RBO Toolbox package for Matlab and R

September 27, 2012

Title: RBO Toolbox

Version: 0.1.1

Author: Allister Bernard, Jose-Maria Fernandez, Lloyd Han, Andrew Lo, James Noraky and Roger Stein

Description: Equations and parameters used by the RBO toolbox to simulate the investments in a portfolio of new drugs being developed in preclinical and clinical phases by a biomedical megafund and financed with a capital structured composed of debt and equity.

The simulation code is coded in both Matlab and R. The structure of this document focuses first, for each section, on the matlab version of code and presents the equivalent function in R at the end of each section.

This code supports the results presented in the paper "Commercializing biomedical research through securitization techniques" (Fernandez, Stein, Lo, Nature Biotechnology, 2012). More information of the assumptions and parameters used in this code can be found in the text and supplementary information of that paper.

Topics documented:

MAIN FUNCTIONS:

Init_params_fn

Simulate_CFS_fn

HELPER FUNCTIONS:

Determine_current_state_fn

Trial_cost_fn

Compound_sale_price_fn

Calc_amort_pmt_due_fn

Summarize_results_fn

Plot_standard_diagnostics_fn

OTHER:

Name2index_fn

Phase2index

Lognrnd

MakeVectorLabels

Quantile

Init_params _fn

Description:

This function initializes all parameters used by the simulation code and hence, the function contains all default values used in the code.

More information on the estimation of these parameters is included on the Supplementary Information of the paper "Commercializing biomedical research through securitization techniques" (Fernandez, Stein, Lo, Nature Biotechnology 2012)

Params is of type struct and has 3 fields simu (simulation parameters) assets (asset parameters), bonds (bond parameters)

Usage:

params = init_params_fn()

Inputs/fields:

Rho	Correlation factor of the values of the compounds
trans.prob	Transition probabilities to shift from state i into state j over the next period of time t (t is equal to 6 months in the simulations presented in the paper cited above)
States of nature	DSC (discontinued), PRE (preclinical), P1 (phase 1), P2 (phase 2), P3 (phase 3), NDA (New Drug Application or Biologic License Application), APP (approved drug)
pricing. params	Includes the following parameters (each of which is calculated for each state of nature) • Vmu = mean of the log normal distribution (adjusted by the cap) used to calculate the value of each compound • Sigma = standard deviation of the log normal distribution (adjusted by the cap) used to calculate the value of each compound • Vmx = maximum valuation of the compound per phase (cap) • UpFront = amount paid upfront to the owner of the compound in exchange for ownership of the equity and the obligation to pay milestones and development costs as the compound progresses along its development stages • MileStone = incentive paid whenever the compound transitions to the next phase

	• Mu = mean of the log normal distribution (adjusted by the cap) used to calculate the development cost of each compound in each phase • Sigma = standard deviation of the lognormal distribution (adjusted by the cap) used to calculate the development cost of each compound in each phase • Max = maximum investment in development costs per compound per phase (cap) • FutureCostEst = estimate of the future expected development costs per compound per phase. It is used to calculate the cash that will be saved to pay, in future periods of time, the clinical trials
sell.in.phase	Indicates the phase in which compounds are sold. It is the target end phase for the simulation running.
sale.time	Time it takes to cash in the proceeds of the sale of a compound in the units of time defined for the simulation (semester)
equity.stake	share of the value of the compound that is acquired by the Megfund un exchange for the upfront, milestone and clinical development costs financed
capital.structure	share of senior debt (A1), junior debt (A2) and equity tranches (EQ) that conform the capital structure of the Megafund
interest.coverage	Indicates the number of periods for which the interest coverage ratio has to be observed per capital structure tranche. The first column refers to the senior debt (A!), the second to the junior debt (A2) and the third to the Equity (EQ) tranche
IC.pers	Number of periods for which expected future interest and principal payments are reserved
amort.timing	Indicates the periods (start and end period) in which the capital structure tranches are amortized in equal instalments. The first two columns refer to the senior debt start and end redemption periods and the second two columns refer to the start and end redemption date for the junior debt.
coupon	Value of the semi annual coupon payment. The first column refers to the senior debt (A!), the second to the junior debt (A2) and the third to the Equity (EQ) tranche which coupon

	is always zero
servicing.rate	Semi-annual fee paid to the megafund managers for servicing the fund operations
svc.accrual.int.rate	In case a megafund liquidation occurs, it is the rate at which the unpaid service fees are accrued
cash.accrual.int.rate	Rate at which cash surplus is invested
NSIMUS	Number of simulations run
TIMESTEPS=	max(b.params$amort.timing)+1 Number of time steps in the life of the fund, defined as the tenor of the bonds plus an additional period of time in which the equity tranche will be liquidated
initial.cash	Initial cash raised by the Megafund
initial.compounds	Number of compounds targeted in the initial portfolio classified by state. Each of the columns in the vector refer to each alternative state of nature in the following order: DIS, PRE, P1, P2, P3, NDA, APP The final number of compounds invested in may be equal or smaller than this number. This depends on the amount of capital raised

