SimSAD

Clavet, Hébert, Michaud, Navaux et Raîche

nov. 02, 2023

Contents:

Bienvenue dans la documentation du modèle de projection du soutien à l'autonomie du Québec produit par la Chaire de recherche Jacques-Parizeau en politiques économiques de HEC Montréal.

SimSAD est un modèle de simulations qui a été créé par une équipe de chercheurs sous la direction de Pierre-Carl Michaud et composée de Nicholas-James Clavet, Réjean Hébert, Julien Navaux et Michel Raîche.

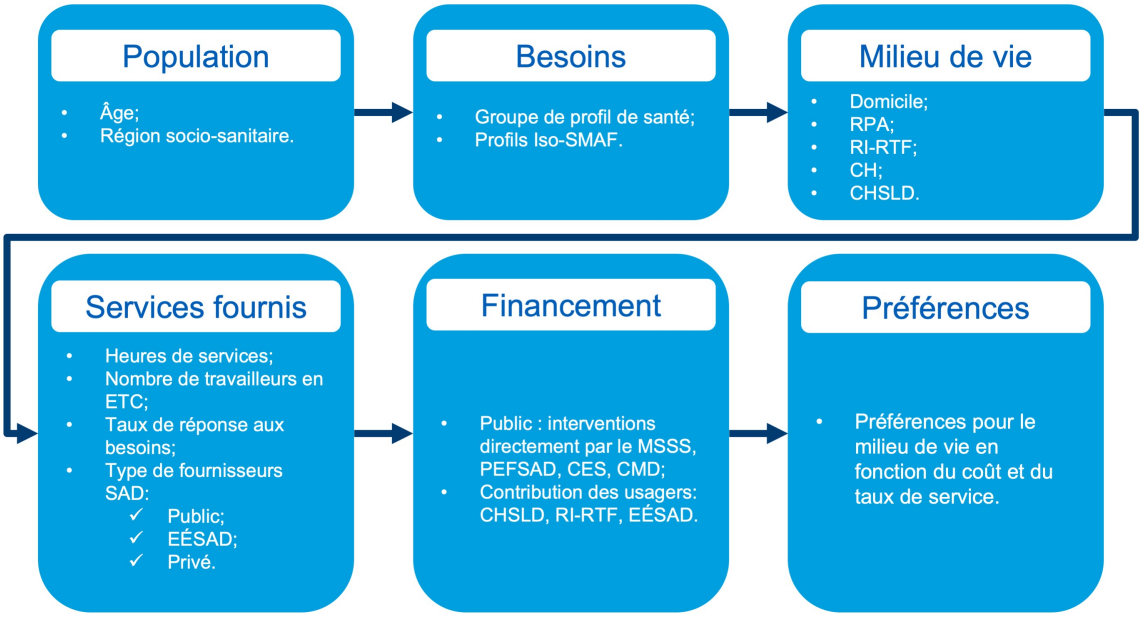
Le modèle SimSAD est issu d'un mandat confié par le Commissaire à la santé et au bien-être (CSBE) à l'équipe de chercheurs, avec l'objectif de modéliser la demande, l'offre et le financement des services de soutien à domicile (SAD) au Québec. La modélisation permet de réaliser des projections dans le futur proche, de réaliser des simulations de scénarios et d'en estimer les impacts. Etant donné que les services de soutien à l'autonomie représentent un continuum du domicile à l'hébergement, les services dispensés dans les autres milieux de vie sont également inclus au modèle. Les résidences pour aînés (RPA), les ressources intermédiaires et de types familiales (RI-RTF) et les centres d'hébergement de soins de longue durée (CHSLD) ont une place importante dans SimSAD.

Pour rester informé.e des mises à jour de SimSAD, inscrivez-vous à notre liste d'envoi dédiée.

CHAPITRE 1

Présentation

Le modèle SimSAD comporte six modules qui sont présentés schématiquement dans la Figure ci-contre. Chacun des modules fait l'objet d'une partie qui lui est dédiée à la suite de la figure.



Population

L'approche retenue quant à l'évolution de la structure d'âge de la population s'appuie sur les projections démographiques de l'Institut de la statistique du Québec (ISQ) publiées dans le cadre des Perspectives démographiques du

Québec et des régions. Parmi ces projections, nous avons choisi celles au niveau des régions sociosanitaires du Québec (RSS). Ces données par âge, année et RSS constituent le premier intrant du modèle.

Besoins

La deuxième étape consiste à déterminer les besoins de soutien à l'autonomie. Nous avons recours aux profils Iso-SMAF qui sont utilisés par le réseau de la santé et des services sociaux pour gérer les services aux personnes présentant des incapacités. La porte d'entrée du système vers le soutien à l'autonomie est l'évaluation des besoins par un intervenant. En se basant sur le fichier du Réseau de services intégrés pour les personnes adultes (RSIPA), croisé avec les données médicales par groupes de profils de santé (GPS), nous imputons la proportion de personnes ayant une évaluation SMAF par âge et région.

Nous imputons par la suite la distribution des profils Iso-SMAF, de 1 à 14 dans pour chacun de ces groupes (âge et région) [1]. Comme présenté dans la figure ci-contre, les profils 1 à 3 sont des profils avec des atteintes principales aux tâches domestiques allant en grandissant. Les profils 4, 6 et 9 sont des profils avec une atteinte surtout motrice, donc physique. Les personnes ayant des profils 5, 7, 8 et 10 ont une atteinte principalement au niveau des fonctions mentales avec des besoins toujours allant en grandissant. Par exemple, ce groupe de profils inclut les personnes souffrant de troubles neurocognitifs à différents stades. Finalement, les profils 11 et plus regroupent des personnes ayant des atteintes mixtes importantes nécessitant de l'aide dans pratiquement toutes les dimensions et étant dépendantes dans plusieurs d'entre elles.

Les profils Iso-SMAF sont rattachés à un nombre d'heures requis de services dans trois domaines, soit les soins infirmiers, les soins d'assistance visant les activités de la vie quotidienne (AVQ) et finalement les activités de la vie domestique (AVD). Ces sont ces trois types de besoins qui servent de base à la modélisation de la demande et de l'offre pour le soutien à l'autonomie.

Milieu de vie

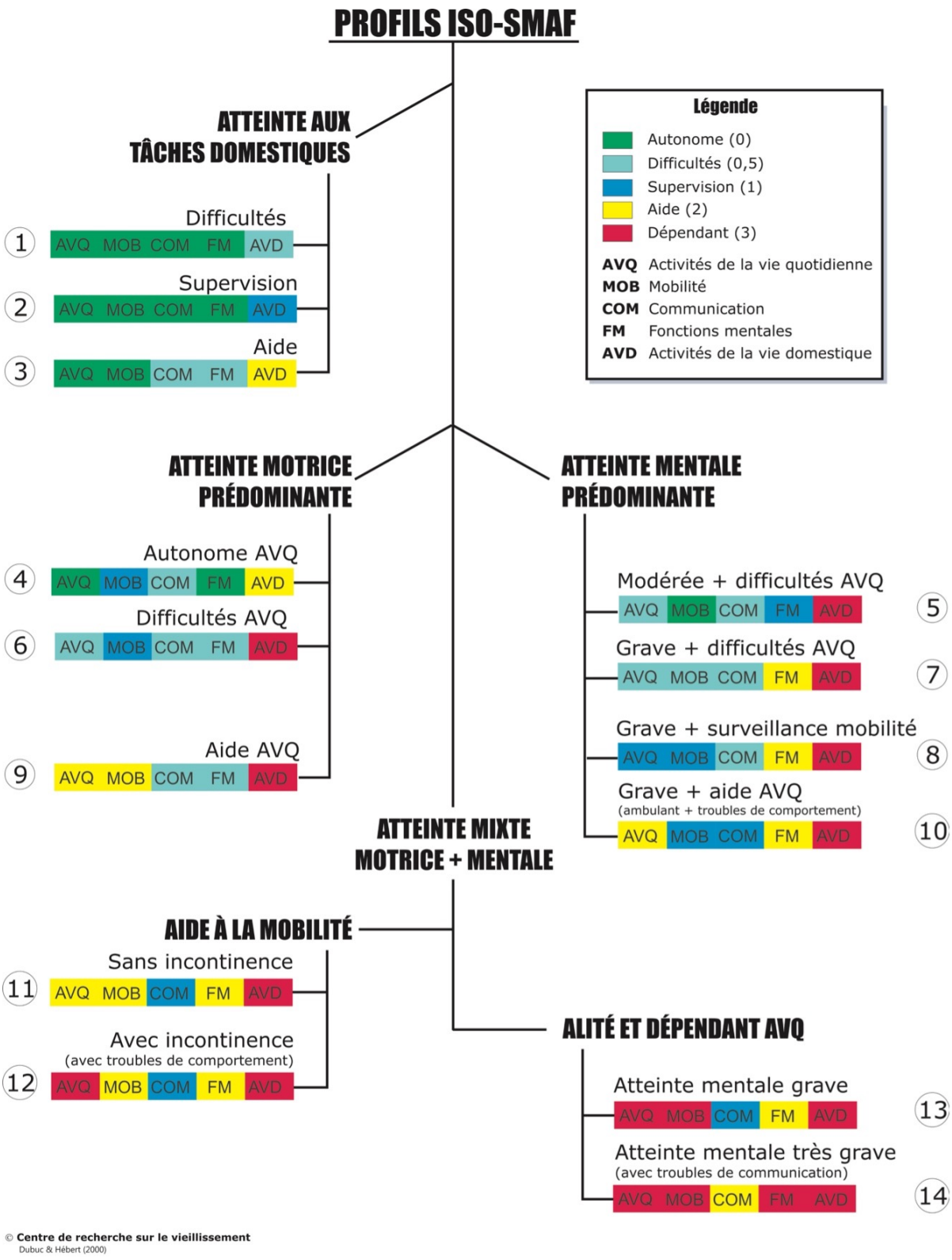
Les personnes ayant un profil Iso-SMAF peuvent résider dans divers milieux de vie. Pour les fins de la modélisation, nous distinguons le domicile privé, la résidence pour personnes âgée (RPA), la ressource-intermédiaire (RI-RTF), le Centre d'hébergement en soins de longue durée (CHSLD) et finalement le centre hospitalier en niveau de soins alternatifs (NSA) [2]. Le croisement de plusieurs fichiers administratifs nous permet d'observer les dates d'entrée et de sortie dans ces différents milieux. La modélisation est réalisée au niveau mensuel pour tenir compte des transitions de milieu de vie à l'intérieure d'une année selon l'âge, la région et le profil Iso-SMAF. Au terme d'une année de simulation, nous sommes en mesure d'obtenir le nombre de personnes par mois dans chacun des milieux de vie. La simulation tient compte des contraintes de capacité en termes de places en RPA, RI-RTF et CHSLD. Elle prend aussi en compte les personnes en attente d'une place dans un milieu de vie. Une description plus détaillée de cette modélisation se trouve dans le rapport méthodologique.

