
CPR
Version 1.2.2

RSI Team / Équipe IRE

mai 31, 2021

Table des matières

1 Premiers pas avec le CPR	3
2 Main	5
3 Simulator	7
4 Initialisation	11
5 Macro	15
6 Assets	17
7 Debts	23
8 Taxes	25
9 Annuities	27
10 Life	29
11 Balance Sheets	31
12 Analysis	33
13 Outils	35
14 Tutoriels	37
15 Contributeurs et droits d'utilisation	39
16 Documentation en PDF	41
Index des modules Python	43
Index	45

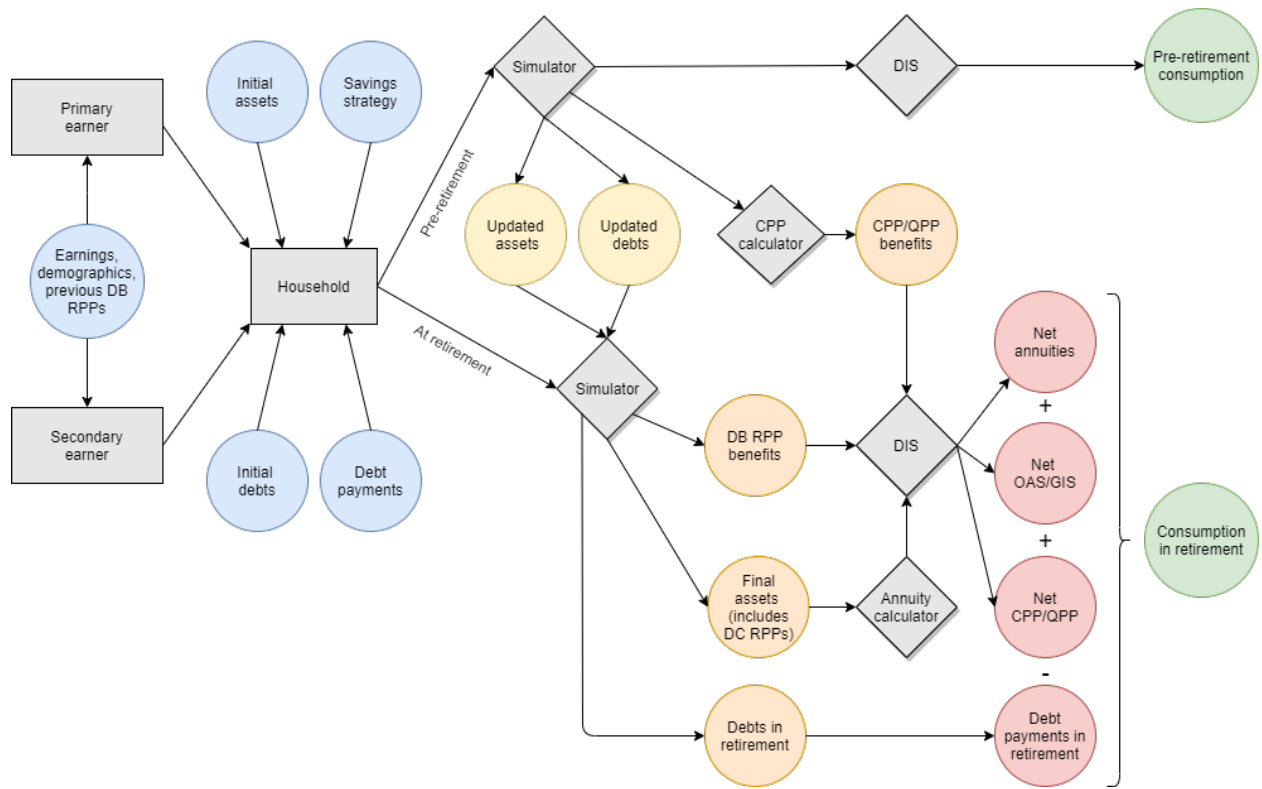
L'outil de Préparation à la retraite des Canadiens (Canadians' Preparation for Retirement, ou CPR) a été développé par une équipe à l'Institut sur la retraite et l'épargne (IRE) de HEC Montréal, avec du soutien financier du National Pension Hub de l'Institut du risque mondial. Il permet de calculer la préparation pour la retraite de tout ménage ou groupe de ménages pour le(s)quel(s) l'utilisateur dispose des intrants requis (informations personnelles et financières) à tout moment donné dans le temps. Il a été utilisé avec des données confidentielles de 2018 pour préparer un rapport de recherche.

Tel que mentionné ailleurs, le CPR est fourni « tel quel », sous une *licence MIT*.

Le CPR simule les années entre l'année de base et la retraite, puis convertit tout le patrimoine financier en rentes actuariellement équitables. Il compare ensuite les situations avant et après la retraite. Comparé aux autres outils disponibles, le CPR offre une flexibilité et une transparence complètes puisque son code est entièrement disponible en ligne sans frais, et qu'il peut être modifié et adapté à volonté. De manière importante et distinctive, le CPR permet également de réaliser des simulations stochastiques, qui permettent à plusieurs dimensions de varier dans le temps selon des hypothèses et des processus de pointe. Tout cela peut être modifié par l'utilisateur sophistiqué, tout comme le nombre de répliquions que l'outil utilise pour calculer des résultats moyens et déterminer la (probabilité de) préparation pour la retraite. La version actuelle du CPR utilise 2018 comme année de base; les systèmes fiscaux du Québec et de l'Ontario sont présentement modélisés (on peut approximer les autres provinces en utilisant l'une des ces 2 provinces, de préférence l'Ontario).

Le CPR produit comme extrant un niveau de remplacement de la consommation » pour chaque ménage passant dans le simulateur, et il classe chaque ménage comme étant « préparé » ou « non préparé » pour la retraite sur la base du critère fixé au préalable. Comme dans les rapports produits par d'autres groupes au cours des années 2010, la préparation pour la retraite peut être évaluée en calculant le ratio entre la consommation après la retraite et la consommation avant la retraite et en comparant ce ratio à un seuil, fixé à 65% pour les 4 quintiles de revenu supérieurs et à 80% pour le quintile inférieur. Ces mesures peuvent ensuite être agrégées sur l'ensemble des ménages analysés afin d'obtenir a) la moyenne du taux de remplacement, ou sa distribution; et b) la proportion de ménages qui sont classés « préparés ». Lorsque la version stochastique de l'outil est utilisée, des probabilités d'être préparé peuvent également être calculées et visualisées, pour un ou plusieurs ménage(s). La figure ci-dessous illustre de manière conceptuelle les différentes parties et les flux du CPR.

Le CPR et les autres outils qu'il utilise (le SRD et le SRPP) sont écrits en Python, un langage simple, rapide et moderne. Afin de pouvoir l'utiliser, il faut s'assurer d'avoir au préalable installé une distribution à jour de Python, par exemple à l'aide d'Anaconda. Bien que ce ne soit pas essentiel, il sera également utile de se familiariser un minimum, au préalable, avec les environnements et, si possible, le vocabulaire Python (p.ex. fonction, classe, instance, profil).



Premiers pas avec le CPR

Le CPR et les autres outils qu'il utilise (le SRD et le SRPP) sont écrits en Python, un langage simple, rapide et moderne. Afin de pouvoir l'utiliser, il faut s'assurer d'avoir au préalable installé une distribution à jour de Python, par exemple à l'aide d'Anaconda. Dans tous les cas, les exigences minimales pour utiliser le CPR sont Python 3.6+ avec *numpy*, *pandas* et *xlrd*. Bien que ce ne soit pas essentiel, il sera également utile de se familiariser un minimum, au préalable, avec les environnements et, si possible, le vocabulaire Python (p.ex. fonction, classe, instance, profil).

