Package ‘ifm’

June 2, 2016

Type Package

Title Set of functions for financial evaluation of Software Projects

Version 1.0

Date 2016-04-20

Author Eber Schmitz

Maintainer Alexandre Costa <afcosta@br.ibm.com> and
Antoanne Pontes <antoanne@ufrj.br> and
Eduardo Chiote <eduardochiote@gmail.com>

Description R packeage with a set of functions for financial evaluation of
Software Project.

License LGPL (>= 2.1)

URL https://github.com/afcosta-ibm/ifm

BugReports https://github.com/afcosta-ibm/ifm/issues

NeedsCompilation no

RoxygenNote 5.0.1

Imports igraph, XLConnect

R topics documented:

`ifm-package` *Set of functions for financial evaluation of Software Projects*

Description

R packeage with a set of functions for financial evaluation of Software Project.

Details

The DESCRIPTION file: This package was not yet installed at build time.

Index: This package was not yet installed at build time.
~~ An overview of how to use the package, including the most important functions ~~

Author(s)

Eber Schmitz

Maintainer: Alexandre Costa <afcosta@br.ibm.com> and Antoanne Pontes <antoanne@ufrj.br> and Eduardo Chiote <eduardochiote@gmail.com>

References

~~ Literature or other references for background information ~~

See Also

~~ Optional links to other man pages, e.g. ~~

Examples

```
## examples here...
```

`cpm` *The critical path method (CPM) is a step-by-step project management technique for process planning that defines critical and non-critical tasks with the goal of preventing time-frame problems and process bottlenecks#' activities are "critical," meaning that they have to be done on time or else the whole project will take longer.*

Description

The Critical Path Method or Critical Path Analysis, is a mathematically based algorithm for scheduling a set of project activities.

CPM will get how long your complex project will take to complete and which activities are "critical," meaning that they have to be done on time or else the whole project will take longer.

Usage

```
cpm(activities.duration = c(1, 4, 5, 7, 2, 3, 1),
  activities.successors = list(c(2, 3), 4, c(4, 5), 6, 7, 7, c(0)))
```

Arguments

`activities.duration`	Vector with activities duration.
`activities.successors`	Vector with dependencies between activities.

Value

Returns list of EST (Early Start Time), EFT(Early Finish Time),LST(Lately Start Time), LFT (Lately Finish Time) using Forward Pass and Backward Pass

See Also

Other scheduling: `cpm.all.schedule`, `mmf.all.sequences`, `mmf.get.breakeven`, `mmf.get.selffunding`, `mmf.npv`

Examples

```
ex.cpm.activities.duration <- c(1,4,5,7,2,3,1)
ex.cpm.activities.successors <- list(c(2,3), 4, c(4,5), 6, 7, 7, c(0))
ex.cpm <- cpm(ex.cpm.activities.duration,
              ex.cpm.activities.successors)
```

cpm.all.schedule *Generates all possible schedules for a cpm network*

Description

Generates all possible schedules for a cpm network

Usage

```
cpm.all.schedule(est, slack)
```

Arguments

`est`	early start time vector
`slack`	activities slack

Value

Matrix with all mininum makespan (the time to complete all jobs) cpm start time schedules

See Also

Other scheduling: `cpm`, `mmf.all.sequences`, `mmf.get.breakeven`, `mmf.get.selffunding`, `mmf.npv`

Examples

```
#Use critical.path.method function to calculate a set of project
#activities:

ex.cpm.activities.duration <- c(1,4,5,7,2,3,1)
ex.cpm.activities.successors <- list(c(2,3), 4, c(4,5), 6, 7, 7, c(0))
ex.cpm <- cpm(ex.cpm.activities.duration,
              ex.cpm.activities.successors)

# Now, we have the CPM vector with:
# - est (Early Start Time)  - ex.cpm[[1]]
# - eft (Early Finish Time) - ex.cpm[[2]]
# - lst (Late Start Time)   - ex.cpm[[3]]
# - lft (Late Finish Time)  - ex.cpm[[4]]

ex.cpm.activities.schedule <-
   cpm.all.schedule(ex.cpm[['est']], ex.cpm[['lst']] - ex.cpm[['est']])

# note: ex.cpm['lst'] - ex.cpm['est'] is the slack time (or 'float') for each task
```

`discount.rate.vector` *Vectorize the Discount Rate*

Description

Generate a vector with discount rate to be applied to each of the time periods.