Services fournis

Le modèle tient compte de plusieurs acteurs dans la fourniture des services. Certains fournisseurs sont présents dans un seul milieu de vie, alors que d'autres œuvrent dans plusieurs. En CHSLD, les soins sont fournis directement par la main d'œuvre du MSSS. Pour les RI-RTF, les soins infirmiers sont fournis par le CLSC, alors que les soins d'assistance (AVQ et AVD) sont fournis par les travailleurs des RI-RTF. À domicile, les soins sont fournis par les CLSC, les entreprises d'économie sociale en aide à domicile (EÉSAD), ainsi que par le secteur privé à but lucratif. Notons aussi que dans les CLSC, certains services sont sous-traités aux EÉSAD et au secteur privé à but lucratif. Ces aspects sont pris en compte dans le modèle. La modélisation de l'offre de service est caractérisée en termes de nombre d'heures travaillées, qui se résument en équivalents temps complet de travail (ETC) par fournisseur et par milieu de vie, qui sont calculées selon les 18 régions sociosanitaires du Québec. En plus des heures travaillées en services directs, le modèle considère également les heures de travail indirectes qui incluent du temps de travail administratif et du temps de transport. Le nombre d'heures de services fournis par usager est stable dans le temps selon le profil Iso-SMAF, le milieu de vie et la

1. Cette méthode présente deux limites : 1) le nombre de personnes présentant des besoins ne tient pas compte des personnes n'ayant pas été évaluées mais ayant possiblement des besoins, car il n'a pas été possible de capter ces personnes avec les données disponibles. 2) les proportions et distributions par âge et région qui sont utilisées datent d'avant la pandémie, soit de 2015-2016 à 2019-2020 (d'avril à mars). Il convient aussi de noter qu'il nous a été impossible de déceler des tendances dans ces distributions étant donné la courte période pour laquelle nous avions accès aux données sur les profils Iso-SMAF.

2. Correspond aux personnes qui sont en centre hospitaliers, mais qui sont en attente d'une place dans un autre milieu de vie.


PROFILS ISO-SMAF
ATTEINTE AUX TÂCHES DOMESTIQUES
Difficultés
1 AVQ MOB COM FM AVD
Supervision
2 AVQ MOB COM FM AVD
Aide
3 AVQ MOB COM FM AVD
Légende
Autonome (0)
Difficultés (0,5)
Supervision (1)
Aide (2)
Dépendant (3)
AVQ Activités de la vie quotidienne
MOB Mobilité
COM Communication
FM Fonctions mentales
AVD Activités de la vie domestique
ATTEINTE MOTRICE PRÉDOMINANTE
Autonome AVQ
4 AVQ MOB COM FM AVD
Difficultés AVQ
6 AVQ MOB COM FM AVD
Aide AVQ
9 AVQ MOB COM FM AVD
ATTEINTE MENTALE PRÉDOMINANTE
Modérée + difficultés AVQ
AVQ MOB COM FM AVD 5
Grave + difficultés AVQ
AVQ MOB COM FM AVD 7
Grave + surveillance mobilité
AVQ MOB COM FM AVD 8
Grave + aide AVQ
(ambulant + troubles de comportement)
AVQ MOB COM FM AVD 10
ATTEINTE MIXTE MOTRICE + MENTALE
AIDE À LA MOBILITÉ
Sans incontinence
11 AVQ MOB COM FM AVD
Avec incontinence
(avec troubles de comportement)
12 AVQ MOB COM FM AVD
ALITÉ ET DÉPENDANT AVQ
Atteinte mentale grave
AVQ MOB COM FM AVD 13
Atteinte mentale très grave
(avec troubles de communication)
AVQ MOB COM FM AVD 14
© Centre de recherche sur le vieillissement
Dubuc & Hébert (2000)

région, à moins que le nombre d'ETC du fournisseur soit insuffisant pour donner le même nombre d'heures de services que l'année précédente. Dans cette situation, les heures de services fournis par usager sont réduites proportionnellement à travers tous les usagers de manière à ce que la sommes des ces heures correspondent à la capacité en main d'œuvre du fournisseur. Le modèle permet aux fournisseurs d'augmenter leur nombre d'ETC à un rythme plus ou moins élevé, lorsque des besoins de main d'œuvre supplémentaire surviennent. Le nombre de places qui sont disponibles dans les milieux de vie institutionnels exercent une contrainte sur les capacités d'hébergement, tout particulièrement en CHSLD public pour lesquels les coûts de construction sont assumés par le gouvernement du Québec. Le modèle permet également l'achat de places en CHSLD privé non-conventionné.

Financement

Le calcul de coûts sont réalisés indépendamment pour chaque type de fournisseur : CHSLD, CLSC, RI-RTF, EÉSAD, et le secteur privé à but lucratif. À ces fournisseurs sont également ajoutés les coûts du crédit pour maintien à domicile des aînés (CMD) et les coûts des personnes en attente d'une place qui sont dans un centre hospitalier en NSA. Il est à noter que les coûts liés au Programme d'exonération financière pour les services d'aide domestique (PEFSAD) sont inclus dans les coûts des EÉSAD, et les coûts liés au Chèque emploi-service (CES) sont inclus dans le secteur privé à but lucratif. Des coûts de fonctionnement, qui intègrent des coûts variables et des coûts fixes, sont calculés. Les coûts variables dépendent du nombre d'heures de travail nécessaires pour fournir les soins infirmiers, les soins d'assistance en AVQ et les soins d'assistance en AVD. Des coûts fixes sont également pris en compte en CLSC, en CHSLD et en RI-RTF. Ils correspondent à l'ensemble des coûts qui ne sont pas intégrés aux coûts variables, comme par exemple les coûts d'hébergement, de gestion ou de repas en CHSLD ou les services psychosociaux et d'ergothérapie dispensés par les CLSC à domicile. Des coûts de constructions sont inclus en CHSLD, lorsque de nouvelles places sont ajoutées au parc immobilier existant. De plus, la construction de places entraîne des coûts de fonctionnement supplémentaires qui correspondent aux frais d'intérêts et d'amortissement. Le modèle inclut également les sources de financement des soins de longue durée, que ce soit par le secteur public, donc le gouvernement du Québec, ou bien par les usagers, qui payent par exemple une contribution des adultes hébergés (CAH) en CHSLD et en RI-RTF.

Préférences

Le module de préférences, qui repose sur une nouvelle enquête sur les préférences pour la prise en charge de la perte d'autonomie, a pour objectif d'intégrer les préférences des usagers quant à différents scénarios de prise en charge de la perte d'autonomie. Il est basé sur une méthode de préférences déclarées à partir de différents scénarios de prise en charge présentés au répondant (qui font varier le milieu de vie, le type de fournisseur, le niveau de service offert par rapport aux besoins, le temps d'attente, le prix net payé par l'usager).

CHAPITRE 2

Installation

SimSAD est programmé en langage Python. Il est ainsi nécessaire de posséder la version 3.9 de Python ou une version supérieure pour faire fonctionner SimSAD sur votre ordinateur. Malgré tout, si vous n'avez pas accès au logiciel Python mais que vous disposez d'un compte Google, il vous sera possible d'utiliser SimSAD en accès à distance via Google Colab. Ainsi, il est possible d'avoir accès à SimSAD selon trois méthodes présentées ci-dessous. Dans tous les cas, veuillez lire les condition d'utilisation du site internet pypi qui héberge le package.

2.1 1) Installation automatisée

Si vous avez accès à Python et à votre invite de commande, il est possible d'installer SimSAD de manière automatisée en écrivant simplement cette commande dans l'invite de commande (terminal) :

```
pip install simsad
```

Par la suite, il est possible d'invoquer SimSAD dans un notebook ou un script en tant que module de la manière suivante :

```
import simsad
```

2.2 2) Installation manuelle

Si vous avez accès au logiciel Python, mais que vous ne pouvez utiliser l'invite de commande, il est possible d'installer manuellement SimSAD en complétant les étapes suivantes :

1. Allez sur le site internet Pypi et faites une recherche du package « simsad ».
2. Cliquez sur l'onglet « simsad-x.x.x », où « x.x.x » correspond au numéro de version.
3. Ensuite, cliquez sur « Download files » dans le menu à gauche et puis cliquez sur le nom du fichier « simsad-x.x.x.tar.gz » pour télécharger le fichier compressé.
4. Une fois le fichier téléchargé, décompressez le fichier « simsad-x.x.x.tar.gz » une première fois.
5. Ouvrez le dossier créé par l'extraction (ex. simsad-x.x.x.tar), continuez ensuite en ouvrant le dossier « dist » et décompressez le fichier « simsad-x.x.x.tar ».

