CPR

Release 1.2.2

RSI Team / Équipe IRE

May 31, 2021

CONTENTS

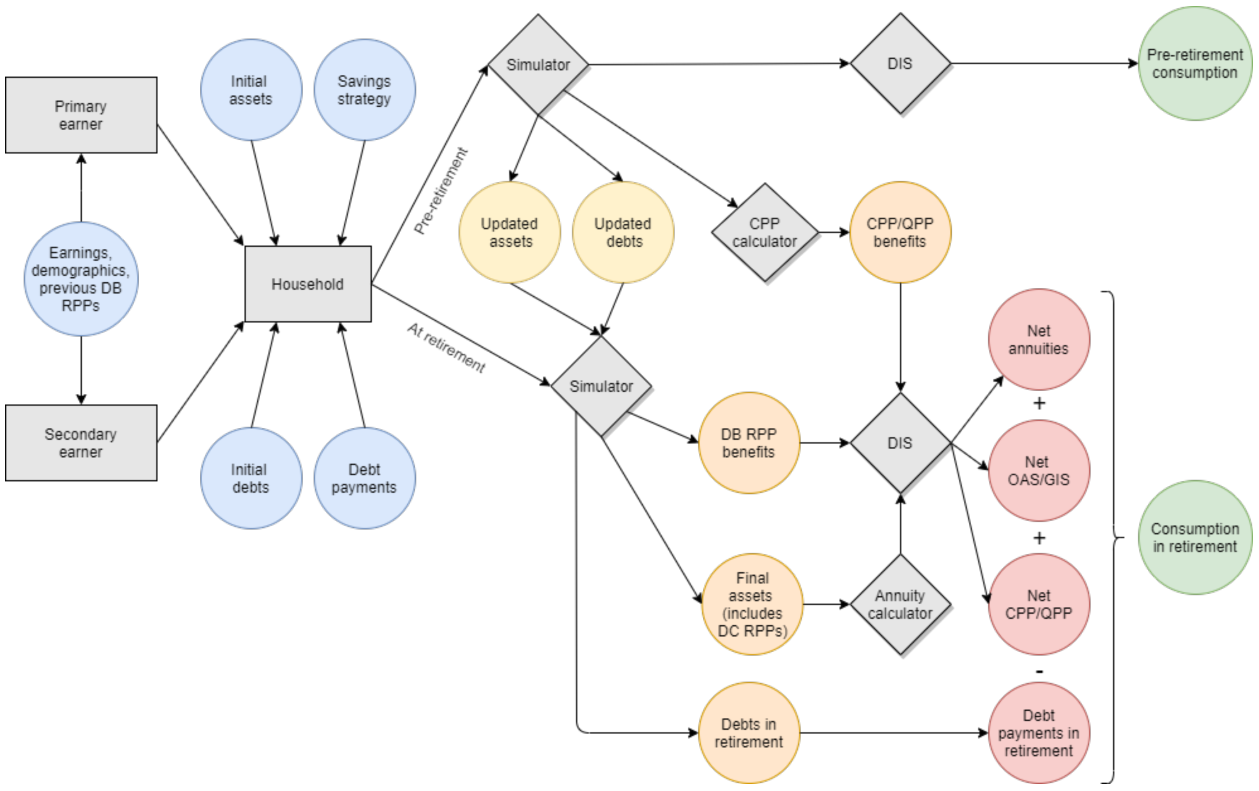
The Canadians' Preparation for Retirement (CPR) tool was developed by a team at HEC Montréal's Retirement and Savings Institute (RSI), with financial support from the National Pension Hub at the Global Risk Institute (GRI). It allows to compute retirement preparation for any household or group of households for which the user has the required input (personal and financial information) at any given point in time. It has been used with 2018 proprietary data to prepare a report published by in June 2020.

As mentioned elsewhere, the CPR is provided "as is" under *an MIT licence*.

The CPR simulates the years between the base year and retirement, then converts all financial wealth into actuarially fair annuities upon retirement and compares post- and pre-retirement situations. Compared to other available tools, the CPR offers complete flexibility and transparency, in that its code is fully and freely available online and it can be modified and adapted at will. Importantly and distinctively, the CPR also allows to perform stochastic simulations, letting several dimensions vary over time according to cutting-edge processes and assumptions. All those can be modified by the sophisticated user, as can the number of replications that the tool uses to average outcomes and determine retirement preparation (or the probability thereof). The current version of the CPR takes 2018 as base year and the tax systems of Quebec and Ontario are currently modelled (other provinces can be approximated using one of these 2 provinces, preferably Ontario).

The CPR outputs a level of "consumption replacement" for each household that passes through the simulator and classifies each household as "prepared" or "unprepared", according to the pre-set criterion. As in reports published by other groups in the 2010s, retirement preparation can be assessed by computing the ratio of post-retirement consumption to pre-retirement consumption and comparing it to a cutoff set at 65% for the top 4 income quintiles, and at 80% for the bottom income quintile. These measures can then be aggregated over all households analyzed to obtain a) the average of the replacement rate, or its distribution; and b) the proportion of households that are deemed "prepared". When using the stochastic version of the tool, preparation probabilities can also be computed and visualized, for one or several households. The figure below illustrates conceptually the parts and flows of the CPR.

