

BondProc - zbiór procedur i funkcji związanych z obligacjami

Zygmunt Zawadzki

June 20, 2013

1 Co to?

BondProc to pakiet mojego autorstwa, w którym jest kilka funkcji związanych z obligacjami.

2 Dlaczego? I po co?

Dlaczego? Bo mogę. Po co? Nie wiem. Generalnie są tutaj różne funkcje które może komuś się przydadzą w najbliższej pracy zaliczeniowej. Jednak szczerze wątpię, by na obecnym stadium pakiet był jakoś bardzo użyteczny - gdyż nie ma dokumentacji prawie wcale - a bez tego ani rusz (a nie mam teraz czasu by móc nad tym przysiedzieć). Nie ma też wielu funkcji które powinny być, by móc wygodnie korzystać z tegoż pakietu, a te które są (a mimo wszystko jest ich kilka) oczywiście praktycznie bezużyteczne bez dokumentacji... Jednak może ktoś w tej krótkiej notce zobaczy coś fajnego, co go zmobilizuje do roboty... No!

3 Instalacja

Instalacja jest prosta - wystarczy odpalić poniższy kod. To wszystko.

```
# Tutaj można kod zobaczyć:  
# https://github.com/zzawadz/BondProc/tree/master/R  
if (require(lpSolve)) install.packages("lpSolve")  
if (require(devtools)) install.packages("devtools")  
devtools::install_github("BondProc", "zzawadz")  
require(BondProc)
```

4 Obligacja, koszyk obligacji

Obligacja jest najniższym elementem w hierarchii. Tworzy się ją tak:

```
# Wczytanie pakietu
library(BondProc)
# Stworzenie obligacji
bond = newBond(face = 100, maturity = 10, coupon = 0.1, n.pay = 2)
```

Obliczyć cenę obligacji można na trzy sposoby:

```
# Pojedyncza stopa w całym okresie
getBondPrice(bond, r = 0.1)

## [1] 98.43

# Stopa jako funkcja czasu - pakiet sam obliczy potrzebne stopy do
# dyskonta
rf <- function(t) (t * 2)/100
getBondPrice(bond, rf)

## [1] 71.17

# Stopa podana jako model N-S, również tu pakiet sam liczy co trzeba
ns = newNelsonSiegel(alpha1 = 0.06, 0.03, 0.02, 1)
getBondPrice(bond, ns)

## [1] 123.2
```

Na wyższym poziomie stoi `bondBasket` - jest to po prostu zbiór obligacji. `bondBasket` tworzy się niezmiernie prosto. I ma fikuśne własności (a przynajmniej powinien mieć, bo jeszcze nie byłem w stanie zakodzić wszystkiego). Między innymi, można jedną funkcją dostać wektor z cenami dla wszystkich elementów z koszyka.

```
basket = newBondBasket()

for (i in 1:10) {
  # dodawanie obligacji do basketu wystarczy zwykły plus
  bond = newBond(face = 100, maturity = i, coupon = 0.1, n.pay = 2)
  basket = basket + bond
}

# Parametry NS jak wyżej
ns = newNelsonSiegel(alpha1 = 0.06, 0.03, 0.02, 1)

getBondPrice(basket, ns)

## [1] 101.3 103.5 106.1 108.8 111.5 114.1 116.6 118.9 121.1 123.2
```

Ale to nie wszystko - ważniejsze jest to, co jest w następnej sekcji - czyli gotowa funkcja do której wrzuca się `basket` i dostaje się wagi zimmunizowane według M^2 .

5 Immunizacja przy pomocy M^2

Jedno z zadań do kolokwium. W weights są wagi.

```
require(BondProc)

# Obligacje:
bond1 = newBond(face = 100, maturity = 12, coupon = 0.1, n.pay = 1)
bond2 = newBond(face = 100, maturity = 9, coupon = 0.08, n.pay = 1)
bond3 = newBond(face = 100, maturity = 5, coupon = 0.07, n.pay = 1)

# By stworzyc koszyk mozna tez same obligacje dodac
basket = bond1 + bond2 + bond3

# Struktura stóp terminowych podana jako funckja czasu
rf = function(t) (0.5 * t + 2.5)/100

# Immunizacja
portfolio = immunizationMsqr(basket = basket, r = rf, H = 7)

## Loading required package: lpSolve
print(portfolio)

## $weights
## [1] 0.3665 0.6335 0.0000
##
## $portfolio.Msqr
## [1] 11.41
```