Package ‘maRketSim’

February 20, 2015

**Type**  Package
**Title**  Market simulator for R
**Version**  0.9.2
**Date**  2013-7-16
**Author**  Ari Friedman
**Maintainer**  Ari Friedman &lt;abfriedman@gmail.com&gt;

**Description**  maRketSim is a market simulator for R. It was initially designed around the bond market, with plans to expand to stocks. maRketSim is built around the idea of portfolios of fundamental objects. Therefore it is slow in its current incarnation, but allows you the flexibility of seeing exactly what is in your final results, since the objects are retained.

**License**  GPL (>= 2)
**LazyLoad**  yes
**LazyData**  yes
**Suggests**  taRifx, gsl
**Depends**  R (>= 2.15.1)
**NeedsCompilation**  no

**Repository**  CRAN
**Date/Publication**  2013-07-17 07:13:19

**R topics documented:**

- maRketSim-package ........................................ 2
- aa ......................................................... 3
- account ................................................... 4
- as.data.frame.bond ....................................... 5
- cashflow .................................................. 5
- connectTheDots ........................................... 6
- count.history.account ................................. 7
- current.market ........................................... 7
- current.portfolio.bond ................................. 8
Description

maRketSim is a market simulator for R. It was initially designed around the bond market, with plans to expand to stocks. maRketSim is built around the idea of portfolios of fundamental objects. Therefore it is slow in its current incarnation, but allows you the flexibility of seeing exactly what is in your final results, since the objects are retained.

Details

<table>
<thead>
<tr>
<th>Package:</th>
<th>maRketSim</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>Package</td>
</tr>
<tr>
<td>Version:</td>
<td>1.0</td>
</tr>
<tr>
<td>Date:</td>
<td>2010-01-01</td>
</tr>
<tr>
<td>License:</td>
<td>What license is it under?</td>
</tr>
<tr>
<td>LazyLoad:</td>
<td>yes</td>
</tr>
</tbody>
</table>

See the demo file for a quick introduction.
Author(s)

Ari B. Friedman

Maintainer: Ari B. Friedman <abfriedman@gmail.com>

References

Fabozzi: Fixed Income Mathematics

Examples

```r
mkt1 <- market(market.bond(i=.05),t=0)
mkt2 <- market(market.bond(i=.07),t=2)
bnd <- bond(mkt=mkt1,mat=5)
summary(bnd,mkt=mkt2)
```

Description

Returns the proportion of the portfolio in each asset class

Usage

```r
aa(x, ...)
## Default S3 method:
aa(x, ...)
## S3 method for class 'account'
aa(x, sort = TRUE, force.cash = TRUE, ...)
```

Arguments

- `x`: Object, typically an account object.
- `sort`: Sort alphabetically by class name
- `force.cash`: If TRUE, add a cash asset class if none present in account.
- `...`: Pass-alongs.

Details

Return the proportion of the portfolio in each asset class

Value

Numeric vector of proportions
account  

Create fundamental market objects

Description

These functions create the fundamental objects from which everything else is built.

Usage

```
cash(value=0,mkt,compound="continuous",name=NA)
bond(mkt,mat=NA,par=1000,f=.5,dur=NA)
portfolio(bonds,mkt,name=NA)
account(prts, hist.mkt,t.issue=hist.mkt[[1]]$t)
```

Arguments

- **value**: Cash value to be stored
- **bonds**: List of bonds to include
- **compound**: "continuous" compounding or compounding rate
- **name**: Name of portfolio
- **mat**: Maturity (in units of t). Specify NA if you would rather create a bond of a particular duration
- **par**: Par value
- **f**: Frequency of coupons
- **dur**: Duration. If NA must specify maturity.
- **prts**: List of portfolios
- **mkt**: Market object
- **hist.mkt**: A history.market object
- **t.issue**: What time point was the account created at?

Value

Returns a market object
as.data.frame.bond  Returns a data-frame summary of object.

Description
Returns a data-frame summary of bond or portfolio.

Usage
```r
## S3 method for class 'bond'
as.data.frame(x, ...)
```

Arguments
- `x`: Bond or portfolio.bond object
- `...`: Pass-alongs.

cashflow  Functions to return cash-flows from bonds and portfolio objects

Description
These functions return the amounts and timings of cash flows from bond-type objects.

