires sera un plus pour le modèle.

5.2.1.3 Marché du travail

La question de la situation sur le marché de l'emploi est évidemment un élément crucial pour TaxIPP-Life, qui veut étudier la redistribution. C'est à ce niveau que l'investissement le plus important devra être fait. Les équations de transition devraient être estimées le plus précisément possible en tenant compte d'un nombre important de paramètres afin de reproduire la réalité du marché du travail. On est pour l'instant loin d'un niveau satisfaisant mais il sera, là encore, facile, au moins en termes de programmation d'améliorer cette partie du modèle.

Éducation L'entrée sur le marché du travail correspond à la fin des études. L'âge de fin d'études est déterminé à partir de l'âge de fin d'études des parents (au moment de la naissance de l'enfant donc). On peut penser à des modèles plus compliqués de décision (voir ??). Pour l'instant, on impute pour les individus qui ne sont pas encore sortis du système scolaire un âge de fin d'études qui est la moyenne de celui de ses parents auxquels on ajoute un élément correcteur en fonction de leur date de naissance pour tenir compte de l'allongement moyen des études.

Lorsque l'individu est encore en étude, son niveau scolaire est entièrement déterminé par son âge. Ce niveau d'étude servira ensuite à imputer la valeur du transfert en nature associée à l'éducation.

Statut Dans l'idéal, on détermine l'état à la date $T+1$ en fonction de l'état à la date T mais aussi des états précédents pour être plus précis. Pour l'instant deux méthodes simplistes ont été implémentées. La première calcule un score en fonction de l'âge puis tire un nombre de transition en suivant des proportions en input du modèle. La seconde sélectionne aléatoirement 2,5 % des individus actifs et inactifs pour les faire changer de statut. Au passage à l'activité, la catégorie d'emploi est tirée au sort.

Type d'emploi et d'employeur Comme pour le logement, on crée un fichier d'entreprise. Tout est à faire mais on pourrait utiliser les informations disponibles dans l'enquête Patrimoine, par exemple la taille de l'entreprise des salariés ou encore le secteur d'activité. Une des entreprises peut regrouper les indépendants et libéraux. Cet élément n'est pas prioritaire mais l'idée est de pouvoir simuler un marché du travail par secteur ou bien d'avoir un modèle de travailleurs indépendants ou encore de se focaliser sur un secteur d'activité particulier, la médecine par exemple. Enfin, pouvoir simuler ou imputer les taxes sur les entreprises pourraient être envisagé.

Salaire et revenus Les salaires sont pour l'instant imputés à partir d'une équation de Mincer relativement simple. Si la personne a un revenu d'activité initialement, on détermine une productivité individuelle comme étant le rapport entre son salaire et le salaire de l'équation de Mincer. Pour les autres, la productivité est tirée aléatoirement. Par la suite, l'équation de Mincer sera toujours multipliée par la productivité pour déterminer les salaires.

Retraites Le passage à la retraite est pour l'instant entièrement déterminé par l'âge.

5.2.1.4 Épargne et consommation

Un taux d'épargne est simulé. Il faut ensuite déterminer la structure de consommation ainsi que la structure d'épargne. L'achat immobilier demandera une imputation particulière. On se servira de l'enquête Budget des Familles, pour imputer une structure de consommation en catégorie permettant de calculer les taxes sur la consommation.

A partir du nouveau stock d'épargne de la période T , on détermine des revenus du patrimoine à la période $T+1$. Lors du décès d'un individu, on répartit son capital parmi ses héritiers connus dans le modèle (conjoint, enfants et parents). Si l'individu n'est lié à personne au moment de son décès, on considère pour l'instant le capital perdu. Dans un avenir proche, les droits de succession seront aussi calculés.

5.2.1.5 Calcul de la législation

En fin de simulation, la législation est calculée. Cela inclut les cotisations salariales, l'impôt sur le revenu, l'impôt sur la fortune, le bouclier fiscal et les prestations. Les prestations associées au handicap sont déjà simulées mais cette information n'est pas encore dans la projection de la population. Les taxes (habitation et

foncière) sont encore à travailler. Tout le volet santé est aussi à travailler. La simulation peut-être effectué *ex post* ou bien à chaque période. Ces deux options permettent de mesurer l'impact d'une réforme à comportement inchangé mais aussi de permettre des réactions comportementales.