The CPR and the other tools it uses (SRD and SRPP) are written in Python, a modern, simple and fast language. In order to use it, it is essential to ensure first that an up-to-date Python distribution is installed, for example using Anaconda. Although not essential, it will also be useful to have a minimal understanding of Python environments and, if possible, vocabulary (e.g. function, class, instance, profile).



CHAPTER

ONE

GETTING STARTED WITH THE CPR

The CPR and the other tools it uses (SRD and SRPP) are written in Python, a modern, simple and fast language. In order to use it, it is essential to ensure first that an up-to-date Python distribution is installed, for example using Anaconda. In all cases, the minimum requirements to use the CPR are Python 3.6+ with *numpy*, *pandas* and *xlrd*. Although not essential, it will also be useful to have a minimal understanding of Python environments and, if possible, vocabulary (e.g. function, class, instance, profile).

1.1 Installing the CPR

The CPR can be easily installed using pip:

```
pip install cpr-rsi
```

Please read the terms of use for *pypi*, the website hosting the package.

The CPR uses two other packages that are produced by the RSI and the Research Chair on Intergenerational Economics (CREEi), which is supported by the RSI: the Disposable Income Simulator (Simulateur de revenu disponible, or SRD; documentation in French), which computes taxes and major benefits; and the Public Pension Plans Simulator (Simulateur de régimes de pensions publiques, or SRPP; documentation in French), which simulates QPP and CPP contributions and benefits. Both the SRD and the SRPP are automatically installed along with the CPR.

In a notebook or a project, the CPR is called by adding the following command:

```
import CPR
```

To uninstall the CPR, the SRD and the SRPP, pip can be used:

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

1.2 Alternative Install

If one is unable to install the CPR, a zip file containing CPR and its dependencies, SRD and SRPP, can be downloaded from Github (choose the file CPR_with_dep.zip in the latest release). To start using the CPR, it suffices to unzip the file in the directory of your choice and either work from that directory or add the directory to the path (for example by using the module *sys*). A jupyter notebook explaining how CPR works, *Tutorial CPR*, is also included in the zip file. If one does not want the CPR anymore, the *cpr*, *srd* and *srpp* folders can simply be deleted.

For further details on how to use the CPR, please consult the *tutorials*.

As mentioned elsewhere, note the CPR is provided "as is" under *an MIT licence*.

If you have questions, comments or suggestions, do not hesitate to *contact us*.

CHAPTER TWO

MAIN

This module calls the simulator.

The function *run_simulations* runs the simulations. Its arguments are used to select whether the stochastic version of the tool is to be used, as well as the number of simulations (replications) that should be carried out. Any parameter value can also be changed here, to use values other than the default ones provided.

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

This function launches the simulations. Any parameter can be changed using extra_params.

Parameters

- **`inputs`** (*`_io.TextIOWrapper`*) – csv file
- **`nsim`** (*`int, optional`*) – number of simulations, by default 1
- **`non_stochastic`** (*`bool, optional`*) – deterministic prices and price-rent ratio, by default False

Returns instance of the class Results

Return type *Results*

CHAPTER THREE

SIMULATOR

This module runs the simulations.

`CPR.simulator.`**`simulate`**`(`*job*, *common*, *prices*`)`
Function that projects assets, RPPs and debts until the time of retirement.

Parameters

- **job** (*tuple(Hhold, int)*) – instance of the class Hhold and simulation number
- **common** (*Common*) – instance of the class Common
- **prices** (Prices) – instance of the class Prices

Returns dictionary containing households' characteristics before and after retirement

Return type dict

`CPR.simulator.`**`extract_time_series`**`(`*sim*, *common*, *prices*`)`
Function that attaches stochastic processes on asset returns to households.

Parameters

- **sim** (*int*) – simulation number
- **common** (*Common*) – instance of the class Common
- **prices** (Prices) – instance of the class Prices

Returns

- *dict* – returns on assets
- *dict* – price-rent ratio

`CPR.simulator.`**`prepare_wages`**`(`*p*, *sim*, *common*, *prices*`)`
Function that attaches wage profiles to individuals.

Parameters

- **p** (Person) – instance of the class Person
- **sim** (*int*) – simulation number
- **common** (*Common*) – instance of the class Common
- **prices** (Prices) – instance of the class Prices

`CPR.simulator.`**`initialize_cpp_account`**`(`*p*, *hh*, *common*`)`
Function that creates a CPP/QPP account and enters past CPP/QPP contributions based on past wages.

Parameters

- **p** (Person) – instance of the class Person

- **hh** (Hhold) – household
- **common** (*Common*) – instance of the class Common

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

Function that computes age for a given year.

