Package 'FinancialMath'

January 20, 2025

Type Package

Title Financial Mathematics for Actuaries

Version 0.1.1

Author Kameron Penn [aut, cre],
Jack Schmidt [aut]

Maintainer Kameron Penn <kameron.penn.financialmath@gmail.com>

Description Contains financial math functions and introductory derivative functions included in the Society of Actuaries and Casualty Actuarial Society 'Financial Mathematics' exam, and some topics in the 'Models for Financial Economics' exam.

License GPL-2

Encoding UTF-8

LazyData true

NeedsCompilation no

Repository CRAN

Date/Publication 2016-12-16 22:51:34

Contents

`amort.period` *Amortization Period*

Description

Solves for either the number of payments, the payment amount, or the amount of a loan. The payment amount, interest paid, principal paid, and balance of the loan are given for a specified period.

Usage

```
amort.period(Loan=NA,n=NA,pmt=NA,i,ic=1,pf=1,t=1)
```

Arguments

`Loan`	loan amount
`n`	the number of payments/periods
`pmt`	value of level payments
`i`	nominal interest rate convertible ic times per year
`ic`	interest conversion frequency per year

pf	the payment frequency- number of payments per year
t	the specified period for which the payment amount, interest paid, principal paid, and loan balance are solved for

Details

Effective Rate of Interest: $eff.i = (1 + \frac{i}{ic})^{ic} - 1$

$j = (1 + eff.i)^{\frac{1}{pf}} - 1$

$Loan = pmt * a_{\overline{n}|j}$

Balance at the end of period t: $B_t = pmt * a_{\overline{n-t}|j}$

Interest paid at the end of period t: $i_t = B_{t-1} * j$

Principal paid at the end of period t: $p_t = pmt - i_t$

Value

Returns a matrix of input variables, calculated unknown variables, and amortization figures for the given period.

Note

Assumes that payments are made at the end of each period.

One of n, pmt, or Loan must be NA (unknown).

If pmt is less than the amount of interest accumulated in the first period, then the function will stop because the loan will never be paid off due to the payments being too small.

If the pmt is greater than the loan amount plus interest accumulated in the first period, then the function will stop because one payment will pay off the loan.

t cannot be greater than n.

Author(s)

Kameron Penn and Jack Schmidt

See Also

amort.table

Examples

```
amort.period(Loan=100,n=5,i=.01,t=3)

amort.period(n=5,pmt=30,i=.01,t=3,pf=12)

amort.period(Loan=100,pmt=24,ic=1,i=.01,t=3)
```

amort.table *Amortization Table*

Description

Produces an amortization table for paying off a loan while also solving for either the number of payments, loan amount, or the payment amount. In the amortization table the payment amount, interest paid, principal paid, and balance of the loan are given for each period. If n ends up not being a whole number, outputs for the balloon payment, drop payment and last regular payment are provided. The total interest paid, and total amount paid is also given. It can also plot the percentage of each payment toward interest vs. period.

Usage

```
amort.table(Loan=NA,n=NA,pmt=NA,i,ic=1,pf=1,plot=FALSE)
```

Arguments

`Loan`	loan amount
`n`	the number of payments/periods
`pmt`	value of level payments
`i`	nominal interest rate convertible ic times per year
`ic`	interest conversion frequency per year
`pf`	the payment frequency- number of payments per year
`plot`	tells whether or not to plot the percentage of each payment toward interest vs. period

Details

Effective Rate of Interest: $eff.i = (1 + \frac{i}{ic})^{ic} - 1$

$j = (1 + eff.i)^{\frac{1}{pf}} - 1$

$Loan = pmt * a_{\overline{n}|j}$

Balance at the end of period t: $B_t = pmt * a_{\overline{n-t}|j}$

Interest paid at the end of period t: $i_t = B_{t-1} * j$

Principal paid at the end of period t: $p_t = pmt - i_t$

Total Paid= $pmt * n$

Total Interest Paid= $pmt * n - Loan$

If $n = n^* + k$ where n^* is an integer and $0 < k < 1$:

Last regular payment (at period n^*) $= pmt * s_{\overline{k}|j}$

Drop payment (at period $n^* + 1$) $= Loan * (1 + j)^{n^*+1} - pmt * s_{\overline{n^*}|j}$

Balloon payment (at period n^*) $= Loan * (1 + j)^{n^*} - pmt * s_{\overline{n^*}|j} + pmt$

Value

A list of two components.

`Schedule`	A data frame of the amortization schedule.
`Other`	A matrix of the input variables and other calculated variables.

Note

Assumes that payments are made at the end of each period.

One of n, Loan, or pmt must be NA (unknown).

If pmt is less than the amount of interest accumulated in the first period, then the function will stop because the loan will never be paid off due to the payments being too small.

If pmt is greater than the loan amount plus interest accumulated in the first period, then the function will stop because one payment will pay off the loan.

Author(s)

Kameron Penn and Jack Schmidt

See Also

`amort.period`

`annuity.level`

Examples

```
amort.table(Loan=1000,n=2,i=.005,ic=1,pf=1)

amort.table(Loan=100,pmt=40,i=.02,ic=2,pf=2,plot=FALSE)

amort.table(Loan=NA,pmt=102.77,n=10,i=.005,plot=TRUE)
```

`annuity.arith` *Arithmetic Annuity*

Description

Solves for the present value, future value, number of payments/periods, amount of the first payment, the payment increment amount per period, and/or the interest rate for an arithmetically growing annuity. It can also plot a time diagram of the payments.

Usage

```
annuity.arith(pv=NA,fv=NA,n=NA,p=NA,q=NA,i=NA,ic=1,pf=1,imm=TRUE,plot=FALSE)
```

Arguments

`pv`	present value of the annuity
`fv`	future value of the annuity
`n`	number of payments/periods
`p`	amount of the first payment
`q`	payment increment amount per period
`i`	nominal interest frequency convertible ic times per year
`ic`	interest conversion frequency per year
`pf`	the payment frequency- number of payments per year
`imm`	option for annuity immediate or annuity due, default is immediate (TRUE)
`plot`	option to display a time diagram of the payments

Details

Effective Rate of Interest: $eff.i = (1 + \frac{i}{ic})^{ic} - 1$

$j = (1 + eff.i)^{\frac{1}{pf}} - 1$

$fv = pv * (1 + j)^n$

Annuity Immediate:

$pv = p * a_{\overline{n|}j} + q * \frac{a_{\overline{n|}j} - n*(1+j)^{-n}}{j}$

Annuity Due:

$pv = (p * a_{\overline{n|}j} + q * \frac{a_{\overline{n|}j} - n*(1+j)^{-n}}{j}) * (1 + i)$

Value

Returns a matrix of the input variables, and calculated unknown variables.

Note

At least one of pv, fv, n, p, q, or i must be NA (unknown).

pv and fv cannot both be specified, at least one must be NA (unknown).