1.1 Installation du CPR

On peut installer facilement le CPR en utilisant pip :

```
pip install cpr-rsi
```

Veuillez lire les [conditions d'utilisation \(en anglais\)](#) de *pypi*, le site Web qui héberge le package.

Le CPR utilise deux autres packages produits par l'IRE et la Chaire de recherche sur les enjeux économiques inter-générationnels (CRREi), soutenue par l'IRE : le [Simulateur de revenu disponible \(SRD\)](#), qui calcule les impôts et les principales prestations ; et le [Simulateur de régimes de pensions publiques](#), qui simule les cotisations et les prestations de RRQ et de RPC. Tant le SRD que le SRPP sont installés automatiquement en même temps que le CPR.

Dans un notebook ou un projet, on invoque le CPR en ajoutant la commande suivante :

```
import CPR
```

Pour désinstaller le CPR, le SRD et le SRPP, il suffit d'utiliser pip :

```
pip uninstall cpr-rsi srd srpp
```

1.2 Installation alternative

Si l'on est dans l'impossibilité d'installer le CPR, un fichier zip contenant le CPR et ses dépendances, le SRD et le SRPP, peut être téléchargé de [Github](#) (choisir le fichier CPR_with_dep.zip dans le plus récent « release »). Pour utiliser le CPR, il suffit d'extraire les fichiers du zip dans le répertoire de son choix et de travailler de ce dernier ou d'ajouter ce répertoire au chemin d'accès (par exemple en utilisant le module `sys`). Un notebook jupyter qui explique le fonctionnement du CPR, *Tutorial CPR*, est également inclus dans le fichier zip. Si l'on ne veut plus le CPR, il suffit d'effacer les répertoires `cpr`, `srd` et `srpp`.

Pour plus de détails concernant la manière d'utiliser le CPR, veuillez consulter les vidéos (en anglais pour le moment) disponibles à la section *tutoriels*.

Tel que mentionné ailleurs, le CPR est fourni « tel quel », sous une *licence MIT*.

En cas de questions, commentaires ou suggestions, n'hésitez pas à *nous contacter*.

Ce module appelle le simulateur.

La fonction `run_simulations` fait tourner les simulations. Ses arguments sont utilisés pour choisir si la version stochastique de l'outil doit être utilisée, ainsi que le nombre de simulations (réplications) qui doivent être effectuées. N'importe quelle valeur de paramètre peut également être modifiée ici, afin d'utiliser des valeurs autres que celles fournies par défaut.

`CPR.main.run_simulations(inputs, nsim=1, non_stochastic=False, n_jobs=None, **extra_params)`

Cette fonction lance les simulations. N'importe quel paramètre peut être modifié à l'aide de `extra_params`.

Paramètres

- **inputs** (`_io.TextIOWrapper`) – fichier csv
- **nsim** (`int`, *optional*) – nombre de simulations, par défaut 1
- **non_stochastic** (`bool`, *optional*) – prix et ratio prix-loyers déterministes, par défaut à Faux

Renvoie instance de la classe `Results`

Type renvoyé `Results`

Ce module fait tourner les simulations.

`CPR.simulator.simulate(job, common, prices)`

Fonction projetant les actifs, les RPA et les dettes jusqu'au moment de la retraite.

Paramètres

- **job** (*tuple*(`Hhold`, *int*)) – instance de la classe `Hhold` et numéro de simulation
- **common** (`Common`) – instance de la classe `Common`
- **prices** (`Prices`) – instance de la classe `Prices`

Renvoie dictionnaire contenant les caractéristiques des ménages avant et après la retraite

Type renvoyé dict

`CPR.simulator.extract_time_series(sim, common, prices)`

Fonction qui rattache aux ménages des processus stochastiques en lien avec les rendements sur les actifs.

Paramètres

- **sim** (*int*) – numéro de simulation
- **common** (`Common`) – instance de la classe `Common`
- **prices** (`Prices`) – instance de la classe `Prices`

Renvoie

- *dict* – rendements sur les actifs
- *dict* – ratio prix-loyers

`CPR.simulator.prepare_wages(p, sim, common, prices)`

Fonction qui rattache des profils de revenus de travail aux individus.

Paramètres

- **p** (`Person`) – instance de la classe `Person`
- **sim** (*int*) – numéro de simulation
- **common** (`Common`) – instance de la classe `Common`
- **prices** (`Prices`) – instance de la classe `Prices`

`CPR.simulator.initialize_cpp_account(p, hh, common)`

Fonction qui crée un compte RPC/RRQ et entre les cotisations passées au RPC/RRQ sur la base des revenus de travail passés.

Paramètres

- **p** (*Person*) – instance de la classe *Person*
- **hh** (*Hhold*) – household
- **common** (*Common*) – instance de la classe *Common*

`CPR.simulator.update_ages(hh, year)`

Fonction qui calcule l'âge pour une année donnée.

Paramètres

- **hh** (*Hhold*) – household
- **year** (*int*) – année

`CPR.simulator.update_debts(hh, year, sim, common, prices)`

Fonction qui met à jour les paiements sur les dettes ainsi que les soldes.

Paramètres

- **hh** (*Hhold*) – household
- **year** (*int*) – année
- **sim** (*int*) – numéro de simulation
- **common** (*Common*) – instance de la classe *Common*
- **prices** (*Prices*) – instance de la classe *Prices*

`CPR.simulator.adjust_contributions(hh, year, common, prices)`

Fonction qui ajuste les cotisations à tous les types de comptes (REER, autres comptes enregistrés, CELI) et aux régimes de retraite (RPA à CD et à PD) de manière à respecter les espaces de cotisation disponibles.

Paramètres

- **hh** (*Hhold*) – household
- **year** (*int*) – année
- **common** (*Common*) – instance de la classe *Common*
- **prices** (*Prices*) – instance de la classe *Prices*

`CPR.simulator.update_assets(hh, year, d_returns, common, prices)`

Fonction qui met à jour les actifs chaque année.

Paramètres

- **hh** (*Hhold*) – household
- **year** (*int*) – année
- **d_returns** (*dict*) – dictionnaire de rendements
- **common** (*Common*) – instance de la classe *Common*
- **prices** (*Prices*) – instance de la classe *Prices*

`CPR.simulator.manage_liquidations(hh, year, common, prices)`

Fonction gérant la liquidation des actifs au moment de la retraite.

Paramètres

- **hh** (*Hhold*) – household
- **year** (*int*) – année
- **common** (*Common*) – instance de la classe *Common*
- **prices** (*Prices*) – instance de la classe *Prices*

`CPR.simulator.contribute_cpp(p, year, common)`

Fonction qui enregistre les cotisations au RPC/RRQ.

Paramètres

- **p** (*Person*) – instance de la classe *Person*
- **year** (*int*) – année
- **common** (*Common*) – instance de la classe *Common*

`CPR.simulator.claim_cpp(p)`

Fonction pour faire débiter les prestations de RPC/RRQ.

Paramètres **p** (*Person*) – instance de la classe *Person*

`CPR.simulator.check_tax_unreg(hh)`

Fonction qui vérifie s'il y a des rendements imposables sur les actifs.

Paramètres `hh` (`Hhold`) – household

Renvoie Vrai ou Faux

Type renvoyé bool

`CPR.simulator.check_liquidation(hh)`

Fonction qui vérifie s'il y a des actifs liquidés à imposer.