6. Une fois le fichier décompressé, transférez le dossier « simsad-x.x.x » dans le dossier où vous entreposez vos packages (si vous n'en avez pas, créez-en un à l'endroit qui vous convient le mieux).
7. Enfin, ajoutez dans votre notebook ou votre script le chemin d'accès de votre dossier de packages et vous pourrez invoquer simsad en tant que module.

```
import sys
sys.path.append('.../packages')

import simsad
```

2.3 3) Accès à distance

Si vous ne possédez pas ou ne pouvez pas avoir accès au logiciel Python, il est possible d'utiliser SimSAD par l'entremise de Google Colab. Après avoir ouvert un compte Google Colab et avoir créé un nouveau notebook, vous n'avez qu'à utiliser la commande suivante pour installer SimSAD :

```
pip install simsad
```

Par la suite, il est possible d'invoquer SimSAD dans un notebook ou un script en tant que module.

```
import simsad
```

CHAPITRE 3

Utilisation

Cette section a pour objectif de guider les utilisateurs de SimSAD dans l'utilisation de celui-ci. Pour davantage d'information, la section *Dictionnaire (classes et fonctions)* donne une description des classes et des fonctions du modèle. Lors de la rédaction d'un notebook ou d'un script Python, cinq étapes principales doivent être suivies afin d'obtenir des résultats de simulation avec SimSAD :

3.1 1. Importation de SimSAD

La première étape est d'importer les modules de SimSAD comme suit :

```
from SimSAD import projection, policy
```

3.2 2. Choix des paramètres

Ensuite, on fixe l'année de début et de fin des projections :

```
annee_debut = 2023
annee_fin = 2040
```

On choisit les paramètres du scénario, qui par défaut sont les suivants :

```
params = policy()

params.chsld_build = True
params.chsld_build_rate = 0.2
params.chsld_restricted_eligibility = False
params.chsld_restriction_rate = 0.95
params.ri_build = True
params.ri_build_rate = 0.2
```

(suite sur la page suivante)

(suite de la page précédente)

```
params.rpa_penetrate = False
params.rpa_penetrate_rate = 0.25
params.rpa_adapt_rate = 0.5
params.chsld_purchase = True
params.chsld_purchase_rate = 0.25
params.nsa_open_capacity = 0.06
params.chsld_mda = True
params.infl_construction = 0.01
params.interest_rate = 0.03
params.clsc_cap = True
params.prive_cap = True
params.eesad_cap = True
params.purchase_prive = True
params.purchase_eesad = True
params.clsc_inf_rate = 0.25
params.clsc_avq_rate = 0.25
params.clsc_avd_rate = 0.25
params.eesad_avd_rate = 0.25
params.prive_avq_rate = 0.25
params.prive_avd_rate = 0.25
params.chsld_inf_rate = 1.0
params.chsld_avq_rate = 1.0
params.ri_avq_rate = 1.0
params.ri_avd_rate = 1.0
params.delta_inf_rate = 0.0
params.delta_avq_rate = 0.0
params.delta_avd_rate = 0.0
params.delta_cah_chsld  = 0.0
params.delta_cah_ri  = 0.0
params.clsc_shift_avq_eesad = 0.0
params.clsc_shift_avq_prive = 0.0
params.clsc_shift_avd_eesad = 0.0
params.clsc_shift_avd_prive = 0.0
```

La commande suivante crée un gabarit permettant entre autres de comptabiliser les résultats propres à la simulation selon les paramètres d'utilisation choisis :

```
scenario = projection(base_yr = annee_debut, stop_yr = annee_fin, scn_policy = params,␣
↪opt_welfare=True)
```

où *base_yr* correspond à l'année à partir de laquelle les résultats de projection sont rapportés, *stop_yr* correspond à l'année de fin de la simulation, *scn_policy* correspond à l'ensemble des paramètres spécifiés pour le scénario, et *opt_welfare* correspond à l'option de calcul de l'utilité des individus [1]. Cette option doit être fixée à *True* pour que cette opération s'effectue, puisque celle-ci est fixée à *False* par défaut (le calcul de l'utilité augmente le temps d'exécution des projections). En pratique, la simulation débute en 2020, car les données initiales du modèle sont de 2019, mais les résultats ne sont rapportés dans les tableaux de sortie qu'à partir de l'année *base_yr*. Si les arguments *base_yr* et *stop_yr* ne sont pas spécifiés, SimSAD produira par défaut des projections allant de 2023 à 2040.

1. voir la partie 7 et en particulier la partie 7.3 « Utiliser les estimations pour évaluer le bien-être » de la documentation technique pour obtenir plus de détails à propos du calcul de l'utilité dans SimSAD.

3.3 3. Ajout d'un tableau de résultats

Plusieurs tableaux de sortie de résultats existent par défaut, mais ceux-ci ne contiennent pas l'ensemble des valeurs évaluées par le modèle lors des projections. Cette approche a été adoptée afin d'améliorer la rapidité d'exécution des projections. Plus le nombre de tableaux de sortie est élevé, plus le temps d'exécution des projections augmente. Voici la liste des noms des 13 tableaux de sortie par défaut et les informations qu'ils contiennent :

0. **pop_region_age** : nombre de personnes (par région et âge) ;
1. **smaf_region_age** : nombre de personnes avec un profil Iso-SMAF donné (par région) ;
2. **chsld_users** : nombre total d'usagers en CHSLD, taux de réponse aux besoins en soins infirmiers dans ce milieu de vie, taux de réponse aux besoins en assistance aux AVQ dans ce milieu de vie, et nombre de personnes en attente d'une place en CHSLD (par région) ;
3. **nsa_users** : nombre total d'usagers en NSA (par région) ;
4. **ri_users** : nombre total d'usagers en RI-RTF (par région) ;
5. **rpa_users** : nombre total d'usagers en RPA avec services financés par le secteur public (par région) ;
6. **home_none_users** : nombre total de personnes avec un profil Iso-SMAF vivant à domicile et ne recevant aucun services de soutien (par région) ;
7. **home_svc_users** : nombre total de personnes avec un profil Iso-SMAF vivant à domicile et recevant des services de soutien à domicile (par région) ;
8. **ces_users** : nombre total d'usagers du CES, nombre total d'heures de services en soutien aux AVD reçus dans le cadre du CES, et nombre total d'heures de services d'assistance aux AVQ reçus dans le cadre du CES (par région) ;
9. **pefsad_users** : nombre total d'usagers du PEFSAD, et nombre total d'heures de services en soutien aux AVD reçus dans le cadre du PEFSAD (par région) ;
10. **clsc_workforce** : nombre total d'ETC en soins infirmiers en CLSC, nombre total d'ETC en assistance aux AVQ en CLSC, et nombre total d'ETC en soutien aux AVD en CLSC (par région) ;
11. **total_cost** : coûts totaux par fournisseurs (CLSC, CHSLD, RI-RTF, NSA) et programmes (CES, PEFSAD, CMD), contributions totales des usagers par milieu de vie (RI-RTF, CHSLD, NSA), coûts totaux pour le gouvernement, coûts totaux pour les usagers, et sommes de l'ensemble des coûts (par région) ;
12. **clsc_worker_needs** : besoin de recrutement en nombre total d'ETC pour les soins infirmiers, l'assistance aux AVQ, et le soutien aux AVD (par région) ;

Ces tableaux de sortie font appel à différent objets du modèle : *pop.count*, *iso.count_eval*, *registry*, et *users*. L'objet *pop.count* contient des informations uniquement liées au nombre de personnes dans la population. L'objet *iso.count_eval* contient, pour sa part, les informations sur le nombre de personnes par profil Iso-SMAF. L'objet *regitry* contient les informations se rattachant aux caractéristiques des milieux de vie et des fournisseurs. Ainsi, il existe un *registry* pour chaque milieu de vie et chaque fournisseur (ex. chsld.registry, clsc.regitry, ets.). L'objet *users* contient les informations sur les usagers de chacun des milieux de vie (Domicile, RPA, RI-RTF, CHSLD, NSA) dans 5 objets différents (home.users, rpa.users, ri.users, chsld.users, nsa.user). Ces informations sont exprimées sous la forme d'une base de microdonnées synthétique où chaque individu a des caractéristiques déterminées par le modèle. L'ensemble de ces objets contiennent seulement des informations par rapport à l'année en cours dans la simulation (à la fin de la simulation **les objets contiennent les informations de la dernière année de simulation**). Pour connaître les variables contenues dans chacun des objets, vous pouvez par exemple utiliser les commandes suivantes pour afficher les différentes variables incluses dans les objets :

```
scenario.clsc.registry.columns
```

ou

```
scenario.home.users.columns
```

La dernière commande fonctionnera seulement lorsque la simulation des projections aura été complétée au moins une première fois. En effet, les *users* sont uniquement créés au cours de la simulation, alors que les *registry* sont créés en même temps que le gabarit de la simulation.