Usage

```
discount.rate.vector(interest.rate, number.of.periods,
  begin.of.period = FALSE)
```

Arguments

`interest.rate`	A number that represents the nominal Interest Rate, presented by year.
`number.of.periods`	Times that interest rate should be applied.
`begin.of.period`	A boolean that represents if the Tax Rate will be applied at the begining of period. FALSE by default, represents that Tax Rate will be applied at second period .

Value

vector with discount rates

See Also

Other financial: `discounted.csf`, `draw.cfs`, `ifir`, `net.future.value`, `net.present.value`

Examples

```
ex.disc.vector <- discount.rate.vector(0.0619, 12)
```

`discounted.csf` *The cash flows incomes/outcomes applying the Tax Rate to the present time.*

Description

The cash flows incomes/outcomes applying the Tax Rate to the present time.

Usage

```
discounted.csf(cfs = c(-350, 100, 200, 150, 75), interest.rate = 0.0619,
  begin.of.period = FALSE)
```

Arguments

`cfs`	A vector with a series of cash flows.
`interest.rate`	A number that represents the nominal Interest Rate, presented by year.
`begin.of.period`	A boolean that represents if the Tax Rate will be applied at the begining of period. FALSE by default, the Tax Rate will be applied to the second period.

Value

The vector of cash flows incomes/outcomes applying the Tax Rate to the present time.

See Also

Other financial: `discount.rate.vector`, `draw.cfs`, `ifir`, `net.future.value`, `net.present.value`

Examples

```
ex.disc.csf <- discounted.csf(c(-350,100,200,150,75), 0.0619, FALSE)
```

`draw.cfs` *Draw the graph of cash flow*

Description

Draw the graph of cash flow in order to facilitate the study and the effects of the analysis of a certain application.

Cash flow is a mathematical concept that can be plotted in order to facilitate the study and the effects of the analysis of a certain

application, which may be an investment loan, finance, etc.

Normally a cash flow contains inputs and outputs of capital, marked in

the timeline starting at t = 0.

A typical example is the graph that represents a bank loan held by a form

of business that shall return this loan in n equal installments over

the following months.

E1 E2 E3 ... En-1 En

^

I

0 1 2 3 ... n-1 n

I I I

V V V

S1 S2 S3 ... Sn-1 Sn

Is possible to note that the value is entered in the company's cash (cash was

positive) and S1, S2, ..., Sn are the values of the parcels will leave the company's cash (negative).

The fact that each arrow is pointing upward (positive) or down (negative),

it is assumed by convention, and the cash flow will depend on who receives

or pays the Capital at a certain time, and:

t = 0 indicates the current day;

Ek is the capital input at a time k;

Sk is the capital output at a time k.

Usage

```
draw.cfs(cfs, gt = "Cash Flow Graphic", to.file = FALSE,
  filename = "output/draw.cfs.graph.png")
```

Arguments

`cfs`	A vector with a series of cash flows.
`gt`	A title for the graph.
`to.file`	Save or not the graph in the file
`filename`	File's name

Value

A plot with cash flow series

See Also

Other financial: `discount.rate.vector`, `discounted.csf`, `ifir`, `net.future.value`, `net.present.value`

Examples

```
ex.cfs <- c(-2000,1000,1500,-500,500)
draw.cfs(ex.cfs,'My Cash Flow')
```

`draw.graph` *Draw the graph imported from the spreadsheet.*

Description

This function is responsible for ploting the graph based on the edges and export the image to a file.

Arguments

`edges`	- A vector defining the edges, the first edge points from the first element to the second, the second edge from the third to the fourth, etc.

Value

graph.image.path - The path to the generated graph file.