Usage
```r
cashflow(x, mkt, ...)
## Default S3 method:
cashflow(x, mkt, ...)
## S3 method for class 'cash'
cashflow(x, mkt, future=TRUE, ...)
## S3 method for class 'bond'
cashflow(x,mkt,future=TRUE,f=x$f,...)
## S3 method for class 'portfolio.bond'
cashflow(x,mkt=x$orig.mkt,sort=FALSE,condense=TRUE,future=TRUE,...)
## S3 method for class 'account'
cashflow(x, mkt, ...)
```

Arguments
- `x`: Bond or portfolio object
- `mkt`: Market object under which to evaluate cash flows
- `future`: TRUE/FALSE. Add in future=FALSE (past cash flows) for use with fv()
- `f`: Frequency of coupon payments
connectTheDots - Interpolate between points on a yield curve and return a maturity. Used for the quoted yield.curve statements. read.yield.curve - Function to take a data.frame of columns (i,mat,t) and turn it into a market.history with yield curves

### Usage

```r
connectTheDots(mat, df, constant.max.mat = TRUE, ...)
read.yield.curve(df, drop.if.no.MMrate=FALSE, MM.mat=1/12, ...)
```

### Arguments

- `mat` Maturity
- `df` Data.frame that contains yield curve information
- `constant.max.mat` Set to TRUE if you want it to return interest rates equal to the maximum maturity available if a mat is requested which exceeds the max maturity.
- `drop.if.no.MMrate` Whether or not to drop years with no money market rate
- `MM.mat` Consider times below this maturity to be money market
- `...` Pass-alongs

### Value

- `yield.curve object`
count.history.account  Function to return the number of objects of different types in a history.account object

Description
Function to return the number of objects of different types in a history.account object

Usage
count(x, ...)
## S3 method for class 'history.account'
count(x, type = "history.account", ...)

Arguments
x
A history.account object

Arguments
x
A history.account object

Arguments
t
which time to extract

Value
Numeric count

current.market  Find current market object

Description
Function to return the market object at a given time from a history.market object. If no time point
exists, interpolate by grabbing the previous market and returning a market object with those char-
acteristics but the current time.

Usage
current.market(hist.mkt, t, ...)

Arguments
hist.mkt
a history.market object

t
which time to extract

Arguments
x
A history.account object

Arguments
x
A history.account object

Arguments
t
which time to extract

Value
Numeric count
Value

A market object

---

current.portfolio.bond

*Get portfolio.bond at time t*

---

Description

Exclude bonds which have start time after current market time (e.g. which have not yet been purchased), and which have already matured

Usage

`current.portfolio.bond(object, mkt, drop.expired = TRUE, drop.future = TRUE, ...)`

Arguments

- **object**: portfolio.bond object
- **mkt**: Market under which to evaluate object
- **drop.expired**: Drop expired bonds from the returned object
- **drop.future**: Drop future bonds from the returned object
- **...**: Pass-alongs

Value

portfolio.bond object

---

duration

*Return the duration of a bond-type object*

---

Description

Return the duration of a bond or portfolio of bonds. Duration of cash is 0.
findDur

Usage

duration(x, type = "modified", ...)
## Default S3 method:
duration(x, type="modified", mkt, ...)
## S3 method for class 'cash'
duration(x, type="modified", mkt, ...)
## S3 method for class 'bond'
duration(x, type="modified", mkt, ...)
## S3 method for class 'portfolio.bond'
duration(x, type="modified", mkt, ...)
## S3 method for class 'account'
duration(x, type="modified", mkt, ...)
## S3 method for class 'history.account'
duration(x, type = "modified", ...)
## S3 method for class 'sum.account'
duration(x, type = "modified", ...)

Arguments

x A relevant maRketSim object (bond, portfolio.bond, etc.)
mkt A market object under whose interest rate you want to find the duration.
type Currently only option is "modified", where it returns the modified MacAulay duration
... Pass-alongs.

findDur Find duration/maturity given interest rate and maturity/duration

Description

Calculates the desired value given basic inputs. findDur_ClosedForm(), findMaxDur(), and findMaxMat are various closed-form approximations, whereas findDur and findMat are exact.