Parameters

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

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

Function that updates debt payments and balances.

Parameters

- **hh** (Hhold) – household
- **year** (*int*) – year
- **sim** (*int*) – simulation number
- **common** (*Common*) – instance of the class Common
- **prices** (Prices) – instance of the class Prices

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

Function that adjusts contributions to all account types (RRSP, other registered, TFSA) and to pensions plans (DC and DB RPPs) so as to respect available contribution rooms.

Parameters

- **hh** (Hhold) – household
- **year** (*int*) – year
- **common** (*Common*) – instance of the class Common
- **prices** (Prices) – instance of the class Prices

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

Function that updates assets every year.

Parameters

- **hh** (Hhold) – household
- **year** (*int*) – year
- **d_returns** (*dict*) – dictionary of returns
- **common** (*Common*) – instance of the class Common
- **prices** (Prices) – instance of the class Prices

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

Function that manages the liquidation of assets upon retirement.

Parameters

- **hh** (Hhold) – household
- **year** (*int*) – year
- **common** (*Common*) – instance of the class Common
- **prices** (Prices) – instance of the class Prices

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

Function that records CPP/QPP contributions.

Parameters

- **p** (`Person`) – instance of the class Person
- **year** (*int*) – year
- **common** (*Common*) – instance of the class Common

`CPR.simulator.`**`claim_cpp`**(*p*)

Function to claim CPP/QPP benefits.

Parameters **p** (`Person`) – instance of the class Person

`CPR.simulator.`**`check_tax_unreg`**(*hh*)

Function that checks whether there are taxable returns to assets.

Parameters **hh** (`Hhold`) – household

Returns True or False

Return type bool

`CPR.simulator.`**`check_liquidation`**(*hh*)

Function to check whether there are liquidated assets to be taxed.

Parameters **hh** (`Hhold`) – household

Returns True or False

Return type bool

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

Function to prepare the variables used in the SRD (in nominal terms).

Parameters

- **hh** (`Hhold`) – household
- **year** (*int*) – year
- **common** (*Common*) – instance of the class Common
- **prices** (`Prices`) – instance of the class Prices

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

Function to retrieve assets for social assistance asset test.

Parameters

- **p** (`Person`) – instance of the class Person
- **common** (*Common*) – instance of class Common

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

Compute rpp (DB and pension).

Parameters

- **p** (`Person`) – instance of the class Person
- **nom** (*function*) – function converting to nominal value
- **common** (*Common*) – instance of the class Common

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

Function that computes annual CPP/QPP retirement benefits.

s1 is the part of the supplementary benefit due to higher contribution rates in the 2019-2025 expansion; s2 is the part of the supplementary benefits attributable to changes in the YMPE; and PRB is for post-retirement benefits (for individuals who keep working after claiming – this feature is not used in the CPR, since individuals automatically claim in the year they retire).

Parameters

- **p** (`Person`) – instance of the class Person
- **year** (*int*) – year
- **common** (*Common*) – instance of the class Common

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

Function that computes RRSP income from withdrawals.

Parameters

- **p** (`Person`) – instance of the class Person
- **nom** (*function*) – function converting to nominal value

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

Function that computes other taxable income.

Parameters

- **p** (`Person`) – instance of the class Person
- **nom** (*function*) – function converting to nominal value
- **common** (*Common*) – instance of the class Common

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

Function that computes other non-taxable income.

Parameters

- **p** (`Person`) – instance of the class Person
- **nom** (*function*) – function converting to nominal value

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

Function that computes contributions to registered and unregistered accounts (including DC and DB RPP).

Parameters

- **p** (`Person`) – instance of the class Person
- **year** (*int*) – year
- **common** (*Common*) – instance of the class Common

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

Function that resets all variables to their initial values.

Parameters **hh** (`Hhold`) – household

CHAPTER

FOUR

INITIALISATION

This module sets up households and attaches assets and debts to them. The table below shows all the individual inputs to the CPR (some of them are entered in other 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	

Though an example input file is provided as part of the package, a tutorial is also offered to show users how to modify those inputs to use the ones of their choice. NOTE: users must be careful when using their inputs, as a general error message will be generated if any of the – numerous – inputs is incorrectly specified (i.e. contains a mistake or is highly implausible).

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

Function that creates a household with one or two persons and attaches assets and debts.

Parameters

- **`index`** (*`int`*) – household index
- **`d_hh`** (*`dict`*) – dictionary of household features
- **`common`** (*`Common`*) – instance of the class Common
- **`prices`** (`Prices`) – instance of the class Prices

Returns Instance of the class Hhold.

Return type *Hhold*

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

This class creates a person.