Author(s)

Kameron Penn and Jack Schmidt

See Also

`annuity.geo`

`annuity.level`

`perpetuity.arith`

`perpetuity.geo`

`perpetuity.level`

Examples

```
annuity.arith(pv=NA,fv=NA,n=20,p=100,q=4,i=.03,ic=1,pf=2,imm=TRUE)

annuity.arith(pv=NA,fv=3000,n=20,p=100,q=NA,i=.05,ic=3,pf=2,imm=FALSE)
```

`annuity.geo` *Geometric Annuity*

Description

Solves for the present value, future value, number of payments/periods, amount of the first payment, the payment growth rate, and/or the interest rate for a geometrically growing annuity. It can also plot a time diagram of the payments.

Usage

```
annuity.geo(pv=NA,fv=NA,n=NA,p=NA,k=NA,i=NA,ic=1,pf=1,imm=TRUE,plot=FALSE)
```

Arguments

`pv`	present value of the annuity
`fv`	future value of the annuity
`n`	number of payments/periods for the annuity
`p`	amount of the first payment
`k`	payment growth rate per period
`i`	nominal interest rate convertible ic times per year
`ic`	interest conversion frequency per year
`pf`	the payment frequency- number of payments/periods per year
`imm`	option for annuity immediate or annuity due, default is immediate (TRUE)
`plot`	option to display a time diagram of the payments

Details

Effective Rate of Interest: $eff.i = (1 + \frac{i}{ic})^{ic} - 1$

$j = (1 + eff.i)^{\frac{1}{pf}} - 1$

$fv = pv * (1 + j)^n$

Annuity Immediate:

j != k: $pv = p * \frac{1-(\frac{1+k}{1+j})^n}{j-k}$

j = k: $pv = p * \frac{n}{1+j}$

Annuity Due:

j != k: $pv = p * \frac{1-(\frac{1+k}{1+j})^n}{j-k} * (1 + j)$

j = k: $pv = p * n$

Value

Returns a matrix of the input variables and calculated unknown variables.

Note

At least one of pv, fv, n, pmt, k, or i must be NA (unknown).

pv and fv cannot both be specified, at least one must be NA (unknown).

See Also

annuity.arith

annuity.level

perpetuity.arith

perpetuity.geo

perpetuity.level

Examples

```
annuity.geo(pv=NA,fv=100,n=10,p=9,k=.02,i=NA,ic=2,pf=.5,plot=TRUE)

annuity.geo(pv=NA,fv=128,n=5,p=NA,k=.04,i=.03,pf=2)
```

---

annuity.level *Level Annuity*

Description

Solves for the present value, future value, number of payments/periods, interest rate, and/or the amount of the payments for a level annuity. It can also plot a time diagram of the payments.

Usage

```
annuity.level(pv=NA,fv=NA,n=NA,pmt=NA,i=NA,ic=1,pf=1,imm=TRUE,plot=FALSE)
```

Arguments

pv	present value of the annuity
fv	future value of the annuity
n	number of payments/periods
pmt	value of the level payments
i	nominal interest rate convertible ic times per year
ic	interest conversion frequency per year
pf	the payment frequency- number of payments/periods per year
imm	option for annuity immediate or annuity due, default is immediate (TRUE)
plot	option to display a time diagram of the payments

Details

Effective Rate of Interest: $eff.i = (1 + \frac{i}{ic})^{ic} - 1$

$j = (1 + eff.i)^{\frac{1}{pf}} - 1$

Annuity Immediate:

$pv = pmt * a_{\overline{n}|j} = pmt * \frac{1-(1+j)^{-n}}{j}$

$fv = pmt * s_{\overline{n}|j} = pmt * a_{\overline{n}|j} * (1 + j)^n$

Annuity Due:

$pv = pmt * \ddot{a}_{\overline{n}|j} = pmt * a_{\overline{n}|j} * (1 + j)$

$fv = pmt * \ddot{s}_{\overline{n}|j} = pmt * a_{\overline{n}|j} * (1 + j)^{n+1}$

Value

Returns a matrix of the input variables and calculated unknown variables.

Note

At least one of pv, fv, n, pmt, or i must be NA (unknown).

pv and fv cannot both be specified, at least one must be NA (unknown).

See Also

annuity.arith

annuity.geo

perpetuity.arith

perpetuity.geo

perpetuity.level

Examples

```
annuity.level(pv=NA,fv=101.85,n=10,pmt=8,i=NA,ic=1,pf=1,imm=TRUE)

annuity.level(pv=80,fv=NA,n=15,pf=2,pmt=NA,i=.01,imm=FALSE)
```

`bear.call` *Bear Call Spread*

Description

Gives a table and graphical representation of the payoff and profit of a bear call spread for a range of future stock prices.

Usage

```
bear.call(S,K1,K2,r,t,price1,price2,plot=FALSE)
```

Arguments

`S`	spot price at time 0
`K1`	strike price of the short call
`K2`	strike price of the long call
`r`	yearly continuously compounded risk free rate
`t`	time of expiration (in years)
`price1`	price of the short call with strike price K1
`price2`	price of the long call with strike price K2
`plot`	tells whether or not to plot the payoff and profit

Details

Stock price at time t $= S_t$

For $S_t <= K1$: payoff $= 0$

For $K1 < S_t < K2$: payoff $= K1 - S_t$

For $S_t >= K2$: payoff $= K1 - K2$

payoff = profit + (price1 - price2)$*e^{r*t}$

Value

A list of two components.

`Payoff`	A data frame of different payoffs and profits for given stock prices.
`Premiums`	A matrix of the premiums for the call options and the net cost.

Note

K1 must be less than S, and K2 must be greater than S.

Author(s)

Kameron Penn and Jack Schmidt

See Also

`bear.call.bls`

`bull.call`

`option.call`

Examples

```
bear.call(S=100,K1=70,K2=130,r=.03,t=1,price1=20,price2=10,plot=TRUE)
```

bear.call.bls — *Bear Call Spread - Black Scholes*

Description

Gives a table and graphical representation of the payoff and profit of a bear call spread for a range of future stock prices. Uses the Black Scholes equation for the call prices.

Usage

```
bear.call.bls(S,K1,K2,r,t,sd,plot=FALSE)
```

Arguments

S	spot price at time 0
K1	strike price of the short call
K2	strike price of the long call
r	yearly continuously compounded risk free rate
t	time of expiration (in years)
sd	standard deviation of the stock (volatility)
plot	tells whether or not to plot the payoff and profit

Details

Stock price at time t $= S_t$

For $S_t <= K1$: payoff $= 0$

For $K1 < S_t < K2$: payoff $= K1 - S_t$

For $S_t >= K2$: payoff $= K1 - K2$

payoff = profit$+(price_{K1} - price_{K2}) * e^{r*t}$

Value

A list of two components.

Payoff	A data frame of different payoffs and profits for given stock prices.
Premiums	A matrix of the premiums for the call options and the net cost.

Note

K1 must be less than S, and K2 must be greater than S.