Examples

```
ex.graph.image.path <- draw.graph(c(1,2, 1,3, 2,3, 3,4))
```

`excel.list.to.xls` *Export the generated ifm package results to a spreadsheet.*

Description

This function is responsible for reading a list of objects and export a spreadsheet with the results proccessed by the IFM package. The file contains:

- The raw data frame used to calculate the maxNPV, minSF and minBKE;
- The image oh the generated graph;
- The image oh the "Discounted Cash x Time" chart;
- The image oh the "MPV (ca$h) x Schedulling ID" chart;
- The image oh the "Self Funding (time) x Schedulling ID" chart;
- The image oh the "Breaking Event (time) x Schedulling ID" chart;

Arguments

`list.ifm.result`	The list with all results proccessed by the IFM package.

Value

file.path The path to the generated file.

See Also

Other utility: `excel.xls.to.list`, `predecessors.to.edges`

Examples

```
ex.sheet.data <- excel.xls.to.list("../resources/spreadsheet.xls")
ex.sheet.data.interest.rate <- ex.sheet.data[[1]]
ex.sheet.data.activities <- ex.sheet.data[[2]]
ex.sheet.data.durations <- ex.sheet.data[[3]]
ex.sheet.data.predecessors <- ex.sheet.data[[4]]
ex.sheet.data.cfs <- ex.sheet.data[[5]]

ex.mmf.seq <- mmf.all.sequences(ex.sheet.data.predecessors)

ex.mmf <- mmf.npv(ex.sheet.data.cfs,
                  ex.sheet.data.durations,
                  ex.mmf.seq,
                  ex.sheet.data.interest.rate)

ex.mmf.schedules <- ex.mmf[['schedules']]
ex.mmf.cfs.nominal <- ex.mmf[['cfs.nominal']]
ex.mmf.cfs.discounted <- ex.mmf[['cfs.discounted']]
ex.mmf.npv <- ex.mmf[['npv']]

ex.mmf.npv.selffunding <- mmf.get.selffunding(ex.mmf.cfs.discounted)
ex.mmf.npv.breakeven <- mmf.get.breakeven(ex.mmf.cfs.discounted)

ex.mmf.df.1r <- mmf.df.1r(ex.mmf.seq,
                          ex.mmf.schedules,
                          ex.mmf.npv,
                          ex.mmf.npv.selffunding,
                          ex.mmf.npv.breakeven)

ex.file.path <- excel.list.to.xls(ex.mmf.df.1r)
```

`excel.xls.to.list` *Extract a list of variables from the spreadsheet to be used on the maxNPV function.*

Description

This function is responsible for reading a spreadsheet representing the project, and return a list with the following information (in this order): The interest rate, the list of activities, the list of durations of activities, the list of predecessors of activities, the matrix that represents the cash flow series and the graph edges based on the predecessor list

Arguments

`xls.spreadsheet.path`	The complete path to the spreadsheet that represents the project.

Value

List of variables to be used on the maxNPV function.

See Also

Other utility: `excel.list.to.xls`, `predecessors.to.edges`

Examples

```
ex.sheet.data <- excel.xls.to.list("../resources/spreadsheet.xls")
```

ifir *Calculate the Inflation-free Interest Rate.*

Description

Calculate the Inflation-free Interest Rate.

Usage

```
ifir(interest.rate = 0.1425, inflation.rate = 0.0759)
```

Arguments

`interest.rate`	A number that represents the nominal Interest Rate, presented by year.
`inflation.rate`	A number that represents the Inflation Rate, presented by year.

Value

Returns the inflation-free interest rate

See Also

Other financial: `discount.rate.vector`, `discounted.csf`, `draw.cfs`, `net.future.value`, `net.present.value`

Examples

```
ex.ifir <- inflation.free.interest.rate(0.1425, 0.0759)
```

mmf.all.sequences *Generates the topsorts list*

Description

Generates the list of all possible MMF sequences (topsorts), constrained by the predecessors.