Parameters

- **`d_hh`** (*`dict`*) – dictionary containing all informations about households
- **`l_sp`** (*`list`*) – list of features attached to spouses
- **`common`** (*`Common`*) – instance of the class Common
- **`prices`** (`Prices`) – instance of the class Prices
- **`s`** (*`bool`*) – False for head of household, True otherwise

`create_wage_profile`(*common*, *prices*)

Function that creates wage profiles for each realisation of uncertainty, to be used when the stochastic version of the tool is selected.

Parameters

- **`common`** (*`Common`*) – instance of the class Common
- **`prices`** (`Prices`) – instance of the class Prices

Returns Wage profiles.

Return type np.array

`create_shocks`(*T*, *N*, *prices*)

Function that creates a time series of shocks to the wage.

Parameters

- **`T`** (*`int`*) – length of the time series
- **`N`** (*`int`*) – number of realisations
- **`prices`** (`Prices`) – instance of the class Prices

Returns Time series of shocks to the wage.

Return type np.array

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

This class creates a household.

Parameters

- **`d_hh`** (*`dict`*) – dictionary containing all informations about households

- **l_hhold** (*list*) – list of features attached to the household
- **index** (*int*) – household index
- **common** (*Common*) – instance of the class Common
- **p0** (Person) – first spouse
- **p1** (Person) – second spouse (in the case of a couple)

set_other_years(*common*)

Function that sets years of partial and full retirement as well as years in which pre-retirement and post-retirement consumptions are assessed.

Parameters **common** (*Common*) – instance of the class Common

CHAPTER FIVE

MACRO

The Macro module contains classes that contain parameters common to all households, generate stochastic processes for returns and wages and other prices.

class `CPR.macro.CommonParameters`(*nsim*, *non_stochastic*, *extra_params*)
This class sets and contains the parameters common to all households.

Parameters

- **nsim** (*int*) – number of simulations
- **non_stochastic** (*bool*) – True if non stochastic simulation, False otherwise
- **extra_params** (*dict*) – dictionary of extra parameters

`set_limits`(*name*)
Set contributions limits for RRSP and TFSA.

Parameters **name** (*str*) – RRSP or TFSA

`prepare_ympe`()
" Pre-reform ympe used to adjust DB benefits for CPP

`prepare_cpp`()
Set percentages for cpp/qpp benefits.

class `CPR.macro.Prices`(*common*, *extra_params*)
This class computes the times series for asset returns, interest rates on debt, deterministic wage profiles, housing price growth rate and price/rent ratio.

Parameters

- **common** (*Common*) – instance of the class Common
- **extra_params** (*dict*) – dictionary of extra parameters

`simulate_ret`(*asset*, *common*)
Simulate N series of length T nominal returns distributed lognormally with autocorrelation rho.

Parameters

- **asset** (*str*) – type of asset
- **common** (*Common*) – instance of the class Common

Returns Array of nominal returns

Return type numpy.array

`compute_params_process`(*mu*, *rho*, *sigma*)
Convert arithmetic mean mu and volatility sigma of the returns and autocorrelation rho of the log returns into α, ρ and σ_ϵ of the process:

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

Parameters

- **mu** (*float*) – arithmetic mean
- **rho** (*float*) – autocorrelation :math:
- **sigma** (*float*) – standard deviation

Returns

- *float* – AR(1) coefficient (α)
- *float* – Standard deviation of error term (σ_ϵ)

simulate_housing(*common*)

Simulate series of nominal housing price growth (in $\ln(1 + r)$ form) and price-rent ratio.

Parameters **common** (*Common*) – instance of the class Common

Returns

- *numpy.array* – Array of nominal housing price growth
- *numpy.array* – Array of price-rent ratios

prepare_inflation_factors(*common*)

Compute inflation factors with base year 2018.

Parameters **common** (*Common*) – instance of the class Common

Returns Dictionary of inflation factors for each year

Return type dict

simulate_interest_debt()

Creates N series of yearly nominal interest rate of length T for each type of debt

Returns Dictionary of interest rates by type of debt and year

Return type dict

attach_diff_log_wages()

Creates a dictionary of differences in log wages by education and age.

Returns Dictionary of difference in log wages by education and age

Return type dict

initialize_factors()

This function creates an instance of life.table by gender and province.

Returns dictionary of annuity factors by gender and provinces

Return type dict

CHAPTER

SIX

ASSETS

This module manages all assets using various classes and functions: pension plans, individual savings (registered and unregistered), housing, business assets. It uses the contributions and room used as inputs as well as returns generated in the Macro module, and it updates all asset-related variables as the simulation progresses.

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

This class manages contribution room for TFSAs and RRSPs.

All amounts are nominal.

Parameters

- **init_room_rrsp** (*float*) – initial RRSP contribution room available
- **init_room_tfsa** (*float*) – initial TFSA contribution room available

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

Function to update contribution room for RRSPs and TFSAs, using the 2 other functions below (which themselves call the other functions of the class).

Parameters

- **p** (Person) – instance of the class Person
- **year** (*int*) – year
- **common** (*Common*) – instance of the class Common
- **prices** (Prices) – instance of the class Prices

update_rrsp_room(*p*, *year*, *common*)

Function to update RRSP contribution room.