Paramètres `hh` (`Hhold`) – household

Renvoie Vrai ou Faux

Type renvoyé bool

`CPR.simulator.prepare_taxes(hh, year, common, prices)`

Fonction qui prépare les variables utilisées dans le SRD (en termes nominaux)

Paramètres

- `hh` (`Hhold`) – household
- `year` (`int`) – année
- `common` (`Common`) – instance de la classe Common
- `prices` (`Prices`) – instance de la classe Prices

`CPR.simulator.get_assets(p, common)`

Fonction pour récupérer les actifs aux fins du test d'actifs de l'aide sociale.

Paramètres

- `p` (`Person`) – instance de la classe Person
- `common` (`Common`) – instance de la classe Common

`CPR.simulator.compute_rpp(p, nom, common)`

Calcul des RPA (PD et pension).

Paramètres

- `p` (`Person`) – instance de la classe Person
- `nom` (`function`) – Fonction convertissant en valeur nominale
- `common` (`Common`) – instance de la classe Common

`CPR.simulator.get_benefits_cpp(p, year, common)`

Fonction qui calcule les prestations de retraite annuelles de RPC/RRQ.

`s1` est la portion des prestations supplémentaires liée à un taux de cotisation supérieur dans le cadre de l'expansion 2019-2025; `s2` est la portion des prestations supplémentaires liée aux changements dans le MGAP; et `PRB` est le supplément à la rente de retraite (pour les individus qui continuent à travailler après le début de leur rente – cette aspect n'est pas utilisé dans le CPR, puisque les individus débutent automatiquement leurs prestations dans l'année de leur retraite).

Paramètres

- `p` (`Person`) – instance de la classe Person
- `year` (`int`) – année
- `common` (`Common`) – instance de la classe Common

`CPR.simulator.get_inc_rrsp(p, nom)`

Fonction qui calcule le revenu de retrait de REER.

Paramètres

- `p` (`Person`) – instance de la classe Person
- `nom` (`function`) – Fonction convertissant en valeur nominale

`CPR.simulator.get_other_taxable(p, nom, common)`

Fonction calculant les autres revenus imposables.

Paramètres

- **p** (*Person*) – instance de la classe *Person*
- **nom** (*function*) – Fonction convertissant en valeur nominale
- **common** (*Common*) – instance de la classe *Common*

CPR.simulator.**get_other_non_taxable**(*p, nom*)

Fonction calculant les autres revenus non imposables.

Paramètres

- **p** (*Person*) – instance de la classe *Person*
- **nom** (*function*) – Fonction convertissant en valeur nominale

CPR.simulator.**get_contributions_assets**(*p, year, common*)

Fonction calculant les cotisations aux comptes enregistrés et non enregistrés (incluant les RPA à CD et à PD).

Paramètres

- **p** (*Person*) – instance de la classe *Person*
- **year** (*int*) – année
- **common** (*Common*) – instance de la classe *Common*

CPR.simulator.**reset_accounts**(*hh*)

Fonction qui rétablit les valeurs initiales de toutes les variables.

Paramètres **hh** (*Hhold*) – household

CHAPITRE 4

Initialisation

Ce module configure des ménages et leur attache des actifs et des dettes. Le tableau ci-dessous montre tous les intrants individuels au CPR (certains sont entrés dans d'autres modules).

Earner Characteristics	Initial Assets	Initial Debts	Savings Strategy	Debt Payments
Birth year	Initial balance: RRSPs	First mortgage	RRSP contribution rate	First mortgage payment
Sex	Initial balance: TFSAs	Second mortgage	RRSP withdrawals	Second mortgage payment
Education level	Initial balance: other registered assets	Credit card	TFSA contribution rate	Credit card payment
Province of residence	Initial balance: unregistered assets	Personal loan	TFSA withdrawals	Personal loan payment
Initial wage	Initial balance: DC pensions	Student loan	Contribution rate to other registered assets	Student loan payment
Household type (couple vs. single)	Purchase price of principal residence	Car loan	Withdrawals of other registered assets	Car loan payment
DB pensions from a previous employer	Market value of principal residence	Credit line	Contribution rate to unregistered assets	Credit line payment
Intended CPP/QPP claiming age	Purchase price of second residence	Other debt	Withdrawals of unregistered assets	Other debt payment
Intended retirement age	Market value of second residence		Expected replacement rate of current DB plan	
	Purchase price of business owned		Employee contribution rate to current DB plan	
	Business equity		Employee contribution rate to DC plans	
			Employer contribution rate to DC plans	
			Share of bills in portfolio	
			Share of bonds in portfolio	
			Share of equity in portfolio	
			Fees paid on investments	
			Net unrealized capital gains on unregistered assets	
			Carried-forward past losses on unregistered assets	
			Initial contribution room: RRSP	
			Initial contribution room: TFSA	

Bien qu'un fichier exemple d'intrants soit fourni avec le package, un tutoriel est également offert afin de montrer aux utilisateurs comment modifier ces intrants afin d'utiliser ceux de leur choix. NOTE : il faut être prudent lorsque l'on utilise ses propres intrants, car un message d'erreur général sera généré si l'un des – nombreux – intrants est spécifié

de manière incorrecte (c.-à-d. qu'il contient une erreur ou est très peu plausible).

`CPR.initialisation.create_hh(index, d_hh, common, prices)`

Fonction qui crée un ménage avec une ou deux personnes et y attache des actifs et des dettes.

Paramètres

- **index** (*int*) – indice du ménage
- **d_hh** (*dict*) – dictionnaire de caractéristiques des ménages
- **common** (*Common*) – instance de la classe Common
- **prices** (*Prices*) – instance de la classe Prices

Renvoie Instance de la classe Hhold.

Type renvoyé *Hhold*

`class CPR.initialisation.Person(d_hh, l_sp, common, prices, s_=False)`

Cette classe crée une personne.

Paramètres

- **d_hh** (*dict*) – dictionnaire contenant toutes les informations à propos des ménages
- **l_sp** (*list*) – liste des caractéristiques attachées aux conjoints
- **common** (*Common*) – instance de la classe Common
- **prices** (*Prices*) – instance de la classe Prices
- **s** (*bool*) – Faux pour le chef de ménage, Vrai sinon

`create_wage_profile(common, prices)`

Fonction qui crée des profils de salaires pour chaque réalisation de l'incertitude, utilisée lorsque la version stochastique de l'outil est sélectionnée.

Paramètres

- **common** (*Common*) – instance de la classe Common
- **prices** (*Prices*) – instance de la classe Prices

Renvoie Profils de salaires.

Type renvoyé *np.array*

`create_shocks(T, N, prices)`

Fonction qui crée une série temporelle de chocs sur le salaire.

Paramètres

- **T** (*int*) – longueur de la série temporelle
- **N** (*int*) – nombre de réalisations
- **prices** (*Prices*) – instance de la classe Prices

Renvoie Série temporelle de chocs sur le salaire.

Type renvoyé *np.array*

`class CPR.initialisation.Hhold(d_hh, l_hhold, index, common, p0, p1=None)`

Cette classe crée un ménage.

Paramètres

- **d_hh** (*dict*) – dictionnaire contenant toutes les informations à propos des ménages
- **l_hhold** (*list*) – liste de caractéristiques attachées au ménage
- **index** (*int*) – indice du ménage
- **common** (*Common*) – instance de la classe Common
- **p0** (*Person*) – premier conjoint
- **p1** (*Person*) – deuxième conjoint (pour les couples)

`set_other_years(common)`

Fonction qui fixe les années de retraite partielle et complète (pour les couples) ainsi que les années dans lesquelles la consommation avant et après la retraite sont évaluées.