Equivalent function in R:

Init.params.fn

Simulate_CFs_fn

Description:

For each simulation round this function resets its variables and then simulates the evolution of the compounds and all the cash flows generated by the transition of the assets purchased and financed.

This is the main function that simulates all the cash flows generated by the transition of the compounds purchased and financed. It starts by defining all the variables of the model and it also defines eight nested functions that use these variables

It includes a group of nested functions and the routine required to simulate the evolution and results of the Research Based Obligations (RBO) Megafund.

Usage:

```
results = simulate_CFs_fn(show_progress,debugging, params)
```

Input parameters:

- show_progress is of type boolean and if true displays a status bar of the current simulation
- debugging is of type unsigned integer and can be any number from 0-4 where as the number increases the amount of debugging information printed increases
- params is of type struct and is parameter object which is the output of the init_params_fn function (optional)

Output:

- results is of type struct and provides details of each simulation run

Nested functions:

1. calc_svc_due_this_per_ffn

Description:

Calculates the service payments due in this period

Usage:

```
svc_due_this_per = calc_svc_due_this_per_ffn(values)
```

Arguments:

```
svc_due_this_per = params.bonds.servicing_rate.*sum(values)
```

- params.bonds.servicing_rate = it is the servicing cost defined in init_params (0.0025)

- sum(values)= sum of total debt and equity

If there are unpaid service fees from earlier periods, then they are paid if there is enough cash

svc_past_due = unpaid_svc*(1+params.bonds.svc_accrual_int_rate)

- svc.accrual.int.rate = rate at which unpaid service accrues in time. It is defined in the init_params function

And consequently

svc_due = svc_due_this_per + svc_past_due;

If there is enough cash, then proceed to pay the servicing

If not, then calculate the unpaid service

unpaid_svc = round((svc_due-svc_paid)*10^5)/10^5;

And Update the cash available:

current_cash = current_cash-svc_paid;

Equivalent function in R:

Calc.svc.due.this.per.ffn

Pay.servicing.ffn

2. ic_test_result_ffn

Description:

A function that calculates the value of the IC test ratio

Usage:

IC_ratio = ic_test_result_ffn(b, i, current_cash)

Arguments:

- b = bond values
- i = time periods

To calculate the numerator of the ratio:

We define K as the current cash minus the bond amortization payments during i+m periods minus the bond coupon payments as well over the next i+m periods and minus the service rate for the current period

K = K-AMORT_SCHED(bb,i+m)-params.bonds.coupon(bb)*btemp-params.bonds.servicing_rate*btemp

Where

- m goes from 1 to N and N = min(params.bonds.IC_pers,NPERS-i) where IC.pers is a number of periods defined in the init_params_fn
- bb refers to all the debt in the capital structure
- btemp represents the value of the outstanding bonds

Next is the calculation of the denominator:

cash_due = sched_P + sched_I + sched_svc

where

- sched_P = sched_P + AMORT_SCHED(b,i+m) are the scheduled amortizations of the corresponding bond (the senior or the junior depending on whether we are calculating the cash due for A1 or A2) over the next i+m periods of time
- sched_I = sched_I + params.bonds.coupon(b)*bv are the scheduled coupon payments of the corresponding bond (the senior or the junior depending on whether we are calculating the cash due for A1 or A2)over the next i+m periods of time
- Sched_svc = sched_svc+params.bonds.servicing_rate*bv are the scheduled bond servicing payments of bonds over the next i+m periods of time

Lastly, we calculate the ratio:

IC_ratio = K/cash_due

Equivalent function in R:

Ic.test.result.ffn

3. liquidate_portfolio_ffn

Description:

It is a function that processes the liquidation of compounds when a megafund reaches liquidation. It applies to the compounds that are neither sold nor discontinued at the end of the simulation

Usage:

liquidate_portfolio_ffn(cur_per)

The value at which compounds are liquidated grows as the compound progresses in its development and also depending on whether or not the compound clinical trials have been funded for the current phase and on whether or not the compound is still in its initial phase.