→ **Commandes pour ajouter un tableau de résultats** :

Pour ajouter un tableau de sortie sur le nombre d'ETC en RI-RTF par région et qui récolte des résultats **pour chaque année** entre *annee_debut* et *annee_fin*, il vous suffit d'ajouter dans votre notebook le code suivant :

```
scenario.tracker.add_entry('ri_workforce', 'ri', 'registry',
                           rowvars=['region_id'],
                           colvars=['nb_etc_ri_avq', 'nb_etc_ri_avd'],
                           aggfunc='sum',
                           start_yr=annee_debut, stop_yr=annee_fin)
```

où *ri_workforce* correspond au nom du tableau de sortie, *ri* correpond à la classe de provenance de l'objet utilisé, *registry* correpond à l'objet d'où les résultats sont puissés, *rowvars* correspond aux variables d'agrégation (["region_id"]), *colvars* correspond aux variables de résultat (["nb_etc_ri_avq", "nb_etc_ri_avd"]), *aggfunc* correspond à la fonction d'agrégation (*sum* pour le total ou *mean* pour la moyenne), *start_yr* correspond à l'année de départ de la comptabilisation des résultats, et *stop_yr* correspond à l'année de fin de la comptabilisation des résultats.

3.4 4. Simulation du scénario

Pour lancer la simulation, il vous suffit d'utiliser la commande suivante :

```
scenario.run()
```

3.5 5. Appel des tableaux de résultats

Une fois la simulation terminée, vous pouvez appeler directement le tableau de sortie via son numéro de *tracker*. Par exemple si vous souhaitez appeler le tableau home_svc_users dont le numéro de *tracker* est 7, vous pouvez l'afficher à l'aide du code suivant :

```
df = scenario.tracker.registry[7].table
df
```

Ce tableau de sortie vous donnera le nombre d'usagers à domicile recevant des services de soutien à domicile par région. Veuillez noter que les tableaux que vous aurez créés prendrons la suite des tableaux existants numérotés de 0 à 12. Par conséquent le premier tableau que vous aurez créé prendra le numéro 13, le deuxième tableau que vous aurez créé prendra le numéro 14 et ainsi de suite.

Si vous ne connaissez pas le numéro du *tracker*, mais que vous connaissez le nom de ce dernier, vous pouvez l'afficher à l'aide du code suivant :

```
for k in p.tracker.registry:
  if k.entry_name=='home_svc_users':
      df=k.table
df
```

CHAPITRE 4

Dictionnaire (classes et fonctions)

4.1 Simulation

La classe *projection* permet de réaliser les simulations.

simsad.project.**projection**(*start_yr=2020*, *stop_yr=2040*, *base_yr=2023*, *scn_policy=None*, *scn_name='reference'*, *opt_welfare=False*, *seed=1234*)

Modèle de simulation SimSAD.

Cette classe permet de créer une instance de modèle pour la simulation.

Paramètres

- **start_yr** (*int*) – année de départ de la simulation (défaut=2020)
- **stop_yr** (*int*) – année de fin de la simulation (défaut=2040)
- **base_yr** (*int*) – année de départ pour la comptabilisation des résultats (défaut=2023)
- **scn_name** (*string*) – nom du scénario (défaut="reference")
- **opt_welfare** (*boolean*) – paramètre d'activation du calcul de l'utilité des individus (défaut=False)
- **seed** (*float*) – spécifie la valeur de départ de la séquence aléatoire (défaut=1234)

Cette classe contient les fonctions suivantes (cliquez sur le nom de la fonction pour afficher les détails) :

- load_params()

simsad.project.projection.**load_params**(*self*)

Chargement des paramètres des différentes composantes du modèle.

- load_pop()

simsad.project.projection.**load_pop**(*self*, *geo='RSS'*)

Chargement des paramètres démographiques.

Paramètres

geo (*str*) – unité géographique des projections démographiques (défaut="Québec")

- load_grouper()

simsad.project.projection.**load_grouper**(*self*, *set_yr=2016*)

Chargement des paramètres d'attribution des groupes de profil de santé (GPS).

Paramètres
set_yr (*int*) – année de référence (défaut=2016)

- load_smaf()

simsad.project.projection.**load_smaf**(*self*, *set_yr=2016*)

Chargement des paramètres d'attribution des profil Iso-SMAF.

Paramètres
set_yr (*int*) – année de référence (défaut=2016)

- init_tracker()

simsad.project.projection.**init_tracker**(*self*, *scn_name*)

Fonction qui spécifie les tableaux de sortie de base dans le modèle.

Paramètres
scn_name (*str*) – nom du scénario (défaut="reference")

- init_dispatch()

simsad.project.projection.**init_dispatch**(*self*, *init_smafs*)

Fonction qui formate les paramètres d'attribution des milieux de vie.

Paramètres
init_smafs (*dataframe*) – nombre de personnes par profil Iso-SMAF selon la région et le groupe d'âge

- dispatch()

simsad.project.projection.**dispatch**(*self*)

Fonction qui attribue les milieux de vie aux personnes.

- create_users()

simsad.project.projection.**create_users**(*self*)

Fonction qui crée les bassins d'individus pour l'ensemble des milieux de vie.

- update_users()

simsad.project.projection.**update_users**(*self*)

Fonction qui met à jour les caractéristiques des individus dans les bassins pour l'ensemble des milieux de vie.

- welfare()

simsad.project.projection.**welfare**(*self*)

Fonction qui calcule l'utilité des individus.

- finance()

simsad.project.projection.**finance**(*self*)

Fonction qui collige les coûts pour tous les milieux de vie et les programmes de financement.

- run()

simsad.project.projection.**run**(*self*)

Fonction déclenchant le lancement de la simulation.

- compute()

simsad.project.projection.**compute**(*self*)

Fonction qui appelle l'ensemble des fonctions permettant de calculer les résultats du modèle.

- save()

simsad.project.projection.**save**(*self*, *output_dir*)

Fonction qui permet de sauvegarder les données d'un scénario.

Paramètres

output_dir (*str*) – sentier vers le dossier de sauvegarde des données.

- load()

simsad.project.projection.**load**(*self*)

Fonction qui permet de charger les données d'un scénario préalablement sauvegardé.

La classe *policy* permet de choisir les paramètres du scénario simulé.

simsad.policy.**policy**()

Paramètres du scénario

Cette classe permet de choisir les paramètres du scénario simulé.

Paramètres

- **chsld_build** (*boolean*) – True si on permet la construction de places en CHSLD (défaut=True)
- **chsld_build_rate** (*float*) – taux de construction de places en CHSLD durant une année par rapport aux besoins de places dans ce milieu de vie (défaut=0.2)
- **chsld_restricted_eligibility** (*boolean*) – True si l'admissibilité au CHSLD est limitée pour les personnes avec un profil Iso-SMAF inférieur à 10 (défaut=False)
- **chsld_restriction_rate** (*float*) – taux de restriction des probabilités de transition vers les CHSLD pour les personnes avec un profil Iso-SMAF inférieur à 10 (défaut=0.95)
- **ri_build** (*boolean*) – True si on permet la construction de places en RI-RTF (défaut=True)
- **ri_build_rate** (*float*) – taux de développement de places en RI-RTF durant une année par rapport aux besoins de places dans ce milieu de vie (défaut=0.2)
- **rpa_penetrate** (*boolean*) – True si on permet le développement de nouvelles places en RPA financées par le public (défaut=False)
- **rpa_penetrate_rate** (*float*) – Proportion maximale de places en RPA financées par le public (défaut=0.25)
- **rpa_adapt_rate** (*float*) – taux de transformation des places de RPA en place subventionnée par rapport au nombre de personnes en attente d'une place subventionnée en RPA (défaut=0.5)
- **chsld_purchase** (*boolean*) – True si on permet l'achat de places en CHSLD privé non-conventionné (défaut=True)
- **chsld_purchase_rate** (*float*) – taux d'achat supplémentaire de places en CHSLD non-conventionné durant une année par rapport aux besoins de places en CHSLD (défaut=0.25)
- **nsa_open_capacity** (*float*) – proportion maximale de lit d'hôpitaux pouvant être occupés par les personnes en NSA (défaut=0.06)
- **chsld_mda** (*boolean*) – True si la construction des nouvelles places en CHSLD se fait selon le modèle des maisons des aînés (défaut=True)
- **infl_construction** (*float*) – taux d'inflation des coûts de construction des CHSLD en dollar constant (défaut=0.01)
- **interest_rate** (*float*) – taux d'intérêt en dollar constant (défaut=0.03)