Enfin, les retraites et le chômage devraient être imputés d'ici peu.

CHAPITRE 6

LA SIMULATION : LES DONNÉES

Alexis et Béatrice nous expliquent ici le matching EIC et EP + projection des carrières

Ce chapitre présente les données à partir desquelles ont estimé l'impact du système de retraite français. Il s'agit bien de décrire l'ensemble des données avec leur traitement et pas seulement des données d'enquête utilisées. Pour déterminer l'impact du système de retraite et quantifier les conséquences de réformes, il faut connaître l'ensemble de la carrière des individus. Si on peut mobiliser des données pour connaître les carrières passées, il est nécessaire de procéder à des imputations pour déterminer la fin des carrières qui ne sont pas encore achevées. Ce chapitre se décompose logiquement en deux parties correspondant à chacun de ces deux points.

6.1 Les carrières passées

L'intérêt de la microsimulation est de pouvoir quantifier les effets réels des politiques publiques en tenant compte de la diversité et de la complexité des situations ce que ne permettent pas les études à partir de cas-type. Il est donc important

d'avoir une source représentative de la population, de la population française dans notre cas. L'enquête retenue pour pensipp est l'**enquête patrimoine**.

6.1.1 Intérêt de l'enquête patrimoine

L'enquête patrimoine est une enquête de ménages au niveau nationale. Pensipp s'appuie sur la version 2010 de cette enquête. Comme son nom l'indique, l'objectif premier de cette enquête est de connaître le patrimoine détenu par les ménages. Mais ce qui rend cette enquête intéressante pour Pensipp est la présence dans l'enquête d'une biographie professionnelle. C'est élément est décisif pour pouvoir calculer les pensions de retraite dont le calcul dépend de l'ensemble de la carrière. Outre la carrière et le revenu des individus, il est aussi nécessaire de connaître la situation conjugale des individus en particulier pour calculer les pensions de réversion (les droits connexes, on dit comme ça ?). C'est une justification du recours à une enquête ménage qui contient ce lien systématiquement alors que ce n'est pas le cas de toutes les données administratives et c'est aussi une raison de l'utilisation de l'enquête patrimoine qui non seulement précise la situation du ménage à la date de l'enquête mais pose aussi des questions sur l'historique du couple et les éventuelles précédentes unions.

On trouvera plus d'informations sur le site de l'insee¹. On y apprendra entre autres que la version 2010 de l'enquête contient environ 15 000 ménages contre 10 000 dans les précédentes enquêtes du fait d'une sur-représentation des plus hauts patrimoines.

6.1.2 Limites de l'enquête

Les limites classiques d'une enquête auprès des ménages s'appliquent. En particulier, il s'agit d'une enquête auprès des ménages ordinaires qui exclut donc les

1. METTRE UN LIEN ou bien une autre référence

personnes vivant en foyer, en maison de retraite, en prison, etc.

Mais la principale plus spécifique aux objectifs de pensipp concerne les carrières. Si l'enquête patrimoine donne plus d'information que la plupart des autres enquêtes disponibles en France dans ce domaine, ces informations ne sont pas complètes. En effet, ne sont enregistrées que les étapes de la vie professionnelle s'étalant sur plus d'un an et surtout les revenus ne sont pas connus. C'est pourquoi, nous avons opéré un matching statistique de l'enquête patrimoine avec des données administratives suivant précisément des carrières salariales : l'échantillon inter-régime de cotisants (voir REFERENCE pour plus de détail sur ces données). Ce matching est précisément décrit dans l'annexe 6.1.3. L'idée est d'apparier chaque individu de l'enquête patrimoine à un individu de l'EIC qui lui ressemble (même sexe, même âge, etc.) et ayant la carrière la plus proche possible (avec une définition subtile de la proximité entre deux carrière 6.1.3). Ainsi, on obtient une base de données avec des trajectoires salariales précises et réelle tout en gardant le plus possible les déclarations à l'enquête patrimoine.