Compounds that are “invested” are compounds that have already been developed in their current phase (the clinical trials and other development costs have already been paid for) and consequently their value is higher than that of other compounds that are on the same phase but have not received funding.

For compounds that are “invested”, i.e if after the compound transitioned to the current state there was enough cash available to fund their clinical trials

Price = compound_sale_price_fn(mu, sigma, mx, rho, z)
*equity_in_compound

For compounds that are not invested, that is to say if the clinical trials corresponding to the current state were not financed by the megafund because it did not have enough liquidity reserved:

price = max(price-invest_cost(m),0)

For compounds that might not transitioned to any new state different from the original state they were in since the start of the simulation, the price of the compound would be:

price = price*0.05

Inputs:

- mx, mu, sigma, rho and z are valuation parameters described in the init_params function and in the compounds_sale_price function
- equity in compound as described in init_params

Output:

- Values compounds when they are sold and updates the cash available and compound states

Equivalent function in R:

Liquidate.portfolio.ffn

4. **Sell_compounds_to_cover_shortfall_ffn =**

Description:

It is a function that simulates the sale of a compound triggered by a positive shortfall.

The function computes its value when IC test triggers sales of compounds to cover the shortfall.

The most developed compounds are sold first among those compounds still alive (neither sold nor discontinued). Compounds are sold until the shortfall is covered (until shortfall is smaller than “totall”).

Usage:

sell_compounds_to_cover_shortfall_ffn(shortfall,cur_per)

Inputs:

- mx, mu, sigma, rho and z are valuation parameters described in the init_params function and in the compounds_sale_price function

- equity in compound as described in init_params
- totall is the value of compounds sold to cover the shortfall
- shortfall is explained later in this document. It can be calculated for each of the debt tranches as well we for the equity and it is the difference between the amount that needs to be paid times the IC test factor minus the current and future cash available over the next 2 periods of time.

Output:

- cash generated by the sale of compounds required to cover the shortfall

Equivalent function in R:

Sell.compounds.to.cover.shortfall.ffn

5. sell_compounds_ffn

Description:

A function that simulates the sale of a compound and computes its value

Usage:

```
sell_compound_ffn(comp_idx, cur_per)
```

The sale value of compounds grows as the compound progresses in its development. Another factor impacting the sale value is whether or not the compound has been funded ("invested") in its current phase:.

For compounds that are "invested", i.e if after the compound transitioned to the current state there was enough cash available to fund their clinical trials

```
Price = compound_sale_price_fn(mu, sigma, mx, rho, z)
*equity_in_compound
```

For compounds that are not invested, that is to say if the clinical trials corresponding to the current state were not financed by the megafund because it did not have enough liquidity reserved:

```
price = max(price-invest_cost(m),0)
```

Inputs:

- mx, mu, sigma, rho and z are valuation parameters described in the init_params function and in the compounds_sale_price function
- equity in compound as described in init_params
- invest_cost refers to the cost to develop the compound in this phase

Output:

- sale value of compounds and change in case after the sale

Equivalent function in R:

Sell.compound.ffn

6. withdraw_compounds_ffn

Description:

A function that updates compounds which have been withdrawn

Usage:

```
withdraw_compound_ffn(old_phase,cur_per)
```

Inputs:

- old_phase and cur_per

Output:

- update compounds that have been withdrawn

Equivalent function in R:

Withdraw.compound.ffn

7. transition_compound_ffn

Description:

A function that updates compounds when they transition and then estimates investment cost per compound. Compounds require an investment to shift from each phase into the next one except for if they are discontinued or sold or if they are in NDA phase where we have assumed that the investment required to transition to the next phase is 0 for these simulations.

Usage:

```
transition.compound.ffn (idx)
```

Inputs:

- idx is of type integer and maps the string phase to a corresponding index
- milestone from init_params
- trial_cost_fn(compounds{idx}, params) that is the cost financed to pay for the clinical development of the compound in the current phase

Output:

- NDA compounds can transition without a need to be funded
- Compounds in other phases (except for those in absorption states –SLD, DIS-) need to incur on an investment cost that is updated

using this function.