Usage

```
mmf.all.sequences(predecessors = 0)
```

Arguments

`predecessors`	List of Predecessors - Zero for none. The index of the list of predecessors represents the id of MMF and the value.

Value

List of all possible MMF sequences.

See Also

Other scheduling: `cpm.all.schedule`, `cpm`, `mmf.get.breakeven`, `mmf.get.selffunding`, `mmf.npv`

Examples

```
ex.activities.predecessors<-list(0,1,2,3,1,5,c(4,6))
ex.mmf.seq <- mmf.all.sequences(ex.activities.predecessors)
```

`mmf.df.1r` *Generates a data frame with Sequence, Schedule, NPV, Breakeven and Self Funding*

Description

Generates a data frame with Sequence, Schedule, NPV, Breakeven and Self Funding

Usage

```
mmf.df.1r(mmf.seq, mmf.sched, mmf.npv, mmf.npv.selffunding, mmf.npv.breakeven)
```

Arguments

`mmf.seq`	A list of sequences
`mmf.sched`	A list of schedules
`mmf.npv`	A list of NPV values
`mmf.npv.selffunding`	A list of Selffunding times
`mmf.npv.breakeven`	A list of Breakeven times

Examples

```
ex.sheet.data <- excel.xls.to.list("resources/spreadsheet.xls")
ex.sheet.data.interest.rate <- ex.sheet.data[[1]]
ex.sheet.data.activities <- ex.sheet.data[[2]]
ex.sheet.data.durations <- ex.sheet.data[[3]]
ex.sheet.data.predecessors <- ex.sheet.data[[4]]
ex.sheet.data.cfs <- ex.sheet.data[[5]]

ex.mmf.seq <- mmf.all.sequences(ex.sheet.data.predecessors)

ex.mmf <- mmf.npv(ex.sheet.data.cfs,
                  ex.sheet.data.durations,
                  ex.mmf.seq,
                  ex.sheet.data.interest.rate)

ex.mmf.schedules <- ex.mmf[['schedules']]
ex.mmf.cfs.nominal <- ex.mmf[['cfs.nominal']]
ex.mmf.cfs.discounted <- ex.mmf[['cfs.discounted']]
ex.mmf.npv <- ex.mmf[['npv']]

ex.mmf.npv.selffunding <- mmf.get.selffunding(ex.mmf.cfs.discounted)
ex.mmf.npv.breakeven <- mmf.get.breakeven(ex.mmf.cfs.discounted)

ex.mmf.df.1r <- mmf.df.1r(ex.mmf.seq,
                          ex.mmf.schedules,
                          ex.mmf.npv,
                          ex.mmf.npv.selffunding,
                          ex.mmf.npv.breakeven)
```

`mmf.get.breakeven` *Get a list with all Breakeven points from CFS*

Description

Get a list with all Breakeven points from CFS

Usage

```
mmf.get.breakeven(mmf.cfs)
```

Arguments

`mmf.cfs`	A list with a vector with a series of cash flows for each MMF sechedule.

Value

A list with all Breakeven points for each MMF

See Also

Other scheduling: `cpm.all.schedule`, `cpm`, `mmf.all.sequences`, `mmf.get.selffunding`, `mmf.npv`

Examples

```
ex.sheet.data <- excel.xls.to.list("resources/spreadsheet.xls")
ex.sheet.data.interest.rate <- ex.sheet.data[[1]]
ex.sheet.data.activities <- ex.sheet.data[[2]]
ex.sheet.data.durations <- ex.sheet.data[[3]]
ex.sheet.data.predecessors <- ex.sheet.data[[4]]
ex.sheet.data.cfs <- ex.sheet.data[[5]]

ex.mmf.seq <- mmf.all.sequences(ex.sheet.data.predecessors)

ex.mmf <- mmf.npv(ex.sheet.data.cfs,
                  ex.sheet.data.durations,
                  ex.mmf.seq,
                  ex.sheet.data.interest.rate)

ex.mmf.npv.selffunding <- mmf.get.breakeven(ex.mmf[['cfs.discounted']])
```

`mmf.get.selffunding` *Get a list with all Selffunding points from CFS*

Description

Get a list with all Selffunding points from CFS

Usage

```
mmf.get.selffunding(mmf.cfs)
```

Arguments

`mmf.cfs`	A list with a vector with a series of cash flows for each MMF sechedule.