Parameters

- **p** (Person) – instance of the class Person
- **year** (*int*) – year
- **common** (*Common*) – instance of the class Common

update_tfsa_room(*p*, *year*, *common*)

Function to update TFSA contribution room.

Parameters

- **p** (Person) – instance of the class Person
- **year** (*int*) – year
- **common** (*Common*) – instance of the class Common

`adjust_db_contributions`(*p*, *year*, *common*)
Function to adjust RRSP contribution room to DB RPP contributions.

Parameters

- **p** (`Person`) – instance of the class Person
- **year** (*int*) – year
- **common** (*Common*) – instance of the class Common

`adjust_dc_contributions`(*p*, *year*)
Function to adjust RRSP contribution room to DC RPP contributions.

If contribution room is insufficent for the intended/planned RRSP contributions, the "excess" contributions are channeled to TFSA.

Parameters

- **p** (`Person`) – instance of the class Person
- **year** (*int*) – year

`adjust_employees_contributions`(*p*)
Function to compute employee contributions to DC RPPs (later used to caculate taxes).

Parameters **p** (`Person`) – instance of the class Person

`adjust_rrsp_contributions`(*acc*, *p*, *year*)
Function to adjust RRSP contribution for contributions other than to RPPs.

If contribution room is insufficent for the intended/planned RRSP contributions, the "excess" contributions are channeled to TFSA.

Parameters

- **acc** (*str*) – type of account
- **p** (`Person`) – instance of the class Person
- **year** (*int*) – year

`adjust_tfsa_contributions`(*p*, *year*, *common*, *prices*)
Function to adjust TFSA contribution room to TFSA contributions (including "excess" DC RPP and RRSP contributions).

If contribution room is insufficent for the intended/planned TFSA contributions, the "excess" contributions are channeled to unregistered accounts.

Parameters

- **p** (`Person`) – instance of the class Person
- **year** (*int*) – year
- **common** (*Common*) – instance of the class Common
- **prices** (`Prices`) – instance of the class Prices

`adjust_rrif`(*p*, *year*, *common*, *prices*)
Function to adjust DC RPP and RRSP accounts to mandatory RRIF withdrawals (RRIFs are not separately modelled in this version).

Parameters

- **p** (`Person`) – instance of the class Person
- **year** (*int*) – year

- **common** (*Common*) – instance of the class Common
- **prices** (Prices) – instance of the class Prices

Returns Nominal amount that needs to be withdrawn.

Return type float

adjust_unreg_contributions(*p*, *year*)

Function to adjust contributions to unregistered accounts for "excess" TFSA contributions channeled to them.

Parameters

- **p** (Person) – instance of the class Person
- **year** (*int*) – year

reset()

Reset RRSP and TFSA contribution rooms to their inital values.

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

This class manages registered accounts. All amounts are nominal.

Parameters

- **p** (Person) – instance of the class Person
- **hh** (Hhold) – household
- **acc** (*str*) – type of account

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

Function to update the balance to account for contributions, withdrawals, and returns.

Parameters

- **d_returns** (*dict*) – dictionary of returns
- **year** (*int*) – year
- **common** (*Common*) – instance of the class Common
- **prices** (Prices) – instance of the class Prices

rate(*d_returns*, *year*)

Function to compute the rate of return given the mix of assets in the account.

Parameters

- **d_returns** (*dict*) – dictionary of returns
- **year** (*int*) – year

Returns Rate of return (net of fees).

Return type float

rrif_withdrawal(*p*, *common*)

Function to manage mandatory RRIF withdrawals.

Parameters

- **p** (Person) – instance of the class Person
- **common** (*Common*) – instance of the class Common

Returns Amount of mandatory withdrawal.

Return type float

`liquidate()`
Function to liquidate an account, setting balance, contributions and withdrawals to zero.

Returns Amount from liquidation (before taxes).

Return type float

`reset()`
Reset the balance and withdrawal to its initial balance.

class CPR.assets.**UnregAsset**(*p*, *hh*, *prices*)
This class manages unregistered accounts.

All amounts are nominal.

Parameters

- **p** (Person) – instance of the class Person
- **hh** (Hhold) – household
- **prices** (Prices) – instance of the class Prices

update(*d_returns*, *year*, *common*, *prices*)
Function to update the balance for contributions, withdrawals and returns.

Parameters

- **d_returns** (*dict of float*) – dictionary of returns
- **year** (*int*) – year
- **common** (*Common*) – instance of the class Common
- **prices** (Prices) – instance of the class Prices

compute_income(*d_returns*, *year*)
Function to compute new capital gains and taxable income (dividends and interests).

Parameters

- **d_returns** (*dict of float*) – dictionary of returns
- **year** (*int*) – year

rate(*d_returns*, *year*)
Function that computes the rate of return given the mix of assets in account.