Author(s)

Kameron Penn and Jack Schmidt

See Also

bear.call

bull.call.bls

option.call

Examples

```
bear.call.bls(S=100,K1=70,K2=130,r=.03,t=1,sd=.2)
```

`bls.order1` *Black Scholes First-order Greeks*

Description

Gives the price and first order greeks for call and put options in the Black Scholes equation.

Usage

```
bls.order1(S,K,r,t,sd,D=0)
```

Arguments

S	spot price at time 0
K	strike price
r	continuously compounded yearly risk free rate
t	time of expiration (in years)
sd	standard deviation of the stock (volatility)
D	continuous dividend yield

Value

A matrix of the calculated greeks and prices for call and put options.

Note

Cannot have any inputs as vectors.

t cannot be negative.

Either both or neither of S and K must be negative.

Author(s)

Kameron Penn and Jack Schmidt

See Also

option.put

option.call

Examples

```
x <- bls.order1(S=100, K=110, r=.05, t=1, sd=.1, D=0)
ThetaPut <- x["Theta","Put"]
DeltaCall <- x[2,1]
```

---

bond *Bond Analysis*

Description

Solves for the price, premium/discount, and Durations and Convexities (in terms of periods). At a specified period (t), it solves for the full and clean prices, and the write up/down amount. Also has the option to plot the convexity of the bond.

Usage

```
bond(f,r,c,n,i,ic=1,cf=1,t=NA,plot=FALSE)
```

Arguments

f	face value
r	coupon rate convertible cf times per year
c	redemption value
n	the number of coupons/periods for the bond
i	nominal interest rate convertible ic times per year
ic	interest conversion frequency per year
cf	coupon frequency- number of coupons per year
t	specified period for which the price and write up/down amount is solved for, if not NA
plot	tells whether or not to plot the convexity

Details

Effective Rate of Interest: $eff.i = (1 + \frac{i}{ic})^{ic} - 1$

$j = (1 + eff.i)^{\frac{1}{cf}} - 1$

coupon $= \frac{f*r}{cf}$ (per period)

price = coupon$*a_{\overline{n}|j} + c*(1+j)^{-n}$

$MACD = \frac{\sum_{k=1}^{n} k*(1+j)^{-k}*coupon+n*(1+j)^{-n}*c}{price}$

$MODD = \frac{\sum_{k=1}^{n} k*(1+j)^{-(k+1)}*coupon+n*(1+j)^{-(n+1)}*c}{price}$

$MACC = \frac{\sum_{k=1}^{n} k^2*(1+j)^{-k}*coupon+n^2*(1+j)^{-n}*c}{price}$

$MODC = \frac{\sum_{k=1}^{n} k*(k+1)*(1+j)^{-(k+2)}*coupon+n*(n+1)*(1+j)^{-(n+2)}*c}{price}$

Price (for period t):

If t is an integer: price =coupon$*a_{\overline{n-t}|j} + c*(1+j)^{-(n-t)}$

If t is not an integer then $t = t^* + k$ where t^* is an integer and $0 < k < 1$:

full price $=$ (coupon$*a_{\overline{n-t^*}|j} + c*(1+j)^{-(n-t^*)}) * (1+j)^k$

clean price = full price$-k*$coupon

If price > c :

premium = price$-c$

Write-down amount (for period t) $=$ (coupon$-c*j) * (1+j)^{-(n-t+1)}$

If price < c :

discount $= c-$price

Write-up amount (for period t) $= (c*j-$coupon$) * (1+j)^{-(n-t+1)}$

Value

A matrix of all of the bond details and calculated variables.

Note

t must be less than n.

To make the duration in terms of years, divide it by cf.

To make the convexity in terms of years, divide it by cf^2.

Examples

```
bond(f=100,r=.04,c=100,n=20,i=.04,ic=1,cf=1,t=1)

bond(f=100,r=.05,c=110,n=10,i=.06,ic=1,cf=2,t=5)
```

bull.call — *Bull Call Spread*

Description

Gives a table and graphical representation of the payoff and profit of a bull call spread for a range of future stock prices.

Usage

```
bull.call(S,K1,K2,r,t,price1,price2,plot=FALSE)
```

Arguments

`S`	spot price at time 0
`K1`	strike price of the long call
`K2`	strike price of the short call
`r`	yearly continuously compounded risk free rate
`t`	time of expiration (in years)
`price1`	price of the long call with strike price K1
`price2`	price of the short call with strike price K2
`plot`	tells whether or not to plot the payoff and profit

Details

Stock price at time t $= S_t$

For $S_t <= K1$: payoff $= 0$

For $K1 < S_t < K2$: payoff $= S_t - K1$

For $S_t >= K2$: payoff $= K2 - K1$

profit = payoff + (price2 - price1)$* e^{r*t}$

Value

A list of two components.

`Payoff`	A data frame of different payoffs and profits for given stock prices.
`Premiums`	A matrix of the premiums for the call options and the net cost.

Note

K1 must be less than S, and K2 must be greater than S.

See Also

bull.call.bls

bear.call

option.call

Examples

```
bull.call(S=115,K1=100,K2=145,r=.03,t=1,price1=20,price2=10,plot=TRUE)
```

---

bull.call.bls *Bull Call Spread - Black Scholes*

Description

Gives a table and graphical representation of the payoff and profit of a bull call spread for a range of future stock prices. Uses the Black Scholes equation for the call prices.

Usage

```
bull.call.bls(S,K1,K2,r,t,sd,plot=FALSE)
```

Arguments

S	spot price at time 0
K1	strike price of the long call
K2	strike price of the short call
r	yearly continuously compounded risk free rate
t	time of expiration (in years)
sd	standard deviation of the stock (volatility)
plot	tells whether or not to plot the payoff and profit

Details

Stock price at time t $= S_t$

For $S_t <= K1$: payoff $= 0$

For $K1 < S_t < K2$: payoff $= S_t - K1$

For $S_t >= K2$: payoff $= K2 - K1$

profit = payoff$+(price_{K2} - price_{K1}) * e^{r*t}$

Value

A list of two components.

Payoff	A data frame of different payoffs and profits for given stock prices.
Premiums	A matrix of the premiums for the call options and the net cost.

Note

K1 must be less than S, and K2 must be greater than S.

See Also

bear.call

option.call

Examples

```
bull.call.bls(S=115,K1=100,K2=145,r=.03,t=1,sd=.2)
```

---

butterfly.spread   *Butterfly Spread*

Description

Gives a table and graphical representation of the payoff and profit of a long butterfly spread for a range of future stock prices.