— **clsc_cap** (*boolean*) – True si limitation des heures de services fournis en CLSC selon la capacité en main d'oeuvre de ce fournisseur (défaut=True)
— **prive_cap** (*boolean*) – True si limitation des heures de services fournis par le privé selon la capacité en main d'oeuvre de ce fournisseur (défaut=True)
— **eesad_cap** (*boolean*) – True si limitation des heures de services fournis par les EÉSAD selon la capacité en main d'oeuvre de ce fournisseur (défaut=True)
— **purchase_prive** (*boolean*) – True si achat d'heures de services fournis auprès du privé par les CLSC (défaut=True)
— **purchase_eesad** (*boolean*) – True si achat d'heures de services fournis auprès des EÉSAD par les CLSC (défaut=True)
— **clsc_inf_rate** (*float*) – taux d'ajustement de la main d'oeuvre par rapport aux besoins supplémentaires en main d'oeuvre pour les soins infirmiers en CSLC (défaut=0.25)
— **clsc_avq_rate** (*float*) – taux d'ajustement de la main d'oeuvre par rapport aux besoins supplémentaires en main d'oeuvre pour le soutien aux AVQ en CSLC (défaut=0.25)
— **clsc_avd_rate** (*float*) – taux d'ajustement de la main d'oeuvre par rapport aux besoins supplémentaires en main d'oeuvre pour le soutien aux AVD en CSLC (défaut=0.25)
— **eesad_avd_rate** (*float*) – taux d'ajustement de la main d'oeuvre par rapport aux besoins supplémentaires en main d'oeuvre pour le soutien aux AVD en EÉSAD (défaut=0.25)
— **prive_avq_rate** (*float*) – taux d'ajustement de la main d'oeuvre par rapport aux besoins supplémentaires en main d'oeuvre pour le soutien aux AVQ au privé (défaut=0.25)
— **prive_avd_rate** (*float*) – taux d'ajustement de la main d'oeuvre par rapport aux besoins supplémentaires en main d'oeuvre pour le soutien aux AVD au privé (défaut=0.25)
— **chsld_inf_rate** (*float*) – taux d'ajustement de la main d'oeuvre par rapport aux besoins supplémentaires en main d'oeuvre pour les soins infirmiers en CHSLD (défaut=0.25)
— **chsld_avq_rate** (*float*) – taux d'ajustement de la main d'oeuvre par rapport aux besoins supplémentaires en main d'oeuvre pour le soutien aux AVQ en CHSLD (défaut=0.25)
— **ri_avq_rate** (*float*) – taux d'ajustement de la main d'oeuvre par rapport aux besoins supplémentaires en main d'oeuvre pour le soutien aux AVQ en RI-RTF (défaut=0.25)
— **ri_avd_rate** (*float*) – taux d'ajustement de la main d'oeuvre par rapport aux besoins supplémentaires en main d'oeuvre pour le soutien aux AVD en RI-RTF (défaut=0.25)
— **delta_inf_rate** (*float*) – variation du taux de réponse aux besoins en soins infirmiers au SAD pour les personnes à domicile (défaut=0.0)
— **delta_avq_rate** (*float*) – variation du taux de réponse aux besoins en soutien aux AVQ au SAD pour les personnes à domicile (défaut=0.0)
— **delta_avd_rate** (*float*) – variation du taux de réponse aux besoins en soutien aux AVD au SAD pour les personnes à domicile (défaut=0.0)
— **delta_cah_chsld** (*float*) – variation du taux de contribution des usagers en CHSLD (défaut=0.0)
— **delta_cah_ri** (*float*) – variation du taux de contribution des usagers en RI-RTF (défaut=0.0)
— **clsc_shift_avq_eesad** (*float*) – proportion supplémentaire d'heures de services fournis en soutien aux AVQ par les CLSC transférées en achat de services aux EÉSAD (défaut=0.0)
— **clsc_shift_avq_prive** (*float*) – proportion supplémentaire d'heures de services fournis en soutien aux AVQ par les CLSC transférées en achat de services au privé (défaut=0.0)
— **clsc_shift_avd_eesad** (*float*) – proportion supplémentaire d'heures de services fournis en soutien aux AVD par les CLSC transférées en achat de services aux EÉSAD (défaut=0.0)
— **clsc_shift_avd_prive** (*float*) – proportion supplémentaire d'heures de services fournis en soutien aux AVD par les CLSC transférées en achat de services au privé (défaut=0.0)

La classe *isq* permet de modéliser la démographie.

simsad.demo.**isq**(*region='Québec'*, *pop_fichier='pop-isq.csv'*, *start_yr=2020*, *stop_yr=2040*)

Démographie

Cette classe permet de modéliser la démographie.

Paramètres

- **region** (*str*) – unité géographique des projections démographiques (défaut="Québec")
- **pop_fichier** (*str*) – nom du fichier contenant les projections démographiques (défaut="pop-isq.csv")
- **start_yr** (*int*) – année de départ de la simulation (défaut=2020)
- **stop_yr** (*int*) – année de fin de la simulation (défaut=2040)

Cette classe contient la fonction suivante :

- evaluate()

simsad.demo.isq.**evaluate**(*self*, *yr*)

Fonction qui calcule le nombre de personnes par région et âge durant l'année de simulation en cours.

yr : int
année en cours de la simulation

La classe *gps* permet de modéliser les groupes de profil de santé.

simsad.demo.**gps**(*nregions*, *ages*, *ngps=16*)

Groupes de profil de santé (GPS).

Cette classe permet de modéliser les groupes de profils de santé.

Paramètres

- **nregions** (*int*) – nombre de régions
- **ages** (*int*) – liste des tranches d'âge considérées dans la modélisation
- **ngps** (*int*) – nombre de Groupes de profils de santé (défaut=16)

Cette classe contient les fonctions suivantes :

- load()

simsad.demo.gps.**load**(*self*, *set_yr=2016*)

Fonction qui permet de charger les paramètres liés aux GPS.

Paramètres
start_yr (*int*) – année de référence (défaut=2016)

- evaluate()

simsad.demo.gps.**evaluate**(*self*, *pop*, *yr*)

Fonction qui calcule le nombre de personnes par GPS et région durant l'année de simulation en cours.

pop : objet
objet contenant les informations démographiques

La classe *smaf* permet de modéliser les profils Iso-SMAF.

simsad.demo.**smaf**(*nregions*, *ages*, *ngps=16*, *nsmaf=14*)

Profils Iso-SMAF.

Cette classe permet de modéliser les profils Iso-SMAF.

Paramètres
- **nregions** (*int*) – nombre de régions
- **ages** (*int*) – liste des tranches d'âge considérées dans la modélisation
- **ngps** (*int*) – nombre de groupes de profils de santé (défaut=16)
- **nsmaf** (*int*) – nombre de profils Iso-SMAF (défaut=14)

Cette classe contient les fonctions suivantes :

- load()

simsad.demo.smaf.**load**(*self*, *set_yr=2015*)

Fonction qui permet de charger les paramètres liés aux profils Iso-SMAF.

Paramètres
start_yr (*integer*) – année de référence (défaut=2015)

- evaluate()

simsad.demo.smaf.**evaluate**(*self*, *grouper*, *gps_target=[]*, *rate=0.0*)

Fonction qui calcule le nombre de personnes par profil Iso-SMAF, GPS, région et âge durant l'année de simulation en cours

grouper : objet
objet contenant les informations sur les GPS

gps_target : series
liste de GPS pour lesquels la probabilité d'être évalué pour un profil Iso-SMAF sera augmentée

rate : float
taux de croissance de la probabilité d'être évalué pour un profil Iso-SMAF

La classe *needs* permet de quantifier les besoins en heures de soutien (soins infirmiers, les AVQ et les AVD).

simsad.needs.**needs**()

Besoins en heures de soutien à l'autonomie

Cette classe permet de quantifier les besoins en heures de soutien (soins infirmiers, les AVQ et les AVD) par profil Iso-SMAF.

La classe *dispatcher* permet d'attribuer le milieux de vie aux personnes avec un profil Iso-SMAF.

simsad.dispatch.**dispatcher**()

Milieux de vie

Cette classe permet d'attribuer les milieux de vie aux personnes avec un profil Iso-SMAF.

Cette classe contient les fonctions suivantes :

- setup_milieux()

simsad.dispatch.dispatcher.**setup_milieux**(*self*)

Cette fonction spécifie les milieux de vie.

- setup_ages()

simsad.dispatch.dispatcher.**setup_ages**(*self*)

Cette fonction spécifie les groupes d'âge.

- setup_capacity()

simsad.dispatch.dispatcher.**setup_capacity**(*self*, *n_rpa*, *n_ri*, *n_nsa*, *n_chsld*)

Cette fonction spécifie les capacités maximales par milieu de vie.

- setup_params()

simsad.dispatch.dispatcher.**setup_params**(*self*, *init_prob*, *trans_prob*, *surv_prob*, *wait_count*, *wait_prob_chsld*, *wait_prob_ri*)

Cette fonction spécifie les différentes probabilités utilisées pour l'attribution des milieux de vie.

Paramètres

- **init_prob** (*array*) – répartition au début de l'année des individus par milieu de vie
- **trans_prob** (*array*) – probabilité de transition
- **surv_prob** (*array*) – probabilités de survie
- **wait_count** (*array*) – nombre de personnes en attente pour chacun des milieux de vie
- **wait_prob_chsld** (*array*) – probabilité de provenir d'un milieu de vie donné lorsqu'admis en CHSLD
- **wait_prob_ri** (*array*) – probabilité de provenir d'un milieu de vie donné lorsqu'admis en RI-RTF

- chsld_restriction()

simsad.dispatch.dispatcher.**chsld_restriction**(*self*, *policy*)

Cette fonction empêche les transitions vers les CHSLD à partir d'un autre milieu de vie pour les personnes avec un profil Iso-SMAF de moins de 10, lorsque le paramètre « chsld_restriction_rate » est True.

Paramètres

policy (*object*) – Paramètres du scénario

- marginal_effect()

simsad.dispatch.dispatcher.**marginal_effect**(*self*, *policy*, *pref_pars*, *cah_ri*, *cah_chsld*)

Cette fonction permet l'ajustement des transitions par milieux de vie, lorsqu'on fait varier le taux de services fournis dans le SAD hors-RPA ou lorsqu'on fait varier la contribution d'adulte hébergé en RI-RTF et CHSLD.