6.1.3 Fermeture de l'échantillon

La création de lien parent enfant est un des points techniques importants du modèle. Motivation D'abord, on a besoin des ces liens car on veut créer un système d'héritage, de persistance de la richesse, etc. On en a besoin pour tous les liens intergénérationnels : pension alimentaire, prise en charge de l'éducation, aide pour la dépendance, etc. Il faut aussi avoir en tête que l'on va remonter dans le passé des individus et que a un moment dans leur histoire, les parents d'un enfant ont vécu ensemble, en général. Enfin, ces liens sont centraux parce qu'il nécessite, comme tous les liens entre ménage que les ménages soient de pondération voisine. Sur ce cas précis des liens parents enfants, le fait de répliquer l'échantillon permet d'avoir

un nombre similaire de parents cherchant un enfant et d'enfant cherchant un parent ce qui ne serait pas le cas si on regardait les données d'origine en ne tenant pas compte de leur pondération.

Méthode On parle d'enfant fictif dans cette partie, c'est un abus de langage mais comme tout abus de langage il est pratique. **Principe** On crée deux tables. Dans la première, on met les enfants avec leurs caractéristiques. Dans la seconde, on construit des lignes correspondant aux enfants fictifs des ménages qui déclarent des enfants hors du domicile. Il y a donc un travail de conversion pour passer des parents à l'enfant fictif réalisé à ce moment là. **Sélection** On prend bien sûr tous les enfants fictifs de parents, en veillant à relever à chaque fois si on cherche un enfant du couple, un enfant que d'une personne du couple. Pour les enfants, on utilise les variables *per1e* et *mer1e* qui indique si le parent est vivant et s'il vit à l'extérieur du domicile. On ne sélectionne bien sûr que les individus à qui il faut bien chercher un parent (une Hypothèse consiste à rechercher les parents non connu, en faisant cela, on fait comme si le parent de l'autre côté lui déclarait l'enfant ce qui n'est pas sûr, il n'est pas sûr non plus que le lien soit important pour la transmission, etc dans ce cas très particulier) **Déroulement séquentiel** On pourrait lancer le matching ensuite directement. Seulement, on préfère agir séquentiellement. Dans un premier temps, on cherche les enfants associés à des enfants fictifs de couple. On sait que pour ces EF, il faut absolument chercher des E à qui il manque deux parents. Reciproquement en revanche, un E a qui on cherche deux parents peut très bien avoir un parent dans un ménage et un autre dans un autre. Ensuite, maintenant que le fait que certains parents vivent ensemble a été exploité, on peut chercher un père et une mère relativement successivement pour tout les E. On pourrait exploiter le veuvage pour apparier préférentiellement les individus dont on sait qu'un des parents est décédé. **Variables d'appariement** On utilise les informations intrinsèques des individus comme le sexe et l'âge. On utilise aussi, la situation sur le marché du travail, le diplôme obtenu, le fait de vivre en couple ou non et le nombre d'enfant.

Enfin, important pour la transmission, on utilise aussi l'information selon laquelle les grands-parents sont en vie ou non. On peut faire une première remarque sur ce jeu de variables. On n'utilise ici que des informations sur les enfants, les questions sur l'activité des parents pendant la jeunesse étant un peu compliquée à utiliser puisqu'il faut remonter dans le passé à ce moment là. On a en revanche des informations sur le patrimoine des parents que l'on pourrait utiliser plus facilement, là encore ça peut-être un gain dans l'étude des successions. Une deuxième remarque c'est que l'on utilise de l'information exacte du jeu de données. On pourrait utiliser des informations annexes par exemple liant le revenu des parents ou leur CSP à celle de leur enfant. On verra ci-dessous qu'avec une méthode de score cela peut se faire facilement. Enfin, une remarque importante est que l'on fait confiance au matching via l'enfant pour avoir une distribution jointe des pères et des mères réaliste. Le problème ne se pose que pour les parents ne vivant pas ensemble évidemment. On sait par exemple que dans les couples l'âge du père et de la mère est corrélé et proche. Même si la distribution générale des âges des pères et des mères est conservée, rien ne dit que la corrélation sera bonne. On pourrait donc choisir d'abord les mères par exemple, puis trouver des pères avec un âge proche. On pourrait utiliser des informations extérieures mais aussi utiliser des informations sur le passé matrimonial des parents.