Usage

```
butterfly.spread(S,K1,K2=S,K3,r,t,price1,price2,price3,plot=FALSE)
```

Arguments

S	spot price at time 0
K1	strike price of the first long call
K2	strike price of the two short calls
K3	strike price of the second long call
r	continuously compounded yearly risk free rate
t	time of expiration (in years)
price1	price of the long call with strike price K1
price2	price of one of the short calls with strike price K2
price3	price of the long call with strike price K3
plot	tells whether or not to plot the payoff and profit

Details

Stock price at time t $= S_t$

For $S_t <= K1$: payoff $= 0$

For $K1 < S_t <= K2$: payoff $= S_t - K1$

For $K2 < S_t < K3$: payoff $= 2 * K2 - K1 - S_t$

For $S_t >= K3$: payoff $= 0$

profit = payoff$+$(2$*$price2 - price1 - price3) $* e^{r*t}$

Value

A list of two components.

`Payoff`	A data frame of different payoffs and profits for given stock prices.
`Premiums`	A matrix of the premiums for the call options and the net cost.

Note

K2 must be equal to S.

K3 and K1 must both be equidistant to K2 and S.

K1 < K2 < K3 must be true.

See Also

`butterfly.spread.bls`

`option.call`

Examples

```
butterfly.spread(S=100,K1=75,K2=100,K3=125,r=.03,t=1,price1=25,price2=10,price3=5)
```

`butterfly.spread.bls` *Butterfly Spread - Black Scholes*

Description

Gives a table and graphical representation of the payoff and profit of a long butterfly spread for a range of future stock prices. Uses the Black Scholes equation for the call prices.

Usage

```
butterfly.spread.bls(S,K1,K2=S,K3,r,t,sd,plot=FALSE)
```

Arguments

`S`	spot price at time 0
`K1`	strike price of the first long call
`K2`	strike price of the two short calls
`K3`	strike price of the second long call
`r`	continuously compounded yearly risk free rate
`t`	time of expiration (in years)
`sd`	standard deviation of the stock (volatility)
`plot`	tells whether or not to plot the payoff and profit

Details

Stock price at time t = S_t

For $S_t <= K1$: payoff = 0

For $K1 < S_t <= K2$: payoff = $S_t - K1$

For $K2 < S_t < K3$: payoff = $2 * K2 - K1 - S_t$

For $S_t >= K3$: payoff = 0

profit = payoff$+(2 * price_{K2} - price_{K1} - price_{K3}) * e^{r*t}$

Value

A list of two components.

`Payoff`	A data frame of different payoffs and profits for given stock prices.
`Premiums`	A matrix of the premiums for the call options and the net cost.

Note

K2 must be equal to S.

K3 and K1 must both be equidistant to K2 and S.

K1 < K2 < K3 must be true.

See Also

`butterfly.spread`

`option.call`

Examples

```
butterfly.spread.bls(S=100,K1=75,K2=100,K3=125,r=.03,t=1,sd=.2)
```

`cf.analysis` *Cash Flow Analysis*

Description

Calculates the present value, macaulay duration and convexity, and modified duration and convexity for given cash flows. It also plots the convexity and time diagram of the cash flows.

Usage

```
cf.analysis(cf,times,i,plot=FALSE,time.d=FALSE)
```

Arguments

cf	vector of cash flows
times	vector of the periods for each cash flow
i	interest rate per period
plot	tells whether or not to plot the convexity
time.d	tells whether or not to plot the time diagram of the cash flows

Details

$$pv = \sum_{k=1}^{n} \frac{cf_k}{(1+i)^{times_k}}$$

$$MACD = \frac{\sum_{k=1}^{n} times_k*(1+i)^{-times_k}*cf_k}{pv}$$

$$MODD = \frac{\sum_{k=1}^{n} times_k*(1+i)^{-(times_k+1)}*cf_k}{pv}$$

$$MACC = \frac{\sum_{k=1}^{n} {times_k}^2*(1+i)^{-times_k}*cf_k}{pv}$$

$$MODC = \frac{\sum_{k=1}^{n} times_k*(times_k+1)*(1+i)^{-(times_k+2)}*cf_k}{pv}$$

Value

A matrix of all of the calculated values.

Note

The periods in t must be positive integers.

See Also

TVM

Examples

```
cf.analysis(cf=c(1,1,101),times=c(1,2,3),i=.04,time.d=TRUE)

cf.analysis(cf=c(5,1,5,45,5),times=c(5,4,6,7,5),i=.06,plot=TRUE)
```

---

collar *Collar Strategy*

Description

Gives a table and graphical representation of the payoff and profit of a collar strategy for a range of future stock prices.

Usage

```
collar(S,K1,K2,r,t,price1,price2,plot=FALSE)
```

Arguments

`S`	spot price at time 0
`K1`	strike price of the long put
`K2`	strike price of the short call
`r`	yearly continuously compounded risk free rate
`t`	time of expiration (in years)
`price1`	price of the long put with strike price K1
`price2`	price of the short call with strike price K2
`plot`	tells whether or not to plot the payoff and profit

Details

Stock price at time t $= S_t$

For $S_t <= K1$: payoff $= K1 - S_t$

For $K1 < S_t < K2$: payoff $= 0$

For $S_t >= K2$: payoff $= K2 - S_t$

profit = payoff + (price2 - price1)$*e^{r*t}$

Value

A list of two components.

`Payoff`	A data frame of different payoffs and profits for given stock prices.
`Premiums`	A matrix of the premiums for the call and put options and the net cost.

See Also

`collar.bls`

`option.put`

`option.call`

Examples

```
collar(S=100,K1=90,K2=110,r=.05,t=1,price1=5,price2=15,plot=TRUE)
```

`collar.bls` *Collar Strategy - Black Scholes*

Description

Gives a table and graphical representation of the payoff and profit of a collar strategy for a range of future stock prices. Uses the Black Scholes equation for the call and put prices.

Usage

```
collar.bls(S,K1,K2,r,t,sd,plot=FALSE)
```

Arguments

`S`	spot price at time 0
`K1`	strike price of the long put
`K2`	strike price of the short call
`r`	yearly continuously compounded risk free rate
`t`	time of expiration (in years)
`sd`	standard deviation of the stock (volatility)
`plot`	tells whether or not to plot the payoff and profit

Details

Stock price at time t $= S_t$

For $S_t <= K1$: payoff $= K1 - S_t$

For $K1 < S_t < K2$: payoff $= 0$

For $S_t >= K2$: payoff $= K2 - S_t$

profit = payoff$+(price_{K2} - price_{K1}) * e^{r*t}$

Value

A list of two components.

`Payoff`	A data frame of different payoffs and profits for given stock prices.
`Premiums`	A matrix of the premiums for the call and put options and the net cost.

See Also

`option.put`

`option.call`

Examples

```
collar.bls(S=100,K1=90,K2=110,r=.05,t=1,sd=.2)
```

`covered.call` *Covered Call*

Description

Gives a table and graphical representation of the payoff and profit of a covered call strategy for a range of future stock prices.

Usage

```
covered.call(S,K,r,t,sd,price=NA,plot=FALSE)
```

Arguments

`S`	spot price at time 0
`K`	strike price
`r`	continuously compounded yearly risk free rate
`t`	time of expiration (in years)
`sd`	standard deviation of the stock (volatility)
`price`	specified call price if the Black Scholes pricing is not desired (leave as NA to use the Black Scholes pricing)
`plot`	tells whether or not to plot the payoff and profit

Details

Stock price at time t $= S_t$

For $S_t <= K$: payoff $= S_t$

For $S_t > K$: payoff $= K$

profit = payoff + price$*e^{r*t} - S$

Value

A list of two components.