Equivalent function in R:

Transition.compound.ffn

8. Check_for_transitions_ffn

Description:

It is a function that checks for transitions of compounds. It applies to compounds that are not discontinued (DSC), approved (APP) or sold (SLD)

Usage:

next_phase = check_for_transitions_ffn(compounds_j, invested_j, j, rho, params, debugging)

Inputs:

- P is a random number drawn from a normal distribution

Output:

- In what phase is each compound

Equivalent function in R:

check.for.transitions.ffn

RBO simulation routine:

This routine is the same in R and matlab

Here below is the list of routines, instructions and variables presented as calculated in each simulation:

- Start by setting up the global variables and constants to perform the simulation and keep record of it
- Every simulation starts by calculating the number of compounds that the fund acquires. In the first period of life of each fund simulation the following procedures occur:
    - Calculate cash_left_to_spend: of all cash available we deduct the interest payments of the first IC.pers periods of time to avoid running out of cash to pay interest in early periods of life of the fund
    - Calculate number of compounds we actually buy. Price_temp indicates how much money we spend in each one of them
    - Save enough funding per compound to address future expected development costs per compound. The variable FutureCostEst in init params is used to calculate how much must be expected per compound invested.

- At the start of the second period of the simulation the following routines take place:
    - First the current state of the portfolio if determined. To do so, compounds are checked to confirm if they have transitioned or not to a different state:
        - Compounds that are eligible_for_transition (have been funded or are in NDA) are check_for_transitions_ffn
        - If compounds transition to the sell_in_phase (target phase of the drug development simulation) then they are sold and the proceeds are paid into the fund (sell_compound_ffn) according to the time defined in sale_time vector
        - If compounds shift to the discontinue phase, withdraw_compounds_ffn applies
    - Next calculate how much cash will be required in this period to attend the fund financial obligations.
        - If there is enough cash to pay service, interest and principal redemptions then the fund is not in default and the simulation continues
        - If there is not enough cash then the debt is in default and the portfolio is liquidated. The proceeds of the liquidation are then used to pay servicing and then as much interest and principal from the senior tranche (A1) and from the junior debt tranche (A2) and then whatever cash, if any, is left goes to the equity holders
    - If not in default, then service, interest and principal are paid
    - Next the interest coverage (IC) is tested:
        - Calculate the IC test for the senior debt and the senior bond shortfall. Calculate the service coverage for the equity tranche
        - If the IC is too low (and the shortfall is bigger than 0) then compounds are sold to cover the shortfall
    - Next if IC test is passed and there is enough cash, compounds that have transitioned to a new phase in this period are funded (their clinical trials are paid for so they can be able to transition in a future period of the simulation)
        - As an additional mechanism to protect investors, the cash_to_invest is calculated. This is the amount of cash left after subtracting from the service, principal and interest payments over the next IC.pers periods of time
        - If there is enough cash_to_invest, the invest_cost is paid (clinical trial costs and phase milestones)
- At the end of period cash is updated and then the cash at the start of the next period is calculated and the simulation shifts to the next period of time
- Once all the periods of the simulation are completed, then the portfolio remaining is liquidated and the residual cash flows to the equity holders and their return on equity is calculated.

HELPER FUNCTIONS:

Determine_current_state_fn

Description:

This function determines whether a compound transitions or not to a new phase.

Usage:

```
value = determine_current_state_fn(p, dist, dist_col)
```

Input parameters:

- p is of type double and is a random draw from the uniform distribution across the open interval (0,1)
- dist is a vector of type double and is row from the transition probability matrix params.assets.trans_prob
- dist_col is a struct defined by params.assets.trans_prob_col

Output:

- value is of type String and maps the selected element in dist to its corresponding name from dist_col
- A random number is selected and checked against the vector of cumulative transition probabilities to determine in what state the compound is at the start of each period

Equivalent function in R:

determine.current.state.fn

Trial_cost_fn

Description:

Determines the cost of funding trials (stochastically) for phase and compound

Usage:

```
cost = trial_cost_fn(comp_phase, params)
```

Input:

- comp.phase is of type string and is one of 'DSC','PRE','P1','P2','P3','NDA', 'APP', 'SLD'
- params is a struct containing the initial set of parameters set by the function init_params_fn

Output:

- cost is of type double and is the returned funding cost

Equivalent function in R:

trial.cost.fn

Compound_sale_price_fn

Description:

This function calculates the value of compounds to be sold.