Paramètres **common** (*Common*) – instance de la classe Common

Le module Macro contient des classes qui contiennent des paramètres communs à tous les ménages ; et qui génèrent des processus stochastiques pour les rendements, les salaires et les autres prix.

class CPR.`macro.CommonParameters`(*nsim, non_stochastic, extra_params*)

Cette classe fixe et contient les paramètres communs à tous les ménages.

Paramètres

- **nsim** (*int*) – nombre de simulations
- **non_stochastic** (*bool*) – Vrai s’il s’agit d’une simulation stochastique, Faux sinon
- **extra_params** (*dict*) – dictionnaire de paramètres supplémentaires

set_limits(*name*)

Fixer les limites de cotisation pour les REER et les CELI.

Paramètres name (*str*) – RRSP (REER) ou TFSA (CELI)

prepare_ympc()

MGAP pré-réforme utilisé pour ajuster les prestations de PD pour le RPC

prepare_cpp()

Fixer des pourcentages pour les prestations de RPC/RRQ.

class CPR.`macro.Prices`(*common, extra_params*)

Cette classe calcule la série temporelle pour les rendements sur actifs, les taux d’intérêt sur les dettes, les profils de salaires déterministes, la croissance des prix immobiliers et le ratio prix-loyers.

Paramètres

- **common** (*Common*) – instance de la classe Common
- **extra_params** (*dict*) – dictionnaire de paramètres supplémentaires

simulate_ret(*asset, common*)

Simuler N séries de longueur T de rendements nominaux distribués de façon log-normale, avec une auto-corrélation rho.

Paramètres

- **asset** (*str*) – type d’actif
- **common** (*Common*) – instance de la classe Common

Renvoie Tableau de rendements nominaux

Type renvoyé numpy.array

compute_params_process(*mu, rho, sigma*)

Convertit la moyenne arithmétique μ et la volatilité σ des rendements et l'autocorrélation ρ du log des rendements en α , ρ and σ_ϵ dans le processus :

$$\ln(1 + r_t) = \alpha + \rho * \ln(1 + r_{t-1}) + \epsilon, \text{ où } \epsilon \sim N(0, \sigma_\epsilon).$$

Paramètres

- **mu** (*float*) – moyenne arithmétique
- **rho** (*float*) – autocorrélation
- **sigma** (*float*) – écart-type

Renvoie

- *float* – coefficient AR(1) (α)
- *float* – Écart-type du terme d'erreur (σ_ϵ)

simulate_housing(*common*)

Simule une série de croissance nominale des prix immobiliers (sous la forme $\ln(1 + r)$) et de une série de ratio prix-loyers.

Paramètres **common** (*Common*) – instance de la classe Common

Renvoie

- *numpy.array* – Tableau (array) de croissance nominale des prix immobiliers
- *numpy.array* – Tableau (array) de ratios prix-loyers

prepare_inflation_factors(*common*)

Compute inflation factors with base year 2018.

Paramètres **common** (*Common*) – instance de la classe Common

Renvoie Dictionnaire de facteurs d'inflation pour chaque année

Type renvoyé dict

simulate_interest_debt()

Crée N séries de longueur T de taux d'intérêt annuel nominal pour chaque type de dette

Renvoie Dictionnaire de taux d'intérêt par type de dette et par année

Type renvoyé dict

attach_diff_log_wages()

Crée un dictionnaire de différences dans le log des salaires par scolarité et par âge.

Renvoie Dictionnaire de différences dans le log des salaires par scolarité et par âge

Type renvoyé dict

initialize_factors()

Cette fonction crée une instance de life.table par genre et par province.

Renvoie dictionnaire de facteurs de rentes par genre et par province

Type renvoyé dict

À l'aide de plusieurs classes et fonctions, ce module gère tous les actifs : régimes de retraite, épargne individuelle (enregistrée et non enregistrée), logement, entreprises. Il utilise les cotisations et l'espace entrés comme inputs ainsi que les rendements générés dans le module Macro. Il met à jour toutes les variables reliées aux actifs à mesure que la simulation progresse.

class CPR.assets.**ContributionRoom**(*init_room_rrsp, init_room_tfsa*)

Cette classe gère l'espace de cotisation aux CELI et aux REER.

Tous les montants sont nominaux.

Paramètres

- **init_room_rrsp** (*float*) – espace de cotisation REER initialement disponible
- **init_room_tfsa** (*float*) – espace de cotisation CELI initialement disponible

compute_contributions(*p, year, common, prices*)

Fonction qui met à jour l'espace de cotisation pour les REER et les CELI, en utilisant les 2 autres fonctions ci-dessous (qui elles-mêmes appellent les autres fonctions de la classe).

Paramètres

- **p** (*Person*) – instance de la classe Person
- **year** (*int*) – année
- **common** (*Common*) – instance de la classe Common
- **prices** (*Prices*) – instance de la classe Prices

update_rrsp_room(*p, year, common*)

Fonction qui met à jour l'espace de cotisation REER.

Paramètres

- **p** (*Person*) – instance de la classe Person
- **year** (*int*) – année
- **common** (*Common*) – instance de la classe Common

update_tfsa_room(*p, year, common*)

Fonction qui met à jour l'espace de cotisation CELI.

Paramètres

- **p** (*Person*) – instance de la classe Person
- **year** (*int*) – année
- **common** (*Common*) – instance de la classe Common

adjust_db_contributions(*p, year, common*)

Fonction qui ajuste l'espace de cotisation REER en fonction des cotisations aux RPA à PD.

Paramètres

- **p** (*Person*) – instance de la classe Person
- **year** (*int*) – année
- **common** (*Common*) – instance de la classe Common

adjust_dc_contributions(*p, year*)

Fonction qui ajuste l'espace de cotisation REER en fonction des cotisations aux RPA à CD.

Si l'espace de cotisation est insuffisant pour les cotisations REER prévues / planifiées, les cotisations « excédentaires » sont dirigées vers le CELI.

Paramètres

- **p** (*Person*) – instance de la classe Person
- **year** (*int*) – année

adjust_employees_contributions(*p*)

Fonction qui calcule les cotisations d'employé aux RPA à CD (utilisées plus tard afin de calculer les impôts).

Paramètres **p** (*Person*) – instance de la classe Person

adjust_rrsp_contributions(*acc, p, year*)

Fonction qui ajuste l'espace de cotisation REER en fonction des cotisations autres qu'à des RPA.

Si l'espace de cotisation est insuffisant pour les cotisations REER prévues / planifiées, les cotisations « excédentaires » sont dirigées vers le CELI.

Paramètres

- **acc** (*str*) – type de compte
- **p** (*Person*) – instance de la classe Person
- **year** (*int*) – année

adjust_tfsa_contributions(*p, year, common, prices*)

Fonction qui ajuste l'espace de cotisation CELI en fonction des cotisations CELI (y compris les cotisations « excédentaires » aux REER et aux RPA à CD).

Si l'espace de cotisation est insuffisant pour les cotisations CELI prévues / planifiées, les cotisations « excédentaires » sont dirigées vers des comptes non enregistrés.

Paramètres

- **p** (*Person*) – instance de la classe Person
- **year** (*int*) – année
- **common** (*Common*) – instance de la classe Common
- **prices** (*Prices*) – instance de la classe Prices

adjust_rrif(*p, year, common, prices*)

Fonction qui ajuste les comptes REER et RPA à CD en fonction des retraits obligatoires aux FERR (les FERR ne sont pas modélisés séparément dans cette version).

Paramètres

- **p** (*Person*) – instance de la classe Person
- **year** (*int*) – année
- **common** (*Common*) – instance de la classe Common
- **prices** (*Prices*) – instance de la classe Prices

Renvoie Montant nominal qui doit être retiré.