`Payoff`	A data frame of different payoffs and profits for given stock prices.
`Premium`	The price of the call option.

Note

Finds the put price by using the Black Scholes equation by default.

See Also

`option.call`

`covered.put`

Examples

```
covered.call(S=100,K=110,r=.03,t=1,sd=.2,plot=TRUE)
```

---

covered.put *Covered Put*

Description

Gives a table and graphical representation of the payoff and profit of a covered put strategy for a range of future stock prices.

Usage

```
covered.put(S,K,r,t,sd,price=NA,plot=FALSE)
```

Arguments

S	spot price at time 0
K	strike price
r	continuously compounded yearly risk free rate
t	time of expiration (in years)
sd	standard deviation of the stock (volatility)
price	specified put price if the Black Scholes pricing is not desired (leave as NA to use the Black Scholes pricing)
plot	tells whether or not to plot the payoff and profit

Details

Stock price at time t $= S_t$

For $S_t <= K$: payoff $= S - K$

For $S_t > K$: payoff $= S - S_t$

profit = payoff + price$*e^{r*t}$

Value

A list of two components.

Payoff	A data frame of different payoffs and profits for given stock prices.
Premium	The price of the put option.

Note

Finds the put price by using the Black Scholes equation by default.

See Also

option.put

covered.call

Examples

```
covered.put(S=100,K=110,r=.03,t=1,sd=.2,plot=TRUE)
```

---

forward *Forward Contract*

Description

Gives a table and graphical representation of the payoff of a forward contract, and calculates the forward price for the contract.

Usage

```
forward(S,t,r,position,div.structure="none",dividend=NA,df=1,D=NA,k=NA,plot=FALSE)
```

Arguments

S	spot price at time 0
t	time of expiration (in years)
r	continuously compounded yearly risk free rate
position	either buyer or seller of the contract ("long" or "short")
div.structure	the structure of the dividends for the underlying ("none", "continuous", or "discrete")
dividend	amount of each dividend, or amount of first dividend if k is not NA
df	dividend frequency- number of dividends per year
D	continuous dividend yield
k	dividend growth rate per df
plot	tells whether or not to plot the payoff

Details

Stock price at time t $= S_t$

Long Position: payoff = S_t - forward price

Short Position: payoff = forward price - S_t

If div.structure = "none"

forward price$= S * e^{r*t}$

If div.structure = "discrete"

$eff.i = e^r - 1$

$j = (1 + eff.i)^{\frac{1}{df}} - 1$

Number of dividends: $t^* = t * df$

if k = NA: forward price $= S * e^{r*t}$−dividend$*s_{\overline{t^*}|j}$

if k != j: forward price $= S * e^{r*t}$−dividend$*\frac{1-(\frac{1+k}{1+j})^{t^*}}{j-k} * e^{r*t}$

if k = j: forward price $= S * e^{r*t}$−dividend$*\frac{t^*}{1+j} * e^{r*t}$

If div.structure = "continuous"

forward price= $S * e^{(r-D)*t}$

Value

A list of two components.

`Payoff`	A data frame of different payoffs for given stock prices.
`Price`	The forward price of the contract.

Note

Leave an input variable as NA if it is not needed (ie. k=NA if div.structure="none").

See Also

forward.prepaid

Examples

```
forward(S=100,t=2,r=.03,position="short",div.structure="none")

forward(S=100,t=2,r=.03,position="long",div.structure="discrete",dividend=3,k=.02)

forward(S=100,t=1,r=.03,position="long",div.structure="continuous",D=.01)
```

`forward.prepaid` *Prepaid Forward Contract*

Description

Gives a table and graphical representation of the payoff of a prepaid forward contract, and calculates the prepaid forward price for the contract.

Usage

```
forward.prepaid(S,t,r,position,div.structure="none",dividend=NA,df=1,D=NA,
k=NA,plot=FALSE)
```

Arguments

`S`	spot price at time 0
`t`	time of expiration (in years)
`r`	continuously compounded yearly risk free rate
`position`	either buyer or seller of the contract ("long" or "short")
`div.structure`	the structure of the dividends for the underlying ("none", "continuous", or "discrete")
`dividend`	amount of each dividend, or amount of first dividend if k is not NA
`df`	dividend frequency- number of dividends per year
`D`	continuous dividend yield
`k`	dividend growth rate per df
`plot`	tells whether or not to plot the payoff

Details

Stock price at time t $= S_t$

Long Position: payoff = S_t - prepaid forward price

Short Position: payoff = prepaid forward price - S_t

If div.structure = "none"

forward price$= S$

If div.structure = "discrete"

$eff.i = e^r - 1$

$j = (1 + eff.i)^{\frac{1}{df}} - 1$

Number of dividends: $t^* = t * df$

if k = NA: prepaid forward price $= S-$dividend$*a_{\overline{t^*}|j}$

if k != j: prepaid forward price $= S-$dividend$*\frac{1-(\frac{1+k}{1+j})^{t^*}}{j-k}$

if k = j: prepaid forward price $= S-$dividend$*\frac{t^*}{1+j}$

If div.structure = "continuous"

prepaid forward price$= S * e^{-D*t}$

Value

A list of two components.

`Payoff`	A data frame of different payoffs for given stock prices.
`Price`	The prepaid forward price of the contract.

Note

Leave an input variable as NA if it is not needed (ie. k=NA if div.structure="none").

See Also

forward

Examples

```
forward.prepaid(S=100,t=2,r=.04,position="short",div.structure="none")

forward.prepaid(S=100,t=2,r=.03,position="long",div.structure="discrete",
dividend=3,k=.02,df=2)

forward.prepaid(S=100,t=1,r=.05,position="long",div.structure="continuous",D=.06)
```

IRR *Internal Rate of Return*

Description

Calculates internal rate of return for a series of cash flows, and provides a time diagram of the cash flows.

Usage

```
IRR(cf0,cf,times,plot=FALSE)
```

Arguments

cf0	cash flow at period 0
cf	vector of cash flows
times	vector of the times for each cash flow
plot	option whether or not to provide the time diagram

Details

$cf0 = \sum_{k=1}^{n} \frac{cf_k}{(1+irr)^{times_k}}$

Value

The internal rate of return.

Note

Periods in t must be positive integers.

Uses polyroot function to solve equation given by series of cash flows, meaning that in the case of having a negative IRR, multiple answers may be returned.

Author(s)

Kameron Penn and Jack Schmidt

See Also

NPV

Examples

```
IRR(cf0=1,cf=c(1,2,1),times=c(1,3,4))

IRR(cf0=100,cf=c(1,1,30,40,50,1),times=c(1,1,3,4,5,6))
```

NPV *Net Present Value*

Description

Calculates the net present value for a series of cash flows, and provides a time diagram of the cash flows.