Parameters

- **d_returns** (*dict of float*) – dictionary of returns
- **year** (*int*) – year

Returns Rate of return (net of fees).

Return type float

update_balance()
Function to update to balance and separate between non-taxable funds (i.e. previous post-tax balance), capital gains, dividends, and interests.

prepare_withdrawal()
Function to separate a withdrawal from the balance at the end of the period, and separately identify non-taxable funds, capital gains, dividends, and interests.

adjust_income()
Function to adjust investment income (dividends and interests) for withdrawals.

adjust_cap_losses(*realized_cap_gains*)
Function to compute capital losses from previous years used for deduction against capital gains, and adjust realized capital losses accordingly.

Parameters **realized_cap_gains** (*float*) – capital gains

Returns Capital losses (to be deducted).

Return type float

adjust_final_balance()
Function that adjusts the final balance for withdrawals, dividends and interests.

liquidate()
Function to liquidate the account and adjust capital losses.

reset()
Function to reset the balance, capital gains, and withdrawals to their initial values.

class CPR.assets.**RppDC**(*p*, *common*)
This class manages DC RPPs.

All amounts are nominal.

Parameters

- **p** (Person) – instance of the class Person
- **common** (*Common*) – instance of the class Common

class CPR.assets.**RppDB**(*p*)
This class manages DB RPPs.

All amounts are nominal.

Parameters **p** (Person) – instance of the class Person

compute_benefits(*p*, *common*)
Function that computes RPP DB benefits and adjusts them for CPP/QPP integration.

If RPP benefits are smaller than CPP/QPP benefits, RPP benefits are set to zero once the receipt of CPP/QPP retirement benefits begins.

Parameters

- **p** (Person) – instance of the class Person
- **common** (*Common*) – instance of the class Common

adjust_for_penalty(*p*, *common*)
Function to compute a penalty for individuals who begin to receive DB RPP benefits "early", i.e. before they accumulate the maximum number of years of service, if they are younger than the age at which benefits can start without penalty (the "early retirement age").

By default, the penalty applies to those who begin RPP benefits receipt before reaching 35 years of service and before age 62. These values can be modified.

Parameters

- **p** (Person) – instance of the class Person
- **common** (*Common*) – instance of the class Common

compute_cpp_adjustment(*p*, *common*)
Function to compute an adjustment (reduction) to DB RPP benefits to account for CPP/QPP integration.

Parameters

- **p** (Person) – instance of the class Person
- **common** (*Common*) – instance of the class Common

Returns Amount of benefit adjustment for CPP/QPP integration.

Return type float

reset()
Function that resets the benefits and contribution rates to their initial values.

class CPR.assets.**RealAsset**(*d_hh*, *resid*)
This class manages housing and residences.

All amounts are nominal.

Parameters

- **d_hh** (*dict of float*) – dictionary containing values of residences
- **resid** (*str*) – primary or secondary residence

update(*growth_rates*, *year*, *prices*)
Function to update the balance (residence values) for growth in price.

Parameters

- **growth_rates** (*dict*) – nominal return to housing (capital gains)
- **year** (*int*) – year
- **prices** (Prices) – instance of the class Prices

liquidate()
Function to liquidate the asset, compute capital gains if they apply, and set balance to zero.

Returns Amount from asset liquidation (before taxes).

Return type float

reset()
Function to reset residence value to its initial value and set capital gains to zero.

impute_rent(*hh*, *year*, *common*, *prices*)
Function to compute imputed rent.

Parameters

- **hh** (Hhold) – household
- **year** (*int*) – year
- **prices** (Prices) – instance of the class Prices

class CPR.assets.**Business**(*d_hh*)
This class manages a business (as an asset owned by the houshehold).

All amounts are nominal.

Parameters **d_hh** (*dict of float*) – dictionary containing value of business

update(*d_business_returns*, *year*, *prices*)
Function to update the balance and dividends.

Parameters

- `d_business_returns` (*dict*) – returns on business assets
- `year` (*int*) – year
- `prices` (`Prices`) – instance of the class Prices

`liquidate`(*common*)

Function to liquidate account (business assets), compute capital gains, and set balance to zero.

Parameters `common` (*Common*) – instance of the class Common

Returns Selling price of the business assets.

Return type float

`reset()`

Reset business value and capital gains to their initial values.

CHAPTER SEVEN

DEBTS

The Debts module manages debts (payments and balances) in the simulation.

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

This class manages all debts. All amounts are nominal.

Parameters

- **name** (*str*) – name of the debt
- **d_hh** (*dict*) – dictionary containing debt initial balances and monthly payments
- **common** (*Common*) – instance of the class Common
- **prices** (*Prices*) – instance of the class Prices

estimate_init_term(*common*, *prices*)

Estimate the term of the debt.

Parameters

- **common** (*Common*) – instance of the class Common
- **prices** (*Prices*) – instance of the class Prices

Returns term of the debt

Return type int

update(*year*, *rate*, *prices*)

Updates yearly payment and balance.