Type renvoyé float

adjust_unreg_contributions(*p, year*)

Fonction qui ajuste les cotisations aux comptes non enregistrés en fonction des cotisations CELI « excédentaires » qui sont dirigées vers eux.

Paramètres

- **p** (*Person*) – instance de la classe Person
- **year** (*int*) – année

reset()

Remet les espaces de cotisation REER et CELI à leurs valeurs initiales.

class CPR.assets.**FinAsset**(*p, hh, acc*)

Cette classe gère les comptes enregistrés. Tous les montants sont nominaux.

Paramètres

- **p** (*Person*) – instance de la classe Person
- **hh** (*Hhold*) – ménage
- **acc** (*str*) – type de compte

update(*d_returns, year, common, prices*)

Fonction qui met à jour le solde pour tenir compte des cotisations, des retraits et des rendements.

Paramètres

- **d_returns** (*dict*) – dictionnaire de rendements
- **year** (*int*) – année
- **common** (*Common*) – instance de la classe Common
- **prices** (*Prices*) – instance de la classe Prices

rate(*d_returns, year*)

Fonction qui calcule le taux de rendement étant donné la composition des actifs dans le compte.

Paramètres

- **d_returns** (*dict*) – dictionnaire de rendements
- **year** (*int*) – année

Renvoie Taux de rendement (net de frais).

Type renvoyé float

rrif_withdrawal(*p, common*)

Fonction gérant les retraits obligatoires aux FERR.

Paramètres

- **p** (*Person*) – instance de la classe Person
- **common** (*Common*) – instance de la classe Common

Renvoie Montant du retrait obligatoire.

Type renvoyé float

liquidate()

Fonction qui liquide le compte et met le solde, les cotisations et les retraits à zéro.

Renvoie Montant de la liquidation (avant impôts).

Type renvoyé float

reset()

Remet le solde et les retraits à leur valeur initiale.

class CPR.assets.**UnregAsset**(*p, hh, prices*)

Cette classe gère les comptes non enregistrés.

Tous les montants sont nominaux.

Paramètres

- **p** (*Person*) – instance de la classe Person
- **hh** (*Hhold*) – ménage
- **prices** (*Prices*) – instance de la classe Prices

update(*d_returns, year, common, prices*)

Fonction qui met à jour le solde en fonction des cotisations, retraits et rendements.

Paramètres

- **d_returns** (*dict of float*) – dictionnaire de rendements
- **year** (*int*) – année
- **common** (*Common*) – instance de la classe Common

— **prices** (*Prices*) – instance de la classe *Prices*

compute_income(*d_returns*, *year*)

Fonction qui calcule les nouveaux gains en capital et le revenu imposable de dividendes et d'intérêts.

Paramètres

— **d_returns** (*dict of float*) – dictionnaire de rendements

— **year** (*int*) – année

rate(*d_returns*, *year*)

Fonction qui calcule le taux de rendement étant donné la composition d'actifs dans le compte.

Paramètres

— **d_returns** (*dict of float*) – dictionnaire de rendements

— **year** (*int*) – année

Renvoie Taux de rendement (net de frais).

Type renvoyé float

update_balance()

Fonction qui met à jour le solde et distingue les fonds non imposables (c.-à-d. le solde antérieur après impôts), les gains en capital, les dividendes et les intérêts.

prepare_withdrawal()

Fonction qui sépare un retrait du solde à la fin de la période et identifie séparément les fonds non imposables, les gains en capital, les dividendes et les intérêts.

adjust_income()

Fonction qui ajuste le revenu d'investissement (dividendes et intérêts) en fonction des retraits.

adjust_cap_losses(*realized_cap_gains*)

Fonction qui calcule les pertes en capital d'années antérieures utilisées comme déduction contre des gains en capital, et ajuste en conséquence les pertes en capital réalisées.

Paramètres **realized_cap_gains** (*float*) – gains en capital

Renvoie Pertes en capital (à déduire).

Type renvoyé float

adjust_final_balance()

Fonction qui ajuste le solde final en fonction des retraits, dividendes et intérêts.

liquidate()

Fonction qui liquide le compte et ajuste les pertes en capital.

reset()

Fonction qui remet à leur valeur initiale le solde, les gains en capital et les retraits.

class CPR.assets.**RppDC**(*p*, *common*)

Cette classe gère les RPA à CD.

Tous les montants sont nominaux.

Paramètres

— **p** (*Person*) – instance de la classe *Person*

— **common** (*Common*) – instance de la classe *Common*

class CPR.assets.**RppDB**(*p*)

Cette classe gère les RPA à PD.

Tous les montants sont nominaux.

Paramètres **p** (*Person*) – instance de la classe *Person*

compute_benefits(*p*, *common*)

Fonction qui calcule les prestations de RPA à PD et les ajuste pour tenir compte de l'intégration au RPC/RRQ.

Si les prestations de RPA sont plus faibles que les prestations de RPC/RRQ, les prestations de RPA sont fixées à zéro une fois que débutent les prestations de retraite du RPC/RRQ.

Paramètres

- **p** (*Person*) – instance de la classe *Person*
- **common** (*Common*) – instance de la classe *Common*

adjust_for_penalty(*p, common*)

Fonction qui calcule une pénalité pour les individus débutant leurs prestations de RPA à PD de façon « hâtive », c.-à-d. avant d'avoir accumulé le nombre maximal d'années de service, s'ils sont plus jeunes que l'âge auquel les prestations peuvent débiter sans pénalité (« l'âge de retraite anticipée »).

Par défaut, la pénalité s'applique aux individus qui débutent leurs prestations avant d'avoir atteint 35 années de service et avant 62 ans. Ces valeurs peuvent être modifiées.

Paramètres

- **p** (*Person*) – instance de la classe *Person*
- **common** (*Common*) – instance de la classe *Common*

compute_cpp_adjustment(*p, common*)

Fonction qui calcule un ajustement (une réduction) des prestations de RPA à PD pour tenir compte de l'intégration avec le RPC/RRQ.

Paramètres

- **p** (*Person*) – instance de la classe *Person*
- **common** (*Common*) – instance de la classe *Common*

Renvoie Montant de l'ajustement pour l'intégration au RPC/RRQ.

Type renvoyé float

reset()

Fonction qui remet à leur valeur initiale les prestations et taux de cotisation.

class CPR.assets.**RealAsset**(*d_hh, resid*)

Cette classe gère le logement et les résidences.

Tous les montants sont nominaux.

Paramètres

- **d_hh** (*dict of float*) – dictionnaire contenant les valeurs des résidences
- **resid** (*str*) – résidence principale ou secondaire

update(*growth_rates, year, prices*)

Fonction qui met à jour le solde (les valeurs des résidences) en fonction de la croissance du prix.

Paramètres

- **growth_rates** (*dict*) – rendement nominal sur le logement (gains en capital)
- **year** (*int*) – année
- **prices** (*Prices*) – instance de la classe *Prices*

liquidate()

Fonction qui liquide l'actif, calcule les gains en capital si applicable, et fixe le solde à zéro.

Renvoie Montant de la liquidation d'actif (avant impôts).

Type renvoyé float

reset()

Fonction qui remet la valeur de la résidence à sa valeur initiale et fixe les gains en capital à zéro.

impute_rent(*hh, year, common, prices*)

Fonction qui calcule le loyer imputé.

Paramètres

- **hh** (*Hhold*) – ménage
- **year** (*int*) – année
- **prices** (*Prices*) – instance de la classe *Prices*

class CPR.assets.**Business**(*d_hh*)

Cette classe gère une entreprise (comme un actif détenu par le ménage).