Paramètres

- **policy** (*object*) – paramètres du scénario
- **pref_pars** (*dataframe*) – paramètres de préférences estimés
- **cah_ri** (*float*) – valeur initiale de la CAH en RI-RTF
- **cah_chsld** (*float*) – valeur initiale de la CAH en CHSLD

- init_state()

simsad.dispatch.dispatcher.**init_state**(*self*)

Fonction qui répartit les personnes par milieu de vie au début de l'année.

- next_state()

simsad.dispatch.dispatcher.**next_state**(*self*, *m*)

Fonction qui effectue la transition d'un mois à un autre.

Paramètres

m (*int*) – mois pour lequel on calcul le nombre de personnes par milieu de vie

- assign()

simsad.dispatch.dispatcher.**assign**(*self*)

Fonction qui enclenche le calcul du nombre de personnes par milieu de vie pour tous les mois.

- collect()

simsad.dispatch.dispatcher.**collect**(*self*)

Fonction qui comptabilise les personnes par milieu de vie, ainsi que les personnes en attente pour ceux-ci.

La classe *msss* permet de comptabiliser les coûts pour le ministère de la Santé et des services sociaux.

simsad.msss.**msss**()

Ministère de la Santé et des Services sociaux

Cette classe permet de comptabiliser les coûts pour le ministère de la Santé et des Services sociaux.

Cette classe contient les fonctions suivantes :

- assign()

simsad.msss.msss.**assign**(*self*, *cost*, *item*)

Cette fonction assigne les coûts des différents items au registre du MSSS.

Paramètres

— **cost** (*float*) – valeur du coût de l'item

— **item** (*string*) – nom de l'item

- collect()

simsad.msss.msss.**collect**(*self*)

Cette fonction calcule les coûts agrégés pour le gouvernement, les usagers, ainsi que les coûts totaux.

4.2 Milieux de vie

4.2.1 Domicile

La classe *home* permet de modéliser les personnes vivant à domicile (avec ou sans services financés par le réseau public).

simsad.home.**home**(*policy*)

Personnes à domicile

Cette classe permet de modéliser les personnes à domicile.

Paramètres

policy (*object*) – paramètres du scénario

Cette classe contient les fonctions suivantes :

- load_register()

simsad.home.home.**load_register**(*self*)

Fonction qui permet de créer le registre des personnes à domicile. Ce registre contient des informations sur le nombre de personnes et les profils Iso-SMAF de celles-ci.

- assign()

simsad.home.home.**assign**(*self*, *applicants_none*, *applicants_svc*, *waiting_users*, *region_id*)

Fonction qui répertorie dans le registre des personnes à domicile le nombre de personnes, leur profil Iso-SMAF et le nombre de personnes en attente d'une place en SAD en résidence familiale.

Paramètres

- **applicants_none** (*dataframe*) – Nombre de personnes à domicile sans services par profil Iso-SMAF
- **applicants_svc** (*dataframe*) – Nombre de personnes à domicile avec du SAD par profil Iso-SMAF
- **waiting_users** (*object*) – Nombre de personnes en attente de SAD
- **region_id** (*int*) – numéro de région

- create_users()

simsad.home.home.**create_users**(*self*, *users_none*, *users_svc*)

Fonction qui crée le bassin d'individus à domicile.

Paramètres

- **applicants_none** (*dataframe*) – Nombre de personnes à domicile sans services par région, profil Iso-SMAF et groupe d'âge
- **applicants_svc** (*dataframe*) – Nombre de personnes à domicile avec du SAD par région, profil Iso-SMAF et groupe d'âge

- update_users()

simsad.home.home.**update_users**(*self*)

Fonction qui met à jour les caractéristiques du bassin d'individus à domicile.

4.2.2 RPA

La classe *rpa* permet de modéliser les personnes vivant en RPA qui reçoivent des services financés par le réseau public.

simsad.rpa.**rpa**(*policy*)

Résidence privée pour aînés (RPA)

Cette classe permet de modéliser les différents aspects des résidences privées pour aînés.

Paramètres

policy (*object*) – paramètres du scénario

Cette classe contient les fonctions suivantes :

- load_register()

simsad.rpa.rpa.**load_register**(*self*, *start_yr=2019*)

Fonction qui permet de charger les paramètres liés aux RPA et qui crée le registre de ceux-ci. Ce registre contient de l'information sur le nombre de personnes, leur profil Iso-SMAF et le nombre de personnes en attente d'une place en RPA subventionnée.

Paramètres

start_yr (*integer*) – année de référence (défaut=2019)

- assign()

simsad.rpa.rpa.**assign**(*self*, *applicants*, *waiting_users*, *region_id*)

Fonction qui comptabilise dans le registre des RPA les profils Iso-SMAF des usagers, le nombre de ceux-ci, et le nombre de personnes en attente d'une place.

Paramètres
— **applicants** (*dataframe*) – Nombre d'usagers par profil Iso-SMAF
— **waiting_users** (*dataframe*) – Nombre de personnes en attente d'une place
— **region_id** (*int*) – Numéro de la région

- build()

simsad.rpa.rpa.**build**(*self*)

Fonction qui permet le développement de places en RPA subventionnées par le public.

- create_users()

simsad.rpa.rpa.**create_users**(*self*, *users*)

Fonction qui crée le dataframe du bassin d'individus en RPA subventionnées.

Paramètres
users (*dataframe*) – Nombre d'usagers en RPA subventionnées par région, profil Iso-SMAF et groupe d'âge

- update_users()

simsad.rpa.rpa.**update_users**(*self*)

Fonction qui met à jours les caractéristiques des personnes dans le bassin d'individus en RPA subventionnée.

4.2.3 RI-RTF

La classe *ri* permet de modéliser les personnes vivant en RI-RTF, ainsi que les services fournis par les travailleurs de ce milieu de vie.

simsad.ri.**ri**(*policy*)

Ressources intermédiaires et ressources de type familial (RI-RTF)

Cette classe permet de modéliser les différents aspects des ressources intermédiaires et ressources de type familial.

Paramètres
policy (*object*) – paramètres du scénario

Cette classe contient les fonctions suivantes :

- load_register()

simsad.ri.ri.**load_register**(*self*, *start_yr=2019*)

Fonction qui permet de charger les paramètres liés aux RI-RTF et qui crée le registre de ceux-ci. Ce registre contient des informations sur le nombre de places, le nombre d'usagers, leur profil Iso-SMAF, les heures de services fournis, la main d'oeuvre et les coûts.

Paramètres
start_yr (*int*) – année de référence (défaut=2019)

- assign()

simsad.ri.ri.**assign**(*self*, *applicants*, *waiting_users*, *region_id*)

Fonction qui comptabilise dans le registre des RI-RTF les profils Iso-SMAF des usagers, le nombre de ceux-ci, et le nombre de personnes en attente d'une place.

Paramètres
- **applicants** (*dataframe*) – Nombre d'usagers par profil Iso-SMAF
- **waiting_users** (*dataframe*) – Nombre de personnes en attente d'une place
- **region_id** (*int*) – Numéro de la région

- build()

simsad.ri.ri.**build**(*self*)

Fonction qui permet de développer des places en RI-RTF.

- compute_supply()

simsad.ri.ri.**compute_supply**(*self*)

Fonction qui calcule le nombre total d'heures de services pouvant être fournies en RI-RTF (AVQ et AVD) selon la main d'oeuvre disponible.

- compute_costs()

simsad.ri.ri.**compute_costs**(*self*)

Fonction qui calcule les coûts des RI-RTF.

- create_users()

simsad.ri.ri.**create_users**(*self*, *users*)

Fonction qui crée le dataframe du bassin d'individus en RI-RTF.

Paramètres
users (*dataframe*) – Nombre d'usagers en RI-RTF par région, profil Iso-SMAF et groupe d'âge

- update_users()

simsad.ri.ri.**update_users**(*self*)

Fonction qui met à jours les caractéristiques des personnes dans le bassin d'individus en RI-RTF.

4.2.4 CHSLD

La classe *chsld* permet de modéliser les personnes vivant en CHSLD, ainsi que les services fournis par les travailleurs de ce milieu de vie.

simsad.chsld.**chsld**(*policy*)

Centres d'hébergement de soins de longue durée (CHSLD)

Cette classe permet de modéliser les différents aspects des Centres d'hébergement de soins de longue durée.

Paramètres
policy (*object*) – paramètres du scénario

Cette classe contient les fonctions suivantes :

- load_register()

simsad.chsld.chsld.**load_register**(*self*, *start_yr=2019*)

Fonction qui permet de charger les paramètres liés aux CHSLD et qui crée le registre de ceux-ci. Ce registre contient des informations sur le nombre de places, le nombre d'usagers, leur profil Iso-SMAF, les heures de services fournis, la main d'oeuvre et les coûts.

Paramètres
start_yr (*int*) – année de référence (défaut=2019)

- assign()

simsad.chsld.chsld.**assign**(*self*, *applicants*, *waiting_users*, *region_id*)

Fonction qui comptabilise dans le registre des CHSLD les profils Iso-SMAF des usagers, le nombre de ceux-ci, et le nombre de personnes en attente d'une place.

Paramètres
— **applicants** (*array*) – Nombre d'usagers par profil Iso-SMAF
— **waiting_users** (*array*) – Nombre de personnes en attente d'une place
— **region_id** (*int*) – Numéro de la région

- purchase()

simsad.chsld.chsld.**purchase**(*self*)

Fonction qui permet l'achat de places en CHSLD non-conventionné.

- build()

simsad.chsld.chsld.**build**(*self*)

Fonction qui permet la construction de places en CHSLD public.