Parameters

- **year** (*int*) – year
- **rate** (*float*) – interest rate
- **prices** (*Prices*) – instance of the class Prices

reset()

Reset balance and term to initial values.

CHAPTER

EIGHT

TAXES

The Taxes module uses the SRD to compute taxes and benefits for each household.

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

Compute net nominal return and net amount from withdrawal for unregistered assets.

Parameters

- **`hh`** (`Hhold`) – household
- **`year`** (*`int`*) – year
- **`common`** (*`Common`*) – instance of the class Common
- **`prices`** (`Prices`) – instance of the class Prices

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

Files household using SRD.

Parameters

- **`hh`** (`Hhold`) – household
- **`year`** (*`int`*) – year
- **`common`** (*`Common`*) – instance of the class Common
- **`prices`** (`Prices`) – instance of the class Prices

Returns instance of class Hhold after tax

Return type *Hhold*

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

Files household with taxable income (interests and dividends) from unregistered assets.

Parameters

- **`hh`** (`Hhold`) – household
- **`year`** (*`int`*) – year
- **`common`** (*`Common`*) – instance of the class Common
- **`prices`** (`Prices`) – instance of the class Prices

Returns instance of class Hhold after tax

Return type *Hhold*

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

Computes nominal GIS, OAS and allowance benefits.

Parameters

- **hh** (Hhold) – household
- **hh_tax** (Hhold) – household
- **year** (*int*) – year
- **prices** (Prices) – instance of the class Prices

CHAPTER NINE

ANNUITIES

This module contains the functions needed to convert all financial assets into annuities upon retirement (of the first and/or the second spouse). One of the functions calls the annuity factor built using the Life module.

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

Function that partially converts assets into annuities when the first spouse retires, using the other functions below.

Parameters

- **`hh`** (`Hhold`) – household
- **`d_returns`** (*`dict`*) – dictionary of returns
- **`year`** (*`int`*) – year
- **`prices`** (`Prices`) – instance of the class Prices

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

Function that fully converts assets into annuities when the last spouse retires, using the other functions below.

Parameters

- **`hh`** (`Hhold`) – household
- **`d_returns`** (*`dict`*) – dictionary of returns
- **`year`** (*`int`*) – year
- **`prices`** (`Prices`) – instance of the class Prices

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

Function to liquidate financial assets.

Parameters **`p`** (`Person`) – instance of the class Person

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

Function to compute an individual factor for annuities constant in real terms.

We use an adjusted rate to dampen excess factor volatility. This is because we take the total return on bonds in the year that the annuity is purchased. Ideally, the whole interest structure should be used instead, such that mean reversion would smooth out annuity prices.

Parameters

- **`hh`** (`Hhold`) – household
- **`p`** (`Person`) – instance of the class Person
- **`rate`** (*`float`*) – interest rate
- **`prices`** (`Prices`) – instance of the class Prices

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

Function that converts assets to annuities in real terms.

Parameters

- **`p`** (`Person`) – instance of the class Person
- **`year`** (*`int`*) – year
- **`prices`** (`Prices`) – instance of the class Prices

CHAPTER

TEN

LIFE

The Life module contains the class that computes annuity factors for all individuals in the simulation, using mortality tables by province, gender as provided by Statistics Canada.

`class CPR.life.life.table`(*prov='qc'*, *scenario='M'*, *gender='males'*, *web=False*)
Class computing annuity factors by province, gender, age and birth year.

Parameters

- **prov** (*str*) – province; default=qc
- **scenario** (*str*) – longevity scenario; default=M (Medium mortality)
- **gender** (*str*) – gender; default=males
- **web** (*bool*) – True to pull data from internet (option unavailable at the moment), False otherwise; default=False

`pull_history`(*prov='qc'*, *gender='males'*)
Function to get historical mortalilty tables by province and gender.

Parameters

- **prov** (*str*) – province; default=qc
- **gender** (*str*) – gender; default=males

`splice()`
Function to add survival probabilities for the 2012 birth year and after.

`compute_annuity_factor`(*byear*, *agestart*, *rate*)
Function to compute an annuity factor.

Parameters

- **byear** (*int*) – birth year
- **agestart** (*int*) – age at which annuity starts
- **rate** (*float*) – interest rate

Returns Annuity factor.

Return type float

CHAPTER

ELEVEN

BALANCE SHEETS

This module holds functions that compute the various balance sheet items for each household at different points in the simulation process. A collection of variables are calculated in real terms (base year 2018) and added to the output dataframe retrieved by the module *main*.

At age 55 or in the year preceding retirement, if the latter occurs before age 56, wages, consumption, earlier pensions, account balances (RRSP, other registered, TFSA and unregistered accounts, as well as DC RPPs) are calculated. They are stored in variables ending with '_bef'.