Choix d'appariement On pourrait faire un matching précis sur certaines variables puis en relâcher certaine pour apparier les personnes qui ne l'ont pas été au premier tour et ainsi de suite jusqu'à apparier tout le monde. C'est ce que fait destinie par exemple. Ici, on choisit de construire un score. Les pondérations sont relativement arbitraires même si elles relèvent d'un certain bon sens. Cela ne suffit pas à définir le matching car avec un score, il y a plusieurs façons d'obtenir un matching. Celui que l'on retient est assez basique. Il a l'avantage d'une simplicité dans le temps de calcul, l'inconvénient est par contre qu'il n'est pas optimal en général (pas plus que celui par la méthode précédente d'ailleurs). On balaye simplement

les individus A que l'on veut matcher avec les individus B, pour chacun d'entre eux, on sélectionne le candidat le plus proche que l'on retire ensuite de l'ensemble B. Une variante qui mériterait d'être implémentée consiste à tirer aléatoirement un nombre donné de candidat de B pour chaque A et de sélectionner le meilleur parmi eux. Cela réduit encore le temps de calcul pour les gros échantillons. Et même si on rate le meilleur match pour le premier élément de A par exemple, il sera ainsi le meilleur match de quelqu'un d'autre ce qui peut être un meilleur match global. Le lecteur intéressé pourra lire une annexe sur les techniques de matching qui donne quelques idées générales sur le matching en microsimulation et en général.

ANNEXES : DICTIONNAIRES DES VARIABLES

A. Fichiers sources

TABLEAU 6.1: Titre

Variable	Description
var	Description

B. Fichiers simulation

C. Paramètres

D. Le matching

Matching : Matrice de cout n à affecter et p à être affecté. On dira personnes et jobs par soucis de simplicité. On veut donc minimiser pour résoudre un problème d'assignement. 1) La façon la plus simple de faire un matching à partir d'une matrice c'est de procéder séquentiellement, en choisissant pour chaque ligne la valeur optimale puis en retirant la colonne (l'objet affecté, le job) de la liste des possibilité pour les autres lignes. Evidemment cette méthode, tourne en $O(n \times p)$ mais donne rarement la solution optimale. Même en jouant sur l'ordre, on ne peut pas être sûr

de rien. A choisir vaut mieux traiter les gens de façon à faire tourner d'abord les gens dont le minimum de la ligne est le plus élevé. C'est eux qui vont tirer le coût global vers le haut. Mais pas tellement mieux ? Autre chose, sur une matrice répliquée de taille $N * P$, il faut un temps $N * P$ puisqu'il n'y a pas de raison de prendre l'ordre des n . On peut toutefois se dire que dans ce cas, on est plus proche de l'optimalité puisqu'on laisse à chacun prendre la valeur minimale. Il faudrait calculer si on veut s'amuser, la probabilité d'être affecté à son optimum ? 2) L'autre méthode sans conteste théorique est le recours à une méthode d'assignement optimal. La méthode hongroise est géniale puisqu'elle résout effectivement le problème. Son souci majeur est que pour une matrice carrée, elle tourne en $O(n^3)$. Concrètement, en l'appliquant a priori de bonne façon. Par exemple, en 2013 sur un ordi (config de l'ordi) une matrice de taille 1000 met 22 secondes à tourner. Et même si empiriquement la puissance semble moins grande que 3 ; on arrive vite à de grandes durées, très vite. On aura donc intérêt à faire des catégories assez petites. Cependant, j'ai montré que pour une matrice étendue, on avait au plus un temps $O(n^4)$. Cela dépend en fait du nombre de valeur différente prise par la duplication, nombre qui peut être au plus $n + p$. On aura donc intérêt à dupliquer le plus possible par le même nombre (réhabilite les multiples de 2 tout le temps).

Généralisation de la remarque précédente : chercher les valeurs communes avant d'appliquer l'algorithme ; ça doit se faire en $n^2 + p^2$ cette histoire, donc mieux ? Une autre difficulté est la gestion des matrices non carrées ? La stratégie consistant à ajouter des lignes n'est pas la meilleure probablement. Puisqu'elle fait tourner des calculs sur la matrice la plus grande possible et surtout elle introduit des cas qui sont gourmands en temps. Il faut entrer dans le calcul pour comprendre ça. Bref ça vaut le coup de se pencher à fond sur le problème ?