Tous les montants sont nominaux.

Paramètres `d_hh` (*dict of float*) – dictionnaire contenant la valeur de l’entreprise

update(*d_business_returns, year, prices*)

Fonction qui met à jour le solde et les dividendes.

Paramètres

— **d_business_returns** (*dict*) – rendements sur les actifs d’entreprise

— **year** (*int*) – année

— **prices** (*Prices*) – instance de la classe Prices

liquidate(*common*)

Fonction qui liquide le compte (actifs d’entreprise), calcule les gains en capital et fixe le solde à zéro.

Paramètres `common` (*Common*) – instance de la classe Common

Renvoie Prix de vente des actifs d’entreprise.

Type renvoyé float

reset()

Remet à leur valeur initiale la valeur de l’entreprise et les gains en capital.

Le module Debts gère les dettes (paiements et soldes) dans la simulation.

class CPR.debts.**Debt**(*name, d_hh, common, prices*)

Cette classe gère toutes les dettes. Tous les montants sont nominaux.

Paramètres

- **name** (*str*) – nom de la dette
- **d_hh** (*dict*) – dictionnaire contenant les soldes initiaux des dettes et les paiements mensuels
- **common** (*Common*) – instance de la classe Common
- **prices** (*Prices*) – instance de la classe Prices

estimate_init_term(*common, prices*)

Estime le terme de la dette.

Paramètres

- **common** (*Common*) – instance de la classe Common
- **prices** (*Prices*) – instance de la classe Prices

Renvoie durée de la dette

Type renvoyé int

update(*year, rate, prices*)

Met à jour le paiement annuel et le solde.

Paramètres

- **year** (*int*) – année
- **rate** (*float*) – taux d'intérêt
- **prices** (*Prices*) – instance de la classe Prices

reset()

Remet le solde et la durée à leur valeur initiale.

Le module Taxes utilise le SRD pour calculer les impôts et les prestations pour chaque ménage.

CPR. taxes. **compute_after_tax_amount**(*hh, year, common, prices*)

Calcule le rendement nominal net et le montant de retrait net pour les actifs non enregistrés.

Paramètres

- **hh** (*Hhold*) – ménage
- **year** (*int*) – année
- **common** (*Common*) – instance de la classe Common
- **prices** (*Prices*) – instance de la classe Prices

CPR. taxes. **file_household**(*hh, year, common, prices*)

Remplit la déclaration de revenus du ménage à l'aide du SRD.

Paramètres

- **hh** (*Hhold*) – ménage
- **year** (*int*) – année
- **common** (*Common*) – instance de la classe Common
- **prices** (*Prices*) – instance de la classe Prices

Renvoie instance de la classe Hhold après impôt

Type renvoyé *Hhold*

CPR. taxes. **file_household_inc_to_tax**(*hh, year, common, prices*)

Déclare les revenus imposables (intérêts et dividendes) provenant d'actifs non enregistrés du ménage ayant de tels revenus.

Paramètres

- **hh** (*Hhold*) – ménage
- **year** (*int*) – année
- **common** (*Common*) – instance de la classe Common
- **prices** (*Prices*) – instance de la classe Prices

Renvoie instance de la classe Hhold après impôt

Type renvoyé *Hhold*

CPR. taxes. **get_gis_oas_allowances**(*hh, hh_tax, year, prices*)

Calcule les prestations nominales de PSV et de SRG.

Paramètres

- **hh** (*Hhold*) – ménage
- **hh_tax** (*Hhold*) – ménage
- **year** (*int*) – année
- **prices** (*Prices*) – instance de la classe Prices

Ce module contient les fonctions requises pour convertir tous les actifs financiers en rentes au moment de la retraite (du premier et/ou du deuxième conjoint). Une des fonctions appelle le facteur de rente construit à l'aide du module Life.

`CPR.annuities.compute_partial_annuities`(*hh, d_returns, year, prices*)

Fonction qui convertit partiellement les actifs en rentes au moment de la retraite du premier conjoint, en utilisant les autres fonctions ci-dessous.

Paramètres

- **hh** (`Hhold`) – ménage
- **d_returns** (`dict`) – dictionnaire de rendements
- **year** (`int`) – année
- **prices** (`Prices`) – instance de la classe Prices

`CPR.annuities.compute_annuities`(*hh, d_returns, year, prices*)

Fonction qui convertit entièrement les actifs en rentes au moment de la retraite du dernier conjoint, à l'aide des autres fonctions ci-dessous.

Paramètres

- **hh** (`Hhold`) – ménage
- **d_returns** (`dict`) – dictionnaire de rendements
- **year** (`int`) – année
- **prices** (`Prices`) – instance de la classe Prices

`CPR.annuities.liquidate_fin_assets`(*p*)

Fonction qui liquide les actifs financiers.

Paramètres **p** (`Person`) – instance de la classe Person

`CPR.annuities.compute_factors`(*hh, p, rate, prices*)

Fonction calculant un facteur individuel pour les rentes constant en termes réels.

Nous utilisons un taux ajusté pour amenuiser toute volatilité excédentaire du facteur, car nous prenons le rendement total sur les obligations de LT dans l'année d'achat de la rente. Idéalement, la structure complète des taux d'intérêt devrait plutôt être utilisée, de telle sorte que le retour à la moyenne lisserait le prix des rentes.

Paramètres

- **hh** (`Hhold`) – ménage
- **p** (`Person`) – instance de la classe Person

- **rate** (*float*) – taux d'intérêt
- **prices** (*Prices*) – instance de la classe Prices

CPR.annuities.**convert_to_real_annuities**(*p, year, prices*)

Fonction qui convertit les actifs en rentes en termes réels.

Paramètres

- **p** (*Person*) – instance de la classe Person
- **year** (*int*) – année
- **prices** (*Prices*) – instance de la classe Prices

Le module Life contient la classe qui calcule les facteurs de rente pour tous les individus dans la simulation, à l'aide de tables de mortalité par province et par genre fournies par Statistique Canada.

class `CPR.life.life.table`(*prov='qc', scenario='M', gender='males', web=False*)

Classe qui calcule les facteurs de rente par province, genre, âge et année de naissance.

Paramètres

- **prov** (*str*) – province; défaut=qc
- **scenario** (*str*) – scénario de longévité; défaut=M (mortalité moyenne)
- **gender** (*str*) – genre; défaut=males (masculin)
- **web** (*bool*) – True (Vrai) pour récupérer les données depuis internet (option non disponible pour le moment), False (Faux) sinon; défaut=False

pull_history(*prov='qc', gender='males'*)

Fonction qui obtient les tables de mortalité historiques par province et par genre.

Paramètres

- **prov** (*str*) – province; défaut=qc
- **gender** (*str*) – genre; défaut=males (masculin)

splice()

Fonction qui ajoute des les probabilités de survie pour l'année de naissance 2012 et les suivantes.

compute_annuity_factor(*byear, agestart, rate*)

Fonction qui calcule un facteur de rente.

Paramètres

- **byear** (*int*) – année de naissance
- **agestart** (*int*) – âge auquel débute la rente
- **rate** (*float*) – interest rate

Renvoie Facteur de rente.

Type renvoyé float

Balance Sheets

Ce module contient des fonctions qui calculent les différents éléments du bilan pour chaque ménage à différents moments du processus de simulation. Un ensemble de variables sont calculées en termes réels (avec 2018 comme année de base) et ajoutées au dataframe d’output récupéré par le module *main*.