At age 65 or in the year of retirement, if the latter occurs later, consumption; earlier pensions; annuities purchased with financial assets (balances); DB RPP, CPP/QPP, and OAS/GIS/Allowances benefits; as well as the values of residences and outstanding mortgages, are stored in variables ending with '_aft'.

For couples who do not retire at the same time, wages, earlier pensions, account balances and annuities of the first spouse to retire are calculated. They are stored in variables ending with '_part'. Variables describing the spouse who retires last begin with 's_'.

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

Function to compute the pre-retirement balance sheet.

Parameters

- **hh** (`Hhold`) – household
- **year** (*int*) – year
- **common** (*Common*) – instance of the class Common
- **prices** (`Prices`) – instance of the class Prices

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

Function to compute pre-retirement consumption.

Parameters

- **hh** (`Hhold`) – household
- **year** (*int*) – year
- **prices** (`Prices`) – instance of the class Prices

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

Function to compute the post-retirement balance sheet.

Parameters

- **hh** (`Hhold`) – household
- **year** (*int*) – year
- **common** (*Common*) – instance of the class Common

- **prices** (Prices) – instance of the class Prices

CPR.balance_sheets.**add_output**(*hh*, *year*, *prices*, *key*)
Function to extract output variables.

Parameters

- **hh** (Hhold) – household
- **year** (*int*) – year
- **prices** (Prices) – instance of the class Prices
- **key** (*str*) – before ("bef"), when first spouse retires ("part") or after retirement ("aft")

CHAPTER

TWELVE

ANALYSIS

In this module, the function *get_dataset* retrieves a synthetic (fake) dataset. The class *Results* offers a few functions to summarize and merge the data, as well as assess preparedness for retirement.

`CPR.analysis.`**`get_dataset`**`()`

Function that returns a dataframe of synthetic data.

Returns Dataframe of synthetic data.

Return type pandas.core.frame.DataFrame

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

This class prepares the results.

Parameters

- **input_data** (*pandas.core.frame.DataFrame*) – dataframe of inputs
- **output** (*pandas.core.frame.DataFrame*) – dataframe of outputs
- **common** (*Common*) – instance of the class Common
- **prices** (Prices) – instance of the class Prices
- **extra_params** (*dict*) – dictionary of extra parameters

summarize()

Function to summarize the simulation.

merge(*add_index=True*)

Function to merge input and output variables to create a database.

Parameters **add_index** (*bool, optional*) – add index to database, True by default

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

Function that introduces a consumption floor, computes RRI (NaN if consumption before and after retirement below cons_floor), and checks preparedness for retirement.

Parameters

- **factor_couple** (*int, optional*) – factor to normalize income for couple, by default 2
- **cons_floor** (*int, optional*) – consumption floor, by default 100
- **d_cutoffs** (*dict, optional*) – cutoffs, by default {20: 80, 100: 65}

CHAPTER

THIRTEEN

TOOLS

In this module, several functions, used in the rest of the code, are available to create dictionaries, modify parameters, or convert prices, among other aspects.

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

Function to create a dictionary from a .csv file.

Parameters

- **file** (*_io.TextIOWrapper*) – csv file
- **numerical_key** (*bool, optional*) – numerical key, by default False

Returns Dictionary of parameters.

Return type dict

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

Function to add parameters to a class instance.

Parameters

- **inst** (*object*) – class instance
- **file** (*_io.TextIOWrapper*) – csv file

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

Function to update the value of some parameters in inst with values from *extra_params*.

Parameters

- **inst** (*dict*) – parameters
- **extra_params** (*dict*) – parameters to be updated and their new values

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

Creation of a function to convert prices from real to nominal and from nominal to real, with base year 2018.

Parameters

- **year** (*int*) – year
- **prices** (Prices) – instance of the class Prices

Returns

- *function* – Function converting nominal to real.
- *function* – Function converting real to nominal.

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

Creation of a function that converts prices from real to nominal, with base year 2018. Used for some specific purposes elsewhere in the code.

Parameters

- **year** (*int*) – year
- **prices** (Prices) – instance of the class Prices

Returns Function converting real to nominal.

Return type function

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

Creation of a function that converts prices from nominal to real, with base year 2018.

Parameters

- **year** (*int*) – year
- **prices** (Prices) – instance of the class Prices

Returns function converting nominal to real

Return type function

CHAPTER

FOURTEEN

TUTORIALS

14.1 Basic use of CPR

14.2 Experiments with CPR

Click here to access the notebook.

CHAPTER FIFTEEN

CONTRIBUTORS AND RIGHTS OF USE

15.1 Contributors

David Boisclair, Ismaël Choinière-Crèvecœur, Pierre-Carl Michaud, Pierre-Yves Yanni

15.2 Contact

Retirement and Savings Institute

15.3 Licence

The CPR is provided under the MIT licence (“MIT License”). The conditions of the licence are as follows:

CHAPTER

SIXTEEN

PDF DOCUMENTATION

`pdf`

PYTHON MODULE INDEX

INDEX