Usage

```
NPV(cf0,cf,times,i,plot=FALSE)
```

Arguments

`cf0`	cash flow at period 0
`cf`	vector of cash flows
`times`	vector of the times for each cash flow
`i`	interest rate per period
`plot`	tells whether or not to plot the time diagram of the cash flows

Details

$NPV = cf0 - \sum_{k=1}^{n} \frac{cf_k}{(1+i)^{times_k}}$

Value

The NPV.

Note

The periods in t must be positive integers.

The lengths of cf and t must be equal.

See Also

IRR

Examples

```
NPV(cf0=100,cf=c(50,40),times=c(3,5),i=.01)

NPV(cf0=100,cf=50,times=3,i=.05)

NPV(cf0=100,cf=c(50,60,10,20),times=c(1,5,9,9),i=.045)
```

---

option.call *Call Option*

Description

Gives a table and graphical representation of the payoff and profit of a long or short call option for a range of future stock prices.

Usage

```
option.call(S,K,r,t,sd,price=NA,position,plot=FALSE)
```

Arguments

S	spot price at time 0
K	strike price
r	continuously compounded yearly risk free rate
t	time of expiration (in years)
sd	standard deviation of the stock (volatility)
price	specified call price if the Black Scholes pricing is not desired (leave as NA to use the Black Scholes pricing)
position	either buyer or seller of option ("long" or "short")
plot	tells whether or not to plot the payoff and profit

Details

Stock price at time t $= S_t$

Long Position:

payoff = max$(0, S_t - K)$

profit = payoff - price$*e^{r*t}$

Short Position:

payoff = -max$(0, S_t - K)$

profit = payoff + price$*e^{r*t}$

Value

A list of two components.

`Payoff`	A data frame of different payoffs and profits for given stock prices.
`Premium`	The price for the call option.

Note

Finds the call price by using the Black Scholes equation by default.

Author(s)

Kameron Penn and Jack Schmidt

See Also

`option.put`

`bls.order1`

Examples

```
option.call(S=100,K=110,r=.03,t=1.5,sd=.2,price=NA,position="short")

option.call(S=100,K=100,r=.03,t=1,sd=.2,price=10,position="long")
```

`option.put` *Put Option*

Description

Gives a table and graphical representation of the payoff and profit of a long or short put option for a range of future stock prices.

Usage

```
option.put(S,K,r,t,sd,price=NA,position,plot=FALSE)
```

Arguments

`S`	spot price at time 0
`K`	strike price
`r`	continuously compounded yearly risk free rate
`t`	time of expiration (in years)
`sd`	standard deviation of the stock (volatility)
`price`	specified put price if the Black Scholes pricing is not desired (leave as NA to use the Black Scholes pricing)
`position`	either buyer or seller of option ("long" or "short")
`plot`	tells whether or not to plot the payoff and profit

Details

Stock price at time t $= S_t$

Long Position:

payoff = $\max(0, K - S_t)$

profit = payoff$-price * e^{r*t}$

Short Position:

payoff = -$\max(0, K - S_t)$

profit = payoff$+price * e^{r*t}$

Value

A list of two components.

`Payoff`	A data frame of different payoffs and profits for given stock prices.
`Premium`	The price of the put option.

Note

Finds the put price by using the Black Scholes equation by default.

Author(s)

Kameron Penn and Jack Schmidt

See Also

`option.call`

`bls.order1`

Examples

```
option.put(S=100,K=110,r=.03,t=1,sd=.2,price=NA,position="short")

option.put(S=100,K=110,r=.03,t=1,sd=.2,price=NA,position="long")
```

`perpetuity.arith` *Arithmetic Perpetuity*

Description

Solves for the present value, amount of the first payment, the payment increment amount per period, or the interest rate for an arithmetically growing perpetuity.

Usage

```
perpetuity.arith(pv=NA,p=NA,q=NA,i=NA,ic=1,pf=1,imm=TRUE)
```

Arguments

pv	present value of the annuity
p	amount of the first payment
q	payment increment amount per period
i	nominal interest rate convertible ic times per year
ic	interest conversion frequency per year
pf	the payment frequency- number of payments per year
imm	option for annuity immediate or annuity due, default is immediate (TRUE)

Details

Effective Rate of Interest: $eff.i = (1 + \frac{i}{ic})^{ic} - 1$

$j = (1 + eff.i)^{\frac{1}{pf}} - 1$

Perpetuity Immediate:

$pv = \frac{p}{j} + \frac{q}{j^2}$

Perpetuity Due:

$pv = (\frac{p}{j} + \frac{q}{j^2}) * (1 + j)$

Value

Returns a matrix of input variables, and calculated unknown variables.

Note

One of pv, p, q, or i must be NA (unknown).

Author(s)

Kameron Penn and Jack Schmidt

See Also

perpetuity.geo

perpetuity.level

annuity.arith

annuity.geo

annuity.level

Examples

```
perpetuity.arith(100,p=1,q=.5,i=NA,ic=1,pf=1,imm=TRUE)

perpetuity.arith(pv=NA,p=1,q=.5,i=.07,ic=1,pf=1,imm=TRUE)

perpetuity.arith(pv=100,p=NA,q=1,i=.05,ic=.5,pf=1,imm=FALSE)
```

perpetuity.geo *Geometric Perpetuity*

Description

Solves for the present value, amount of the first payment, the payment growth rate, or the interest rate for a geometrically growing perpetuity.

Usage

```
perpetuity.geo(pv=NA,p=NA,k=NA,i=NA,ic=1,pf=1,imm=TRUE)
```

Arguments

pv	present value
p	amount of the first payment
k	payment growth rate per period
i	nominal interest rate convertible ic times per year
ic	interest conversion frequency per year
pf	the payment frequency- number of payments and periods per year
imm	option for perpetuity immediate or due, default is immediate (TRUE)

Details

Effective Rate of Interest: $eff.i = (1 + \frac{i}{ic})^{ic} - 1$

$j = (1 + eff.i)^{\frac{1}{pf}} - 1$

Perpetuity Immediate:

j > k: $pv = \frac{p}{j-k}$

Perpetuity Due:

j > k: $pv = \frac{p}{j-k} * (1 + j)$

Value

Returns a matrix of the input variables and calculated unknown variables.

Note

One of pv, p, k, or i must be NA (unknown).

See Also

perpetuity.arith

perpetuity.level

annuity.arith

annuity.geo

annuity.level

Examples

```
perpetuity.geo(pv=NA,p=5,k=.03,i=.04,ic=1,pf=1,imm=TRUE)

perpetuity.geo(pv=1000,p=5,k=NA,i=.04,ic=1,pf=1,imm=FALSE)
```

---

perpetuity.level *Level Perpetuity*

Description

Solves for the present value, interest rate, or the amount of the payments for a level perpetuity.