Usage:

```
value = compound_sale_price_fn(mu, sigma, mx, rho, z1)
```

Input parameters:

- mu is of type double and specifies the mean of a normal distribution
- sigma is of type double and specified the standard deviation of the normal distribution
- rho is of type double and acts as a control parameter which if 0 returns draws from the corresponding lognormal distribution else it returns draws from a biased log normal
- z1 is of type double and specifies the amount of bias if rho is non-zero
- mx is of type double and specifies the maximum value beyond which draws from the lognorma distribution are rejected

Output:

- value is of type double and contains the resulting draw of the log-normal distribution
- The calculation of the value of any compound sold is explained in more detail in the Supplementary Information of the paper "Commercializing biomedical research through securitization techniques" (Fernandez, Stein, Lo, Nature Biotechnology, 2012)

Equivalent function in R:

- compound.sale.price.fn

Calc_amort_pmt_due_fn

Description:

Calculates the principal payment to be made every period to redeem the debt issued by the fund. It is applied to the senior debt issued (A1) and the junior debt tranche too (A2)

Usage:

```
pmt = calc_amort_pmt_due_fn(per, a_start, a_end, orig_par, cur_par)
```

Arguments:

Input:

- per is an integer representing the payment period

- a.start, and a.end are integers representing period of start (end) of amortization of the bodns issued
- orig.par and @cur.par are doubles representing par amount (nominal outstanding amount of the bond) at issuance of the bonds and their outstanding amount

Output:

- pmt is a double representing the calculated principal payment due in period per
- pmt is calculated by dividing the orig_par by the number of periods in which the debt is amortized (amort_per)
- amort_per is equal to a_end-a_start+1

Equivalent function in R:

Calc.amort.pmt.due.fn

Summarize_results_fn

Description:

This function displays a summary of relevant results of the simulation. The summary is provided in both numeric and graphical formats

Usage:

summarize_results_fn (results,plot)

Input parameters:

- results is of type struct and is a simulation results summary object which is the resultant output of the simulate_CFs_fn function
- plot is of type boolean and if true displays a graphical output of some summary statistics

Output:

- Asset analysis: target portfolio by phase and compounds initially purchased, compounds exiting per phase and per period and compounds funded in each period (funds needed and invested)
- Equity and bond analysis: return distribution for equity holders and probabilities of default and expected losses for debt holders

Equivalent function in R:

Summarize.results.fn

Plot_standard_diagnostics_fn

Description:

This function generates plots of the simulation and is called within the function summarize_results_fn.

Usage:

plot_standard_diagnostics_fn(results)

Input:

- results is of type struct and is a simulation results summary object which is the resultant output of the simulate_CFs_fn function

Output:

- presents a summary of results from the simulation run including cash at end of each period, ROE annualized, balance of senior A1 and junior A2 bonds outstanding at the end of each period, compounds sold and discontinued per phase and period of time.

Equivalent function in R:

- plot.standard.diagnostics

OTHER FUNCTIONS:

Lognrnd

Description:

This function generates random deviates from a lognormal distribution

Usage:

r = lognrnd(mu,sigma, varargin)

Input parameters:

- mu is the mean of the normal distribution
- sigma is the standard deviation of the normal distribution

Output:

- r is a random draw from a log-normal distribution

MakeVectorLabels

Description:

This function is used to provide row and column labels for a matrix in the form of a structure array. The values of the structure array fields serve as indexes into the row/column of the corresponding matrix.

Usage:

labelstruct = makeVectorLabels(lab, addctr)

Input parameters:

- lab a cell array of names
- addctr a flag to indicate whether to append a count at the end of each name

Output:

- labelstruct a struct whose fields have names from @lab and value is a running count from 1..length(lab)

Name2index

Description:

This function assumes every matrix of name say 'mat' within a structure parentstruct has fields mat_col and mat_row which provides a mapping from name to row/col index. The mapping can be done using the function makeVectorLabels.

Usage:

idx = name2index_fn(parentstruct, fieldname, rowcolname, isrow)

Input parameters:

- parentstruct is of type struct
- fieldname is of type string and parentstruct.(fieldname) should exist
- rowcolname is of type string and is name of a row/column for the matrix parentstruct.(fieldname)
- isrow is boolean and is used to decide whether we search for the index having name 'rowcolname' in the structure parentstruct.(fieldname_row) or parentstruct.(fieldname_col)

Output:

- idx is of type int and provides a mmaping from string (rowcolname) to integer (idx)

Phase2index

Description:

This function converts character string representation of phase into an index

Usage:

idx = phase2index_fn(phase)

Input parameters:

- phase is of type string

Output:

- idx is of type integer and maps the string phase to a corresponding index

---

Quantile

Description:

A simple function that calculates quantiles of an input vector

Usage:

qs = quantile(vec, percs)

Input parameters:

- vec is a numeric vector
- percs is the percentile(s) we want to calculate

Output:

- qs is the quantile(s) of vector @vec at percentile @perc