Value

A list with all Selffunding points for each MMF

See Also

Other scheduling: `cpm.all.schedule`, `cpm`, `mmf.all.sequences`, `mmf.get.breakeven`, `mmf.npv`

Examples

```
ex.sheet.data <- excel.xls.to.list("resources/spreadsheet.xls")
ex.sheet.data.interest.rate <- ex.sheet.data[[1]]
ex.sheet.data.activities <- ex.sheet.data[[2]]
ex.sheet.data.durations <- ex.sheet.data[[3]]
ex.sheet.data.predecessors <- ex.sheet.data[[4]]
ex.sheet.data.cfs <- ex.sheet.data[[5]]

ex.mmf.seq <- mmf.all.sequences(ex.sheet.data.predecessors)

ex.mmf <- mmf.npv(ex.sheet.data.cfs,
                  ex.sheet.data.durations,
                  ex.mmf.seq,
                  ex.sheet.data.interest.rate)

ex.mmf.npv.selffunding <- mmf.get.selffunding(ex.mmf[['cfs.discounted']])
```

`mmf.max.npv` *Return Max NPV*

Description

this function identifies the sequence of activities and respectivies schedules where with the optimized NPV

Usage

```
mmf.max.npv(mmf.npv, mmf.seq, mmf.schedules)
```

Arguments

`mmf.npv`	Vector of Net Present Value
`mmf.seq`	Vector with the sequence of activities
`mmf.schedules`	Vector with the collection of possible schedules

Value

list with NPV, sequence and scheduleof the sequence with the maximum NPV

Examples

```
ex.sheet.data <- excel.xls.to.list("resources/spreadsheet.xls")
ex.sheet.data.interest.rate <- ex.sheet.data[[1]]
ex.sheet.data.activities <- ex.sheet.data[[2]]
ex.sheet.data.durations <- ex.sheet.data[[3]]
ex.sheet.data.predecessors <- ex.sheet.data[[4]]
ex.sheet.data.cfs <- ex.sheet.data[[5]]

ex.mmf.seq <- mmf.all.sequences(ex.sheet.data.predecessors)

ex.mmf <- mmf.npv(ex.sheet.data.cfs,
                  ex.sheet.data.durations,
                  ex.mmf.seq,
                  ex.sheet.data.interest.rate)

ex.mmf.shedules <- ex.mmf[['shedules']]
ex.mmf.cfs.nominal <- ex.mmf[['cfs.nominal']]
ex.mmf.cfs.discounted <- ex.mmf[['cfs.discounted']]
ex.mmf.npv <- ex.mmf[['npv']]
```

`mmf.npv` *Calculates NPV for all schedules*

Description

Calculates NPV for all schedules

Usage

```
mmf.npv(cfs, durations, all.sequences, interest.rate, begin.of.period = FALSE)
```

Arguments

`cfs`	A vector with a series of cash flows.
`durations`	A vector with a list of activities durations.
`all.sequences`	List of all possible MMF sequences.
`interest.rate`	A number that represents the nominal Interest Rate, presented by year.
`begin.of.period`	A boolean that represents if the Tax Rate will be applied at the begining of period. FALSE by default, represents that Tax Rate will be applied at second period .

Value

A list with all shedules, all npv csf and sum of each npv.