ANNEXES : DICTIONNAIRES DES VARIABLES

CHAPITRE 7

INSTALLATION

Cette partie explique comment installer tout le programme de Taxipp. On essaie de faire en sorte qu’aucune connaissance préalable ne soit requise.

On définit deux grandes étapes. La première est l’installation des programmes Python et R qui permettront de faire tourner les programmes de Taxipp. On considère qu’il s’agit de l’environnement de l’ordinateur puisqu’il ne s’agit pas de programmes spécifiques à Taxipp de. On donnera ensuite les indications pour installer les éléments de Taxipp proprement dit.

Avant de commencer, précisons que même si on essaie d’être le plus complet possible, il ne faut pas hésiter à utiliser internet en cas de doute (et à mettre à jour ce document).

7.1 Environnement de l’ordinateur

7.1.1 Installer Python

You must install Python 2.7.x in the 64 bits versions¹. Installation files can be found on : <http://www.python.org/download/releases/>.

Make sure both PYTHONDIR and PYTHONDIR\ Scripts is in your system PATH where PYTHONDIR is the directory where you installed Python (C :\ Python by default). It would be usefull later.

Then you need to install all our other dependencies (third-party packages for Python). If you are on Windows, the easiest way to install them is to use the (unoffical) binary packages from Christoph Gohlke at : <http://www.lfd.uci.edu/~gohlke/pythonlibs/>. You can also install first the pip package and then install from a command prompt²

The package linked to the simulation (and Liam2) are :

- Numpy
- PyTables
- Numexpr
- PyYAML
- carray

In addition, to run the legislation (OpenFisca) :

- scipy
- pandas
- rpy2
- PyQt4
- dateutil
- distribute

1. If you're not using a 64-bits version, you may encounter problem in particular for the OpenFisca part.

2. I think it should be done for one of required dependencies because it is not present on Gohlke's page.

- openpyxl
- matplotlib
- pywin32
- ipython

Some of these packages are used only for visual interface of OpenFisca you may don't need. So far, distinction has not been made. Anyways, it's not too costly to install more package.

You're also adviced to install three more dependencies. They can be useful. You're invited to visit easy-to-find corresponding website to install and to use.

- RunSnakeRun : to be an expert in calculus time used by Taxipp .
- Sphinx : to build an write a documentation. For Taxipp , documentation is done in \LaTeX , bu OpenFisca and Liam2 use it.
- Vitables : to see tables. It need to be written here but there is a way to open Vitables easily you should definitely find³.

And just in case, you should know Cython , used by Liam2, than can build a C extension.

7.1.2 Installer R

L'installation de R, ne devrait pas poser de problème. Il faut une version pas trop vieille pour l'utilisation de la librairie rpy2.

Bien sûr, on ne saurait trop recommander l'usage de RStudio, une interface visuelle bien agréable.

3. So far, I use the viewer.py file which I open with python.exe with a shortcut.

7.1.3 Interface Python et R

Pour que Python et R puisse se parler, il faut modifier le path system comme l'explique la documentation du package rpy2. Trouver comment modifier les variables d'environnement associé à la session. Ajouter deux nouvelles variables :

- R_HOME : Path to your R folder (C : \Program Files\R\R-2.15.2 by default).
- R_USER : Your name. I must confess I didn't find what it is for.

7.1.4 Et \LaTeX ? Et Git?

Pour modifier le présent document, vous êtes encouragé à installer \LaTeX avec votre éditeur préféré.

Pour trouver et installer les fichiers de Taxipp et la meilleure solution est d'installer GitHub. Sans que cela soit indispensable, on vous conseille d'utiliser cet outil même au-delà de l'utilisation de Taxipp. Il en sera plus question dans la section suivante.

7.2 Les fichiers de TaxIPP-Life

7.2.1 Où trouver les sources de TaxIPP-Life

On recommande de lire la section sur les répertoires d'installation qui suit celle-là avant de se lancer.