À 55 ans ou dans l’année précédant la retraite, si cette dernière se produit avant 56 ans, les salaires, la consommation, les pensions antérieures et les soldes des comptes (REER, autres comptes enregistrés, CELI et comptes non enregistrés, ainsi que RPA à CD) sont calculés. Ils sont entreposés dans des variables dont le nom se termine par “_bef”.

À 65 ans ou dans l’année de la retraite, si cette dernière se produit plus tard, la consommation; les pensions antérieures; les rentes achetées avec les (soldes des) actifs financiers; les prestations de RPA à PD, de RPC/RRQ et de PSV/SRG/Allocations; ainsi que les valeurs des résidences et les soldes hypothécaires sont entreposés dans des variables dont le nom se termine par “_aft”.

Pour les couples qui ne prennent pas leur retraite en même temps, les salaires, les pensions antérieures, les soldes de comptes et les rentes du premier conjoint à prendre sa retraite sont calculés. Ils sont entreposés dans des variables dont le nom se termine par “_part”. Le nom des variables décrivant le conjoint qui prend sa retraite en dernier débute avec “s_”.

`CPR.balance_sheets.compute_bs_bef_ret`(*hh, year, common, prices*)

Fonction qui calcule le bilan avant la retraite.

Paramètres

- **hh** (*Hhold*) – ménage
- **year** (*int*) – année
- **common** (*Common*) – instance de la classe *Common*
- **prices** (*Prices*) – instance de la classe *Prices*

`CPR.balance_sheets.compute_cons_bef_ret`(*hh, year, prices*)

Fonction qui calcule la consommation avant la retraite.

Paramètres

- **hh** (*Hhold*) – ménage
- **year** (*int*) – année
- **prices** (*Prices*) – instance de la classe *Prices*

`CPR.balance_sheets.compute_bs_after_ret`(*hh, year, common, prices*)

Fonction qui calcule le bilan après la retraite.

Paramètres

- **hh** (*Hhold*) – ménage
- **year** (*int*) – année
- **common** (*Common*) – instance de la classe Common
- **prices** (*Prices*) – instance de la classe Prices

CPR.balance_sheets.**add_output**(*hh, year, prices, key*)

Fonction qui extrait les variables d'output.

Paramètres

- **hh** (*Hhold*) – ménage
- **year** (*int*) – année
- **prices** (*Prices*) – instance de la classe Prices
- **key** (*str*) – avant (« bef »), lorsque le premier conjoint prend sa retraite (« part »), ou après la retraite (« aft »)

Dans ce module, la fonction `get_dataset` récupère un jeu de données (simulé). La classe `Results` offre quelques fonctions pour résumer et fusionner les données et pour évaluer la préparation pour la retraite.

`CPR.analysis.get_dataset()`

Fonction qui renvoie un dataframe de données synthétiques.

Renvoie Dataframe de données synthétiques.

Type renvoyé `pandas.core.frame.DataFrame`

`class CPR.analysis.Results(input_data, output, common, prices, extra_params)`

Cette classe prépare les résultats.

Paramètres

- **input_data** (`pandas.core.frame.DataFrame`) – dataframe d'intrants
- **output** (`pandas.core.frame.DataFrame`) – dataframe d'extrants
- **common** (`Common`) – instance de la classe `Common`
- **prices** (`Prices`) – instance de la classe `Prices`
- **extra_params** (`dict`) – dictionnaire de paramètres supplémentaires

`summarize()`

Fonction pour résumer la simulation.

`merge(add_index=True)`

Fonction pour fusionner les variables d'entrée et de sortie pour créer une base de données.

Paramètres `add_index` (`bool`, `optional`) – ajouter l'indice à la base de données, `True` (Vrai) par défaut

`check_preparedness(factor_couple=2, cons_floor=100, d_cutoffs={20 : 80, 100 : 65})`

Fonction qui introduit un plancher de consommation, calcule l'IPR (NaN si la consommation avant et après la retraite sont en deçà de `cons_floor`), et vérifie la préparation pour la retraite.

Paramètres

- **factor_couple** (`int`, `optional`) – facteur de normalisation du revenu pour les couples, par défaut 2
- **cons_floor** (`int`, `optional`) – plancher de consommation, par défaut 100
- **d_cutoffs** (`dict`, `optional`) – seuils, par défaut {20 : 80, 100 : 65}

Dans ce module, plusieurs fonctions utilisées dans le reste du code sont disponibles pour créer des dictionnaires, modifier des paramètres ou convertir des prix, entre autres aspects.

`CPR.tools.get_params(file, numerical_key=False)`

Fonction pour créer un dictionnaire à partir d'un fichier .csv.

Paramètres

- **file** (*_io.TextIOWrapper*) – fichier csv
- **numerical_key** (*bool, optional*) – clé numérique, par défaut à Faux

Renvoie Dictionnaire de paramètres.

Type renvoyé dict

`CPR.tools.add_params_as_attr(inst, file)`

Fonction pour ajouter des paramètres à une instance de la classe.

Paramètres

- **inst** (*object*) – instance de classe
- **file** (*_io.TextIOWrapper*) – fichier csv

`CPR.tools.change_params(inst, extra_params)`

Fonction qui met à jour la valeur de certains paramètres dans inst avec des valeurs prises dans extra_params.

Paramètres

- **inst** (*dict*) – paramètres
- **extra_params** (*dict*) – paramètres à mettre à jour ainsi que leurs nouvelles valeurs

`CPR.tools.create_nom_real(year, prices)`

Création d'une fonction convertissant les prix de réels à nominaux et de nominaux à réels, avec 2018 comme année de base.

Paramètres

- **year** (*int*) – année
- **prices** (*Prices*) – instance de la classe Prices

Renvoie

- *function* – Fonction convertissant le nominal en réel.
- *function* – Fonction convertissant le réel en nominal.

CPR.tools.**create_nom**(*year, prices*)

Création d'une fonction convertissant les prix de réels à nominaux, avec 2018 comme année de base, utilisée à certaines fins spécifiques ailleurs dans le code.

Paramètres

- **year** (*int*) – année
- **prices** (*Prices*) – instance de la classe Prices

Renvoie Fonction convertissant le réel en nominal.

Type renvoyé fonction

CPR.tools.**create_real**(*year, prices*)

Création d'une fonction convertissant les prix de nominaux à réels, avec 2018 comme année de base.

Paramètres

- **year** (*int*) – année
- **prices** (*Prices*) – instance de la classe Prices

Renvoie fonction convertissant le nominal en réel

Type renvoyé fonction

CHAPITRE 14

Tutoriels

14.1 Utilisation de base du CPR (vidéo en anglais)

14.2 Expériences avec le CPR (vidéo en anglais)

Cliquer [ici](#) pour accéder au notebook.