- compute_supply()

simsad.chsld.chsld.**compute_supply**(*self*)

Fonction qui calcule le nombre total d'heures de services pouvant être fournies en CHSLD (soins infirmiers et AVQ) selon la main d'oeuvre disponible. On suppose qu'il n'y a pas de contrainte de main d'oeuvre pour le soutient au AVD en CHSLD.

- compute_costs()

simsad.chsld.chsld.**compute_costs**(*self*)

Fonction qui calcule les coûts des CHSLD.

- create_users()

simsad.chsld.chsld.**create_users**(*self*, *users*)

Fonction qui crée le dataframe du bassin d'individus en CHSLD.

Paramètres
users (*dataframe*) – Nombre d'individus en CHSLD par région, profil Iso-SMAF et groupe d'âge

- update_users()

simsad.chsld.chsld.**update_users**(*self*)

Fonction qui met à jours les caractéristiques des personnes dans le bassin d'individus en CHSLD.

4.2.5 NSA

La classe *nsa* permet de modéliser les personnes en centre hospitalier en niveau de soins alternatifs (NSA), ainsi que les services fournis par les travailleurs de ce milieu de vie.

simsad.nsa.**nsa**(*policy*)

Personnes dans un centre hospitalier en niveau de soins alternatifs (NSA)

Cette classe permet de modéliser les personnes dans un centre hospitalier en niveau de soins alternatifs.

Paramètres
policy (*object*) – paramètres du scénario

Cette classe contient les fonctions suivantes :

- load_register()

simsad.nsa.nsa.**load_register**(*self*, *start_yr=2019*)

Fonction qui permet de créer le registre des NSA. Ce registre contient des informations sur le nombre de places en centre hospitalier, le nombre d'usagers et leur profil Iso-SMAF.

Paramètres
start_yr (*integer*) – année de référence (défaut=2019)

- assign()

simsad.nsa.nsa.**assign**(*self*, *applicants*, *region_id*)

Fonction qui comptabilise dans le registre des NSA le nombre de personnes et leur profil Iso-SMAF.

Paramètres
— **applicants** (*dataframe*) – nombre de personnes en NSA par profil Iso-SMAF
— **region_id** (*int*) – numéro de la région

- create_users()

simsad.nsa.nsa.**create_users**(*self*, *users*)

Fonction qui crée le bassin d'individus en NSA.

Paramètres
users (*dataframe*) – Nombre de personnes en NSA par région, profil Iso-SMAF et groupe d'âge

- compute_costs()

simsad.nsa.nsa.**compute_costs**(*self*)

Fonction qui calcule les coûts des NSA.

- update_users()

simsad.nsa.nsa.**update_users**(*self*)

Fonction qui met à jour les caractéristiques du bassin d'individus en NSA.

4.3 Fournisseurs

4.3.1 CLSC

La classe *clsc* permet de modéliser les services fournis par les CLSC.

simsad.clsc.**clsc**(*policy*)

Centres locaux de services communautaires (CLSC)

Cette classe permet de modéliser les services offerts en soutien à l'autonomie par les Centres locaux de services communautaires.

Paramètres
policy (*object*) – paramètres du scénario

Cette classe contient les fonctions suivantes :

- load_registry()

simsad.clsc.clsc.**load_registry**(*self*, *start_yr=2019*)

Fonction qui permet de créer le registre des CSLC. Ce registre contient des informations concernant la main d'oeuvre, les heures de services fournis et les coûts.

Paramètres
start_yr (*int*) – année de référence (défaut=2019)

- load_params()

simsad.clsc.clsc.**load_params**(*self*)

Fonction qui permet de charger les paramètres liés aux heures de services fournis par les CSLC.

- assign()

simsad.clsc.clsc.**assign**(*self*, *users*, *milieu*, *policy*, *yr*)

Fonction qui détermine les usagers des CLSC et qui leur attribue des heures de services fournis (soins infirmier, AVQ, AVD).

Paramètres
— **users** (*dataframe*) – Bassin d'individus d'un milieu de vie donné
— **milieu** (*str*) – Nom du milieu de vie
— **policy** (*object*) – Paramètres du scénario
— **yr** (*int*) – Année en cours de la simulation

Renvoie
users – Bassin d'individus d'un milieu de vie donné

Type renvoyé
dataframe

- compute_supply()

simsad.clsc.clsc.**compute_supply**(*self*, *yr*)

Fonction qui comptabilise les heures de services pouvant être fournies en CLSC (soins infirmiers et AVQ) selon la main d'oeuvre disponible.

Paramètres
yr (*int*) – Année en cours de la simulation

- cap()

simsad.clsc.clsc.**cap**(*self*, *users*, *milieu*)

Fonction qui ajuste les heures de services fournis au niveau individuel (soins infirmier, AVQ, AVD) selon la main d'oeuvre disponible et qui comptabilise les besoins supplémentaires en main d'oeuvre.

Paramètres
— **users** (*dataframe*) – Bassin d'individus d'un milieu de vie donné
— **milieu** (*str*) – Nom du milieu de vie

Renvoie
users – Bassin d'individus d'un milieu de vie donné

Type renvoyé
dataframe

- compute_costs()

simsad.clsc.clsc.**compute_costs**(*self*)

Fonction qui calcule les coûts des CLSC.

- workforce()

simsad.clsc.clsc.**workforce**(*self*, *before_base_yr=False*)

Fonction qui ajuste le nombre d'ETC en CLSC selon les besoins supplémentaires en main d'oeuvre et le taux d'ajustement de celle-ci.

Paramètres
before_base_yr (*boolean*) – True si l'année en cours de la simulation est inférieure à l'année de départ de la comtabilisation des résultats

4.3.2 EÉSAD

La classe *eesad* permet de modéliser les services fournis par les EÉSAD.

simsad.eesad.**eesad**(*policy*)

Entreprises d'économie sociale en aide à domicile (EÉSAD)

Cette classe permet de modéliser les services offerts en soutien à l'autonomie par les entreprises d'économie sociale en aide à domicile.

Paramètres
policy (*object*) – paramètres du scénario

Cette classe contient les fonctions suivantes :

- load_registry()

simsad.eesad.eesad.**load_registry**(*self*, *start_yr=2019*)

Fonction qui permet de créer le registre des EÉSAD. Ce registre contient des informations concernant la main d'oeuvre et les coûts.

Paramètres
start_yr (*integer*) – année de référence (défaut=2019)

- assign()

simsad.eesad.eesad.**assign**(*self*, *users_home*, *users_rpa*)

Fonction qui calcule la contribution des usagers des EÉSAD dans les bassins de d'individus à domicile ou en RPA, et qui comptabilise le nombre d'usagers et le nombre total d'heures de services fournis (en AVD) par région et profil Iso-SMAF.

Paramètres
- **users_home** (*dataframe*) – Bassin d'individus à domicile
- **users_rpa** (*dataframe*) – Bassin d'individus en RPA

Renvoie
- **users_home** (*dataframe*) – Bassin d'individus à domicile
- **users_home** (*dataframe*) – Bassin d'individus en RPA

- compute_supply()

simsad.eesad.eesad.**compute_supply**(*self*)

Fonction qui calcule le nombre total d'heures de services pouvant être fournies en EÉSAD (AVD) selon la main d'oeuvre disponible.

- cap()

simsad.eesad.eesad.**cap**(*self*, *users_home*, *users_rpa*)

Fonction qui ajuste les heures de services fournis au niveau individuel (AVD) selon la main d'oeuvre disponible et qui comptabilise les besoins supplémentaires en main d'oeuvre pour les EÉSAD.

Paramètres
- **users_home** (*dataframe*) – Bassin d'individus à domicile
- **users_home** – Bassin d'individus en RPA

Renvoie
- **users_home** (*dataframe*) – Bassin d'individus à domicile
- **users_home** (*dataframe*) – Bassin d'individus en RPA

- compute_costs()

simsad.eesad.eesad.**compute_costs**(*self*)

Fonction qui calcule les coûts des EÉSAD.

- workforce()

simsad.eesad.eesad.**workforce**(*self*, *before_base_yr=False*)

Fonction qui ajuste le nombre d'ETC en EÉSAD selon les besoins supplémentaires en main d'oeuvre et le taux d'ajustement de celle-ci.

Paramètres
before_base_yr (*boolean*) – True si l'année en cours de la simulation est inférieure à l'année de départ de la comptabilisation des résultats

4.3.3 Privé

La classe *prive* permet de modéliser les services fournis par les EÉSAD.

simsad.prive.**prive**(*policy*)

Secteur privé

Cette classe permet de modéliser les services offerts par le secteur privé et financés par le public.

Paramètres
policy (*object*) – paramètres du scénario

Cette classe contient les fonctions suivantes :

- load_registry()

simsad.prive.prive.**load_registry**(*self*, *start_yr=2021*)

Fonction qui permet de créer le registre du secteur privé. Ce registre contient des informations concernant la main d'oeuvre, les heures fournies et les coûts.

Paramètres
start_yr (*integer*) – année de référence (défaut=2021)

- assign()

simsad.prive.prive.**assign**(*self*, *users*)

Fonction qui comptabilise le nombre d'usagers et le nombre total d'heures de services fournis (en AVQ et AVD) par région et profil Iso-SMAF.

Paramètres
users (*dataframe*) – Bassin d'individus d'un milieu de vie donné

Renvoie
users – Bassin d'individus d'un milieu de vie donné

Type renvoyé
dataframe

- compute_supply()

simsad.prive.prive.**compute_supply**(*self*)

Fonction qui calcule le nombre total d'heures de services pouvant être fournies par le privé (AVQ et AVD) selon la main d'oeuvre disponible.