Les fichiers de Taxipp se trouve sur GitHub à l'adresse suivante : <https://github.com/AlexisEidelman/til>. D'ici vous pouvez charger un zip ou bien charger ces éléments à partir de GitHub si vous l'avez installé. Disons en passant, que si vous souhaitez contribuer, d'une manière ou d'une autre au développement de Taxipp, vous pouvez forker le dossier et user des pull request et des issues. Si vous

ne voulez pas utiliser GitHub, que ce ne soit pas un obstacle à une éventuelle contribution. Vous pouvez envoyer vos remarques et fichiers à alexis.eidelman@ipp.eu .

En fait, le dossier du repository "til" n'est pas suffisant. Il faudra aussi, installer OpenFisca que l'on peut trouver aussi sur GitHub : <https://github.com/openfisca/openfisca>. Et de même, il faut aussi installer le fichier correspondant à Liam2. Là, aussi on trouvera sur GitHub, une version modifiée qui adaptée à Taxipp .

Pourquoi pas un seul dossier ? OpenFisca et Liam2 sont des projets dissociés de Taxipp . En les mettant dans des dossiers séparés, on peut, surtout quand on utilise GitHub, continuer à faire des mises-à-jour facilement. On peut aussi faire remonter à ces projets des modifications. Cela est surtout vrai pour OpenFisca qui est sur GitHub alors que Liam2 utilise plutôt des mises-à-jour de temps en temps.

7.2.2 Chemin d'installation

Pour l'instant, on recommande de tout installer à la racine du lecteur C : directement. Il n'est pas impossible de changer la place des dossiers mais pour l'instant, il n'y a pas un fichier de configuration qu'il suffirait de remplir pour que les programmes sachent à chaque fois trouver le bon emplacement.

Le chemin des fichiers src d'OpenFisca et src_liam de Liam2 doivent être ajoutés au PythonPath. Ceci se fait par exemple très facilement pour les utilisateurs d'Aptana.

7.3 Ma première simulation

De là, les choses sont plutôt simples. Il faut avoir fait tourné `run_all` dans le fichier `data\Patrimoine` (en ayant précisé le chemin de l'enquête patrimoine 2010). Puis il faut ouvrir le fichier `pgm\run_til.py` la première fois, il faut décommenter la ligne `main.main()` en début de programme, c'est elle qui fait la conversion des fichiers `Rdata` que vous avez créés juste avant en un fichier `simple2009.h5` à partir duquel le modèle tourne.

GLOSSAIRE



L'Institut des politiques publiques (IPP) est développé dans le cadre d'un partenariat scientifique entre PSE-Ecole d'économie de Paris (PSE) et le Centre de Recherche en Economie et Statistique (CREST). L'IPP vise à promouvoir l'analyse et l'évaluation quantitatives des politiques publiques en s'appuyant sur les méthodes les plus récentes de la recherche en économie.

PSE-Ecole d'économie de Paris regroupe plus de 120 chercheurs, 200 doctorants et 300 étudiants, et constitue un pôle français en science économique de renommée mondiale. PSE a pour objectif premier de fédérer, animer et assurer le rayonnement de ses chercheurs, tout en proposant des formations généralistes et spécialisées à la pointe de la discipline, du M1 au doctorat. La fondation vise également à tisser des liens pérennes entre les différents univers « consommateurs » de savoirs économiques : les acteurs académiques, institutionnels et privés. www.parisschoolofeconomics.eu



PARIS SCHOOL OF ECONOMICS
ECOLE D'ECONOMIE DE PARIS

Le CREST est le centre de recherche du GENES (Groupe des Ecoles Nationales d'Economie et Statistiques) qui est devenu le 1^{er} janvier 2011 un établissement public à caractère scientifique, culturel et professionnel (EPSCP), sous la tutelle technique de l'INSEE (ministère de l'Economie, des Finances et de l'Industrie). Le GENES regroupe quatre établissements : le CREST, le CEPE, l'ENSAE et l'ENSAI. Il a vocation à conduire des travaux de recherche, des missions d'étude ou d'expertise et des actions de diffusion. Il est en outre habilité à développer des dispositifs d'accès aux données, notamment de la statistique publique. www.crest.fr