Contributeurs et droits d'utilisation

15.1 Contributeurs

David Boisclair, Ismaël Choinière-Crèveœur, Pierre-Carl Michaud, Pierre-Yves Yann

15.2 Contact

Institut sur la retraite et l'épargne

15.3 Droits d'utilisation

Le CPR 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 16

Documentation en PDF

Documentation pdf

C

CPR.annuities, 27
CPR.balance_sheets, 31
CPR.simulator, 7
CPR.taxes, 25
CPR.tools, 35

A

add_output() (dans le module *CPR.balance_sheets*), 32
 add_params_as_attr() (dans le module *CPR.tools*), 35
 adjust_cap_losses() (méthode *CPR.assets.UnregAsset*), 20
 adjust_contributions() (dans le module *CPR.simulator*), 8
 adjust_db_contributions() (méthode *CPR.assets.ContributionRoom*), 18
 adjust_dc_contributions() (méthode *CPR.assets.ContributionRoom*), 18
 adjust_employees_contributions() (méthode *CPR.assets.ContributionRoom*), 18
 adjust_final_balance() (méthode *CPR.assets.UnregAsset*), 20
 adjust_for_penalty() (méthode *CPR.assets.RppDB*), 21
 adjust_income() (méthode *CPR.assets.UnregAsset*), 20
 adjust_rrif() (méthode *CPR.assets.ContributionRoom*), 18
 adjust_rrsp_contributions() (méthode *CPR.assets.ContributionRoom*), 18
 adjust_tfsa_contributions() (méthode *CPR.assets.ContributionRoom*), 18
 adjust_unreg_contributions() (méthode *CPR.assets.ContributionRoom*), 18
 attach_diff_log_wages() (méthode *CPR.macro.Prices*), 16

B

Business (classe dans *CPR.assets*), 21

C

change_params() (dans le module *CPR.tools*), 35
 check_liquidation() (dans le module *CPR.simulator*), 9
 check_preparedness() (méthode *CPR.analysis.Results*), 33

check_tax_unreg() (dans le module *CPR.simulator*), 8
 claim_cpp() (dans le module *CPR.simulator*), 8
 CommonParameters (classe dans *CPR.macro*), 15
 compute_after_tax_amount() (dans le module *CPR.taxes*), 25
 compute_annuities() (dans le module *CPR.annuities*), 27
 compute_annuity_factor() (méthode *CPR.life.life.table*), 29
 compute_benefits() (méthode *CPR.assets.RppDB*), 20
 compute_bs_after_ret() (dans le module *CPR.balance_sheets*), 31
 compute_bs_bef_ret() (dans le module *CPR.balance_sheets*), 31
 compute_cons_bef_ret() (dans le module *CPR.balance_sheets*), 31
 compute_contributions() (méthode *CPR.assets.ContributionRoom*), 17
 compute_cpp_adjustment() (méthode *CPR.assets.RppDB*), 21
 compute_factors() (dans le module *CPR.annuities*), 27
 compute_income() (méthode *CPR.assets.UnregAsset*), 20
 compute_params_process() (méthode *CPR.macro.Prices*), 16
 compute_partial_annuities() (dans le module *CPR.annuities*), 27
 compute_rpp() (dans le module *CPR.simulator*), 9
 contribute_cpp() (dans le module *CPR.simulator*), 8
 ContributionRoom (classe dans *CPR.assets*), 17
 convert_to_real_annuities() (dans le module *CPR.annuities*), 28
 CPR.annuities
 module, 27
 CPR.balance_sheets
 module, 31
 CPR.simulator
 module, 7
 CPR.taxes

module, 25
 CPR.tools
 module, 35
 create_hh() (dans le module CPR.initialisation), 13
 create_nom() (dans le module CPR.tools), 35
 create_nom_real() (dans le module CPR.tools), 35
 create_real() (dans le module CPR.tools), 36
 create_shocks() (méthode CPR.initialisation.Person),
 13
 create_wage_profile() (méthode
 CPR.initialisation.Person), 13

D

Debt (classe dans CPR.debts), 23

E

estimate_init_term() (méthode CPR.debts.Debt), 23
 extract_time_series() (dans le module
 CPR.simulator), 7

F

file_household() (dans le module CPR.taxes), 25
 file_household_inc_to_tax() (dans le module
 CPR.taxes), 25
 FinAsset (classe dans CPR.assets), 19

G

get_assets() (dans le module CPR.simulator), 9
 get_benefits_cpp() (dans le module CPR.simulator),
 9
 get_contributions_assets() (dans le module
 CPR.simulator), 10
 get_dataset() (dans le module CPR.analysis), 33
 get_gis_oas_allowances() (dans le module
 CPR.taxes), 25
 get_inc_rrsp() (dans le module CPR.simulator), 9
 get_other_non_taxable() (dans le module
 CPR.simulator), 10
 get_other_taxable() (dans le module
 CPR.simulator), 9
 get_params() (dans le module CPR.tools), 35

H

Hhold (classe dans CPR.initialisation), 13

I

impute_rent() (méthode CPR.assets.RealAsset), 21
 initialize_cpp_account() (dans le module
 CPR.simulator), 7
 initialize_factors() (méthode CPR.macro.Prices),
 16

L

liquidate() (méthode CPR.assets.Business), 22

liquidate() (méthode CPR.assets.FinAsset), 19
 liquidate() (méthode CPR.assets.RealAsset), 21
 liquidate() (méthode CPR.assets.UnregAsset), 20
 liquidate_fin_assets() (dans le module
 CPR.annuities), 27

M

manage_liquidations() (dans le module
 CPR.simulator), 8
 merge() (méthode CPR.analysis.Results), 33
 module
 CPR.annuities, 27
 CPR.balance_sheets, 31
 CPR.simulator, 7
 CPR.taxes, 25
 CPR.tools, 35

P

Person (classe dans CPR.initialisation), 13
 prepare_cpp() (méthode
 CPR.macro.CommonParameters), 15
 prepare_inflation_factors() (méthode
 CPR.macro.Prices), 16
 prepare_taxes() (dans le module CPR.simulator), 9
 prepare_wages() (dans le module CPR.simulator), 7
 prepare_withdrawal() (méthode
 CPR.assets.UnregAsset), 20
 prepare_ympe() (méthode
 CPR.macro.CommonParameters), 15
 Prices (classe dans CPR.macro), 15
 pull_history() (méthode CPR.life.life.table), 29

R

rate() (méthode CPR.assets.FinAsset), 19
 rate() (méthode CPR.assets.UnregAsset), 20
 RealAsset (classe dans CPR.assets), 21
 reset() (méthode CPR.assets.Business), 22
 reset() (méthode CPR.assets.ContributionRoom), 19
 reset() (méthode CPR.assets.FinAsset), 19
 reset() (méthode CPR.assets.RealAsset), 21
 reset() (méthode CPR.assets.RppDB), 21
 reset() (méthode CPR.assets.UnregAsset), 20
 reset() (méthode CPR.debts.Debt), 23
 reset_accounts() (dans le module CPR.simulator), 10
 Results (classe dans CPR.analysis), 33
 RppDB (classe dans CPR.assets), 20
 RppDC (classe dans CPR.assets), 20
 rrif_withdrawal() (méthode CPR.assets.FinAsset), 19
 run_simulations() (dans le module CPR.main), 5

S

set_limits() (méthode
 CPR.macro.CommonParameters), 15

set_other_years() (méthode
CPR.initialisation.Hhold), 13
simulate() (dans le module CPR.simulator), 7
simulate_housing() (méthode CPR.macro.Prices), 16
simulate_interest_debt() (méthode
CPR.macro.Prices), 16
simulate_ret() (méthode CPR.macro.Prices), 15
splice() (méthode CPR.life.life.table), 29
summarize() (méthode CPR.analysis.Results), 33

T

table (classe dans CPR.life.life), 29

U

UnregAsset (classe dans CPR.assets), 19
update() (méthode CPR.assets.Business), 22
update() (méthode CPR.assets.FinAsset), 19
update() (méthode CPR.assets.RealAsset), 21
update() (méthode CPR.assets.UnregAsset), 19
update() (méthode CPR.debts.Debt), 23
update_ages() (dans le module CPR.simulator), 8
update_assets() (dans le module CPR.simulator), 8
update_balance() (méthode CPR.assets.UnregAsset),
20
update_debts() (dans le module CPR.simulator), 8
update_rrsp_room() (méthode
CPR.assets.ContributionRoom), 17
update_tfsa_room() (méthode
CPR.assets.ContributionRoom), 17