- cap()

simsad.prive.prive.**cap**(*self*, *users*)

Fonction qui ajuste les heures de services fournis au niveau individuel (AVQ et AVD) selon la main d'oeuvre disponible et qui comptabilise les besoins supplémentaires en main d'oeuvre pour le privé.

Paramètres
— **users** (*dataframe*) – Bassin d'individus d'un milieu de vie donné
— **milieu** (*str*) – Nom du milieu de vie

Renvoie
users – Bassin d'individus d'un milieu de vie donné

Type renvoyé
dataframe

- compute_costs()

simsad.prive.prive.**compute_costs**(*self*)

Fonction qui calcule les coûts du privé.

- workforce()

simsad.prive.prive.**workforce**(*self*, *before_base_yr=False*)

Fonction qui ajuste le nombre d'ETC du privé selon les besoins supplémentaires en main d'oeuvre et le taux d'ajustement de celle-ci.

Paramètres
before_base_yr (*boolean*) – True si l'année en cours de la simulation est inférieure à l'année de départ de la comptabilisation des résultats

4.4 Programmes

4.4.1 Chèque emploi-service (CES)

La classe *ces* permet de modéliser le Chèque emploi-service.

simsad.ces.**ces**()

Chèque emploi-service (CES)
Cette classe permet d'attribuer les services reçus dans le cadre du Chèque emploi-service.

Cette classe contient les fonctions suivantes :

- load_params()

simsad.ces.ces.**load_params**(*self*, *start_yr=2021*)

Fonction qui permet de charger les paramètres liés au CES.

Paramètres
start_yr (*int*) – année de référence (défaut=2021)

- assign()

simsad.ces.ces.**assign**(*self*, *users*)

Fonction qui attribue aux individus les heures de services reçues dans le cadre du CES.

Paramètres
users (*dataframe*) – bassin d'individus d'un milieu de vie donné

Renvoie
users – bassin d'individus pour un milieu de vie donné

Type renvoyé
dataframe

- calibrate()

simsad.ces.ces.**calibrate**(*self*, *users*, *targets_by_region*)

Fonction qui calibre les heures de services reçues dans le cadre du CES par rapport aux données observées.

Paramètres
— **users** (*dataframe*) – bassin d'individus d'un milieux de vie donné
— **targets_by_region** (*dataframe*) – valeurs cibles par région

4.4.2 Crédit d'impôt pour maintien à domicile des aînés (CMD)

La classe *cmd* permet de modéliser le crédit d'impôt pour maintien à domicile des aînés.

simsad.cmd.**cmd**()

Crédit d'impôt pour maintien à domicile (CMD) des aînés

Cette classe permet de modéliser les coûts liés au Crédit d'impôt pour maintien à domicile des aînés.

Cette classe contient les fonctions suivantes :

- load_registry()

simsad.cmd.cmd.**load_registry**(*self*)

Fonction qui permet de créer le registre du CMD. Ce registre contient des informations sur les montants agrégés.

- load_params()

simsad.cmd.cmd.**load_params**(*self*, *start_yr=2019*)

Fonction qui permet de charger les paramètres liés aux montants de CMD par individu.

Paramètres

start_yr (*int*) – année de référence (défaut=2019)

- assign()

simsad.cmd.cmd.**assign**(*self*, *users*, *milieu*)

Fonction qui détermine les prestataires du CMD et qui attribue les montants reçu.

Paramètres

— **users** (*dataframe*) – Bassin d'individus d'un milieu de vie donné
— **milieu** (*str*) – Nom du milieu de vie

Renvoie

users – Bassin d'individus d'un milieu de vie donné

Type renvoyé

dataframe

- calibrate()

simsad.cmd.cmd.**calibrate**(*self*, *users_home*, *users_rpa*, *yr*)

Fonction qui calibre les montants simulés de CMD sur les montants agrégés budgétés jusqu'en 2026.

Paramètres

— **users_home** (*dataframe*) – Bassin d'individus à domicile
— **users_home** – Bassin d'individus en RPA
— **yr** (*int*) – Année en cours de la simulation

Renvoie

— **users_home** (*dataframe*) – Bassin d'individus à domicile
— **users_home** (*dataframe*) – Bassin d'individus en RPA

- compute_costs()

simsad.cmd.cmd.**compute_costs**(*self*, *users_home*, *users_rpa*)

Fonction qui calcule les coûts du CMD.

Paramètres

— **users_home** (*dataframe*) – Bassin d'individus à domicile
— **users_home** – Bassin d'individus en RPA

4.4.3 Programme d'exonération financière pour les services d'aide domestique (PEFSAD)

La classe *pefsad* permet de modéliser le Programme d'exonération financière pour les services d'aide domestique.

simsad.pefsad.**pefsad**()

Programme d'exonération financière pour les services d'aide domestique (PEFSAD)

Cette classe permet de modéliser les services offerts en soutien à l'autonomie par les Entreprises d'économie sociale en aide à domicile.

Cette classe contient les fonctions suivantes :

- load_params()

simsad.pefsad.pefsad.**load_params**(*self*)

Fonction qui permet de charger les paramètres liés aux heures de services fournis dans le cadre du PEFSAD.

- assign()

simsad.pefsad.pefsad.**assign**(*self*, *users*, *milieu*)

Fonction qui détermine les usagers du PEFSAD et qui leur attribue des heures de services fournis (AVD).

Paramètres

— **users** (*dataframe*) – Bassin d'individus d'un milieu de vie donné

— **milieu** (*string*) – Nom du milieu de vie

Renvoie

users – Bassin d'individus d'un milieu de vie donné

Type renvoyé

dataframe

4.5 Préférences

La classe *prefs* permet de modéliser les préférences des individus par rapport à la prise en charge de la perte d'autonomie.

simsad.prefs.**prefs**()

Préférences pour la prise en charge de la perte d'autonomie

Cette classe permet de modéliser les préférences par rapport à la prise en charge de la perte d'autonomie.

Cette classe contient les fonctions suivantes :

- utility()

simsad.prefs.prefs.**utility**(*self*, *row*)

Fonction qui calcule l'utilité pour un individu.

Paramètres

row (*dataframe*) – arguments (variables) de la fonction d'utilité

Renvoie

u – utilité de l'individu

Type renvoyé

float

- compute_utility()

simsad.prefs.prefs.**compute_utility**(*self*, *users*)

Fonction qui calcule l'utilité pour l'ensemble d'un groupe en unité et en valeur monétaire.

Paramètres
users (*dataframe*) – Bassin d'individus d'un milieu de vie donné

Renvoie
users – Bassin d'individus d'un milieu de vie donné

Type renvoyé
dataframe

4.6 Outils

La classe *tracker* permet la création de tableaux de sortie par rapport aux différents résultats du modèle.

simsad.tracker.**tracker**(*scn_name='reference'*)

Résultats de simulation

Cette classe permet la création de tableaux de sortie par rapport aux différents résultats du modèle.

Paramètres
scn_name (*str*) – nom du scénario (défaut="reference")

Cette classe contient les fonctions suivantes :

- add_entry()

simsad.tracker.tracker.**add_entry**(*self*, *entry_name*, *class_member*, *domain*, *rowvars*, *colvars*, *aggfunc*, *start_yr*, *stop_yr*)

Fonction qui permet la création d'un nouveau tableau de sortie.

Paramètres
— **entry_name** (*str*) – nom du tableau
— **class_member** (*str*) – nom de la classe d'où proviennent les résultats
— **domain** (*str*) – nom du domaine d'où proviennent les résultats
— **rowvars** (*str*) – variables définissant les groupes d'agrégation (en ligne)
— **colvars** (*str*) – variables de résultats (en colonne)
— **aggfunc** (*str*) – nom de la fonction d'agrégation (ex. sum, mean, etc.)
— **start_yr** (*int*) – année de départ
— **stop_yr** (*int*) – année de fin

- log()

simsad.tracker.tracker.**log**(*self*, *p*, *yr*)

Fonction qui procède à la comptabilisation des résultats dans les tableaux de sortie.

Paramètres
— **p** (*object*) – instance de classe
— **yr** (*int*) – année en cours dans la simulation

- save()

simsad.tracker.tracker.**save**(*self*, *dir*, *scn_policy*)

Fonction qui procède à la sauvegarde des tableaux de sortie dans un fichier excel.

Paramètres
— **dir** (*string*) – sentier où sauvegarder les résultats

— **scn_policy** (*object*) – instance de classe policy

CHAPITRE 5

Nous joindre

5.1 Principaux contributeurs

Nicholas-James Clavet, Réjean Hébert, Pierre-Carl Michaud, Julien Navaux et Michel Raîche.

5.2 Personne-contact

Julien Navaux

5.3 Liste d'envoi

Pour rester informé.e des mises à jour de SimSAD, inscrivez-vous à notre liste d'envoi dédiée.

5.4 Problèmes et améliorations

SimSAD est un modèle dit « open source ». Les utilisateurs sont donc invités à signaler les problèmes liés à SimSAD et à soumettre des propositions d'ajout au code en cliquant sur le lien suivant, et en cliquant sur le bouton vert « New issue ».

CHAPITRE 6

Droits d'utilisation

SimSAD est fourni sous licence MIT (« MIT License »). Les conditions de la licence sont les suivantes :

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the « Software »), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions :

The copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED « AS IS », WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

CHAPITRE 7

Index

— genindex

CHAPITRE 8

Documentation en PDF

— Documentation technique `pdf`
— Documentation sur la programmation `pdf`

Index

W