See Also

Other scheduling: `cpm.all.schedule`, `cpm`, `mmf.all.sequences`, `mmf.get.breakeven`, `mmf.get.selffunding`

Examples

```
# Loading data from XLS
ex.sheet.data <- excel.xls.to.list("../resources/spreadsheet.xls")
ex.sheet.data.interest.rate <- ex.sheet.data[[1]]
ex.sheet.data.activities <- ex.sheet.data[[2]]
ex.sheet.data.durations <- ex.sheet.data[[3]]
ex.sheet.data.predecessors <- ex.sheet.data[[4]]
ex.sheet.data.cfs <- ex.sheet.data[[5]]

# Generating all possible implementation sequences
ex.mmf.seq <- mmf.all.sequences(ex.sheet.data.predecessors)

# Calculating NVP to all possible sequences
ex.mmf.npv <- mmf.max.npv(ex.sheet.data.cfs,
                          ex.sheet.data.durations,
                          ex.mmf.seq,
                          ex.sheet.data.interest.rate)

# Selecting sequence ID which max NPV
ex.mmf.npv.max <- which.max(ex.mmf.npv[[3]])
ex.mmf.sched <- ex.mmf.npv[[1]]
ex.mmf.npv <- ex.mmf.npv[[2]]
ex.mmf.npv.sum <- ex.mmf.npv[[3]]

# Index of sequence with max NPV
# ex.mmf.npv.max <- which.max(ex.mmf.npv.sum)

# Value of max NPV
ex.mmf.npv.max.value <- ex.mmf.npv.sum[[ex.mmf.npv.max]]

# Sequence with best NPV
ex.mmf.npv.max.sequence <- ex.mmf.seq[ex.mmf.npv.max]

# Schedule of sequence with best NPV
ex.mmf.npv.max.sched <- ex.mmf.sched[ex.mmf.npv.max]
```

`net.future.value` *Net Future Value is a combination of different future values from different times, all which are put into one larger present value.*

Description

Net Future Value is a combination of different future values from different times, all which are put into one larger present value.

Usage

```
net.future.value(cfs = c(-350, 100, 200, 150, 75), interest.rate = 0.0619,
  begin.of.period = TRUE)
```

Arguments

`cfs`	A vector with a series of cash flows.
`interest.rate`	A number that represents the nominal Interest Rate, presented by year.
`begin.of.period`	A boolean that represents if the Tax Rate will be applied at the begining of period. FALSE by default.

Value

A future value of a cash flow series.

See Also

Other financial: `discount.rate.vector`, `discounted.csf`, `draw.cfs`, `ifir`, `net.present.value`

Examples

```
ex.nfv <- net.future.value(c(-350,100,200,150,75), 0.0619, TRUE)
```

`net.present.value` *Difference between the present values of cash inflows and outflows*

Description

calculates the difference between the present values of cash inflows and outflows.

Usage

```
net.present.value(cfs = c(-350, 100, 200, 150, 75), interest.rate = 0.0619,
  begin.of.period = TRUE)
```

Arguments

`cfs`	A vector with a series of cash flows.
`interest.rate`	A number that represents the nominal Interest Rate, presented by year.
`begin.of.period`	A boolean that represents if the Tax Rate will be applied at the begining of period. FALSE by default, the Tax Rate will be applied to the second period.

Value

The sum of cash flows incomes/outcomes applying the Tax Rate to the present time

See Also

Other financial: `discount.rate.vector`, `discounted.csf`, `draw.cfs`, `ifir`, `net.future.value`

Examples

```
ex.npv <- net.present.value(c(-350,100,200,150,75), 0.0619, TRUE)
```

`predecessors.to.edges` *Generate a vector with the edges to be plotted by draw.graph function.*

Description

This function is responsible for reading a vector with all activities predecessors and generate a list of the edges that will be plotted by the draw.graph function.

Usage

```
predecessors.to.edges(list.of.predecessors = list())
```

Arguments

`list.of.predecessors`	The vector that contain all activities predecessors.

Value

edges - List of edges to be used on the draw.graph function.

See Also

Other utility: `excel.list.to.xls`, `excel.xls.to.list`

Examples

```
ex.edges <- predecessors.to.edges(ex.sheet.data$predecessors)
```

`schedules.1r` *Generates all schedules for ONE resource, Denne Method.*

Description

Generates all schedules for ONE resource, Denne Method.

Usage

```
schedules.1r(sequences, durations)
```

Arguments

`sequences`	All sequences
`durations`	Duration of activities

Index