Usage

```
perpetuity.level(pv=NA,pmt=NA,i=NA,ic=1,pf=1,imm=TRUE)
```

Arguments

pv	present value
pmt	value of level payments
i	nominal interest rate convertible ic times per year
ic	interest conversion frequency per year
pf	the payment frequency- number of payments per year
imm	option for perpetuity immediate or annuity due, default is immediate (TRUE)

Details

Effective Rate of Interest: $eff.i = (1 + \frac{i}{ic})^{ic} - 1$

$j = (1 + eff.i)^{\frac{1}{pf}} - 1$

Perpetuity Immediate:

$pv = pmt * a_{\overline{\infty}|j} = \frac{pmt}{j}$

Perpetuity Due:

$pv = pmt * \ddot{a}_{\overline{\infty}|j} = \frac{pmt}{j} * (1 + i)$

Value

Returns a matrix of the input variables and calculated unknown variables.

Note

One of pv, pmt, or i must be NA (unknown).

Author(s)

Kameron Penn and Jack Schmidt

See Also

`perpetuity.arith`

`perpetuity.geo`

`annuity.arith`

`annuity.geo`

`annuity.level`

Examples

```
perpetuity.level(pv=100,pmt=NA,i=.05,ic=1,pf=2,imm=TRUE)

perpetuity.level(pv=100,pmt=NA,i=.05,ic=1,pf=2,imm=FALSE)
```

`protective.put` *Protective Put*

Description

Gives a table and graphical representation of the payoff and profit of a protective put strategy for a range of future stock prices.

Usage

```
protective.put(S,K,r,t,sd,price=NA,plot=FALSE)
```

Arguments

`S`	spot price at time 0
`K`	strike price
`r`	continuously compounded yearly risk free rate
`t`	time of expiration (in years)
`sd`	standard deviation of the stock (volatility)
`price`	specified put price if the Black Scholes pricing is not desired (leave as NA to use the Black Scholes pricing)
`plot`	tells whether or not to plot the payoff and profit

Details

Stock price at time t $= S_t$

For $S_t <= K$: payoff $= K - S$

For $S_t > K$: payoff $= S_t - S$

profit = payoff - price$*e^{r*t}$

Value

A list of two components.

`Payoff`	A data frame of different payoffs and profits for given stock prices.
`Premium`	The price of the put option.

Note

Finds the put price by using the Black Scholes equation by default.

Author(s)

Kameron Penn and Jack Schmidt

See Also

`option.put`

Examples

```
protective.put(S=100,K=100,r=.03,t=1,sd=.2)

protective.put(S=100,K=90,r=.01,t=.5,sd=.1)
```

`rate.conv` *Interest, Discount, and Force of Interest Converter*

Description

Converts given rate to desired nominal interest, discount, and force of interest rates.

Usage

```
rate.conv(rate, conv=1, type="interest", nom=1)
```

Arguments

`rate`	current rate
`conv`	how many times per year the current rate is convertible
`type`	current rate as one of "interest", "discount" or "force"
`nom`	desired number of times the calculated rates will be convertible

Details

$1 + i = (1 + \frac{i^{(n)}}{n})^n = (1 - d)^{-1} = (1 - \frac{d^{(m)}}{m})^{-m} = e^\delta$

Value

A matrix of the interest, discount, and force of interest conversions for effective, given and desired conversion rates.

The row named 'eff' is used for the effective rates, and the nominal rates are in a row named 'nom(x)' where the rate is convertible x times per year.

Author(s)

Kameron Penn and Jack Schmidt

Examples

```
rate.conv(rate=.05,conv=2,nom=1)

rate.conv(rate=.05,conv=2,nom=4,type="discount")

rate.conv(rate=.05,conv=2,nom=4,type="force")
```

---

straddle *Straddle Spread*

Description

Gives a table and graphical representation of the payoff and profit of a long or short straddle for a range of future stock prices.

Usage

```
straddle(S,K,r,t,price1,price2,position,plot=FALSE)
```

Arguments

`S`	spot price at time 0
`K`	strike price of the call and put
`r`	continuously compounded yearly risk free rate
`t`	time of expiration (in years)
`price1`	price of the long call with strike price K
`price2`	price of the long put with strike price K
`position`	either buyer or seller of option ("long" or "short")
`plot`	tells whether or not to plot the payoff and profit

Details

Stock price at time t $= S_t$

Long Position:

For $S_t <= K$: payoff $= K - S_t$

For $S_t > K$: payoff $= S_t - K$

profit = payoff - (price1 + price2)$*e^{r*t}$

Short Position:

For $S_t <= K$: payoff $= S_t - K$

For $S_t > K$: payoff $= K - S_t$

profit = payoff + (price1 + price2)$*e^{r*t}$

Value

A list of two components.

`Payoff`	A data frame of different payoffs and profits for given stock prices.
`Premiums`	A matrix of the premiums for the call and put options, and the net cost.

See Also

`straddle.bls`

`option.put`

`option.call`

`strangle`

Examples

```
straddle(S=100,K=110,r=.03,t=1,price1=15,price2=10,position="short")
```

`straddle.bls` *Straddle Spread - Black Scholes*

Description

Gives a table and graphical representation of the payoff and profit of a long or short straddle for a range of future stock prices. Uses the Black Scholes equation for the call and put prices.

Usage

```
straddle.bls(S,K,r,t,sd,position,plot=FALSE)
```

Arguments

S	spot price at time 0
K	strike price of the call and put
r	continuously compounded yearly risk free rate
t	time of expiration (in years)
sd	standard deviation of the stock (volatility)
position	either buyer or seller of option ("long" or "short")
plot	tells whether or not to plot the payoff and profit

Details

Stock price at time t $= S_t$

Long Position:

For $S_t <= K$: payoff $= K - S_t$

For $S_t > K$: payoff $= S_t - K$

profit = payoff$-(price_{call} + price_{put}) * e^{r*t}$

Short Position:

For $S_t <= K$: payoff $= S_t - K$

For $S_t > K$: payoff $= K - S_t$

profit = payoff$+(price_{call} + price_{put}) * e^{r*t}$

Value

A list of two components.

Payoff	A data frame of different payoffs and profits for given stock prices.
Premiums	A matrix of the premiums for the call and put options, and the net cost.

See Also

option.put

option.call

strangle.bls

Examples

```
straddle.bls(S=100,K=110,r=.03,t=1,sd=.2,position="short")

straddle.bls(S=100,K=110,r=.03,t=1,sd=.2,position="long",plot=TRUE)
```

strangle — *Strangle Spread*

Description

Gives a table and graphical representation of the payoff and profit of a long strangle spread for a range of future stock prices.

Usage

```
strangle(S,K1,K2,r,t,price1,price2,plot=FALSE)
```

Arguments

`S`	spot price at time 0
`K1`	strike price of the long put
`K2`	strike price of the long call
`r`	continuously compounded yearly risk free rate
`t`	time of expiration (in years)
`price1`	price of the long put with strike price K1
`price2`	price of the long call with strike price K2
`plot`	tells whether or not to plot the payoff and profit

Details

Stock price at time t $= S_t$
For $S_t <= K1$: payoff $= K1 - S_t$
For $K1 < S_t < K2$: payoff $= 0$
For $S_t >= K2$: payoff $= S_t - K2$
profit = payoff - (price1 + price2)$*e^{r*t}$

Value

A list of two components.

`Payoff`	A data frame of different payoffs and profits for given stock prices.
`Premiums`	A matrix of the premiums for the call and put options, and the net cost.

Note

K1 < S < K2 must be true.

Author(s)

Kameron Penn and Jack Schmidt

See Also

strangle.bls

option.put

option.call

straddle

Examples

```
strangle(S=105,K1=100,K2=110,r=.03,t=1,price1=10,price2=15,plot=TRUE)
```

---

strangle.bls *Strangle Spread - Black Scholes*

Description

Gives a table and graphical representation of the payoff and profit of a long strangle spread for a range of future stock prices. Uses the Black Scholes equation for the call prices.

Usage

```
strangle.bls(S,K1,K2,r,t,sd,plot=FALSE)
```

Arguments

S	spot price at time 0
K1	strike price of the long put
K2	strike price of the long call
r	continuously compounded yearly risk free rate
t	time of expiration (in years)
sd	standard deviation of the stock (volatility)
plot	tells whether or not to plot the payoff and profit

Details

Stock price at time t $= S_t$

For $S_t <= K1$: payoff $= K1 - S_t$

For $K1 < S_t < K2$: payoff $= 0$

For $S_t >= K2$: payoff $= S_t - K2$

profit = payoff$-(price_{K1} + price_{K2}) * e^{r*t}$

Value

A list of two components.

`Payoff`	A data frame of different payoffs and profits for given stock prices.
`Premiums`	A matrix of the premiums for the call and put options, and the net cost.

Note

K1 < S < K2 must be true.

Author(s)

Kameron Penn and Jack Schmidt

See Also

`option.put`

`option.call`

`straddle.bls`

Examples

```
strangle.bls(S=105,K1=100,K2=110,r=.03,t=1,sd=.2)

strangle.bls(S=115,K1=50,K2=130,r=.03,t=1,sd=.2)
```

`swap.commodity` *Commodity Swap*

Description

Solves for the fixed swap price, given the variable prices and interest rates (either as spot rates or zero coupon bond prices).

Usage

```
swap.commodity(prices, rates, type="spot_rate")
```

Arguments

`prices`	vector of variable prices
`rates`	vector of variable rates
`type`	rates defined as either "spot_rate" or "zcb_price"

Details

For spot rates: $\sum_{k=1}^{n} \frac{prices_k}{(1+rates_k)^k} = \sum_{k=1}^{n} \frac{X}{(1+rates_k)^k}$

For zero coupon bond prices: $\sum_{k=1}^{n} prices_k * rates_k = \sum_{k=1}^{n} X * rates_k$

Where $X =$ fixed swap price.

Value

The fixed swap price.

Note

Length of the price vector and rate vector must be of the same length.

Author(s)

Kameron Penn and Jack Schmidt

See Also

`swap.rate`

Examples

```
swap.commodity(prices=c(103,106,108), rates=c(.04,.05,.06))

swap.commodity(prices=c(103,106,108), rates=c(.9615,.907,.8396),type="zcb_price")

swap.commodity(prices=c(105,105,105), rates=c(.85,.89,.80),type="zcb_price")
```

---

swap.rate *Interest Rate Swap*

Description

Solves for the fixed interest rate given the variable interest rates (either as spot rates or zero coupon bond prices).

Usage

```
swap.rate(rates, type="spot_rate")
```

Arguments

`rates`	vector of variable rates
`type`	rates as either "spot_rate" or "zcb_price"

Details

For spot rates: $1 = \sum_{k=1}^{n}[\frac{R}{(1+rates_k)^k}] + \frac{1}{(1+rates_n)^n}$

For zero coupon bond prices: $1 = \sum_{k=1}^{n}(R * rates_k) + rates_n$

Where $R =$ fixed swap rate.

Value

The fixed interest rate swap.

See Also

swap.commodity

Examples

```
swap.rate(rates=c(.04, .05, .06), type = "spot_rate")

swap.rate(rates=c(.93,.95,.98,.90), type = "zcb_price")
```

TVM *Time Value of Money*

Description

Solves for the present value, future value, time, or the interest rate for the accumulation of money earning compound interest. It can also plot the time value for each period.

Usage

```
TVM(pv=NA,fv=NA,n=NA,i=NA,ic=1,plot=FALSE)
```

Arguments

pv	present value
fv	future value
n	number of periods
i	nominal interest rate convertible ic times per period
ic	interest conversion frequency per period
plot	tells whether or not to produce a plot of the time value at each period

Details

$j = (1 + \frac{i}{ic})^{ic} - 1$

$fv = pv * (1 + j)^n$

Value

Returns a matrix of the input variables and calculated unknown variables.

Note

Exactly one of pv, fv, n, or i must be NA (unknown).

See Also

`cf.analysis`

Examples

```
TVM(pv=10,fv=20,i=.05,ic=2,plot=TRUE)

TVM(pv=50,n=5,i=.04,plot=TRUE)
```

`yield.dollar` *Dollar Weighted Yield*

Description

Calculates the dollar weighted yield.

Usage

```
yield.dollar(cf, times, start, end, endtime)
```

Arguments

`cf`	vector of cash flows
`times`	vector of times for when cash flows occur
`start`	beginning balance
`end`	ending balance
`endtime`	end time of comparison

Details

$I = end - start - \sum_{k=1}^{n} cf_k$

$i^{dw} = \frac{I}{start*endtime - \sum_{k=1}^{n} cf_k*(endtime - times_k)}$

Value

The dollar weighted yield.

Note

Time of comparison (endtime) must be larger than any number in vector of cash flow times.

Length of cashflow vector and times vector must be equal.

See Also

`yield.time`

Examples

```
yield.dollar(cf=c(20,10,50),times=c(.25,.5,.75),start=100,end=175,endtime=1)

yield.dollar(cf=c(500,-1000),times=c(3/12,18/12),start=25200,end=25900,endtime=21/12)
```

`yield.time` *Time Weighted Yield*

Description

Calculates the time weighted yield.

Usage

```
yield.time(cf,bal)
```

Arguments

`cf`	vector of cash flows
`bal`	vector of balances

Details

$i^{tw} = \prod_{k=1}^{n}(\frac{bal_{1+k}}{bal_k+cf_k}) - 1$

Value

The time weighted yield.

Note

Length of cash flows must be one less than the length of balances.

If lengths are equal, it will not use final cash flow.

Author(s)

Kameron Penn and Jack Schmidt

See Also

yield.dollar

Examples

```
yield.time(cf=c(0,200,100,50),bal=c(1000,800,1150,1550,1700))
```

Index