Usage

findDur(mat, i, market.rate = NA, f = 0.5, type = "modified", ...)
findMat(dur,i,f=.5)

Arguments

mat Maturity
dur Duration
i Interest rate
market.rate Market rate under which to evaluate mat/dur
f Frequency of coupon payments
For findDur: "modified" currently the only option. For findDur_ClosedForm: "pianca", "macaulay", or "hawawini"

... Pass-alongs.

Value

Returns a numeric value with the maturity/duration as described

---

**findPV**

*Find present/future value given simple inputs*

---

**Description**

Find present/future value by calculating

**Usage**

```python
findPV(i, mat, market.rate, par = 1000, f = 0.5, fractional.method = "30/360")
findFV(P, i, t.elapsed, compound="continuous")
```

**Arguments**

- **i**
  - Interest rate under which bond was purchased (the coupon rate)
- **mat**
  - Maturity
- **market.rate**
  - Market rate under which to evaluate value
- **par**
  - Par value of bond
- **f**
  - Frequency of coupon payments
- **fractional.method**
  - Currently only 30/360 convention is used
- **t.elapsed**
  - Time after which to find the future value
- **compound**
  - Currently only option is "continuous" compounding

**Value**

Returns a numeric valuation
**fund**

Create fund object

**Description**

Placeholder function. fund objects will eventually hold bonds, stocks, etc. and simulate mutual funds.

**Usage**

```r
fund(holdings, price.history, div.history, expense.ratio = 0, index = "", name = NA, ...)
```

**Arguments**

- `holdings` list of holdings of fund
- `price.history` Price history
- `div.history` Dividend history
- `expense.ratio` Expense ratio
- `index` The index followed, if any
- `name` Name of fund
- `...` Pass-alongs

**Value**

A fund object

---

**genHistory.market**

Generate a history.market by applying a quoted function to obtain the interest rates for each period

**Description**

Quickly create a history.market object according to a specified pattern

**Usage**

```r
genHistory.market(i.fxn, start.t, end.t, f, ...)
```

**Arguments**

- `i.fxn` Interest rate function. A quoted function of the current period t.
- `start.t` Start time
- `end.t` End time
- `f` Coupon frequency
- `...` Pass-alongs
genPortfolio.bond

Value

a history.market object

See Also

See Also history.market

---

**genPortfolio.bond**  
*Generate a portfolio of bond objects according to high-level criteria*

**Description**

This function creates a portfolio of bond objects from specification of inputs. Specifically, it allows you to specify the number of bonds and desired duration of the portfolio.

**Usage**

```r
genPortfolio.bond(n, mkt, dur,
  dur.sd, max.mat = 30, type = "random",
  f = 0.5, name = as.character(runif(1)),
  ...)```  

**Arguments**

- `n`: Number of bonds to create in portfolio
- `mkt`: Market conditions under which portfolio is created
- `dur`: Desired duration
- `dur.sd`: SD of bond durations
- `max.mat`: Limit to avoid huge bond lengths
- `type`: "random" for now
- `f`: Frequency of cash flows
- `...`: Pass-alongs.
- `name`: A name to give your portfolio

**Value**

Returns a portfolio object.
**history.account**

*history.account objects contain accounts over time*

---

**Description**

history.account objects store an account for each time period. Used by rebalance() and summary.account()

**Usage**

```r
history.account(accts, ...)
```

## S3 method for class 'history.account'

```r
as.data.frame(x, ...)

as.history.account(x, ...)
```

## S3 method for class 'sum.account'

```r
as.history.account(x, ...)
```

## S3 method for class 'history.account'

```r
print(x, ...)
```

**Arguments**

- `accts` List of account objects
- `x` history.account object or data.frame
- `...` Pass-alongs

**Value**

data.frame or history.account object

**See Also**

summary.account rebalance

---

**history.market**

*Creates a history.market object*

---

**Description**

A history.market object stores markets over time

**Usage**

```r
history.market(mkts, ...)
```
issueTime

Arguments

mkts A list of market objects
... Pass-alongs

Value

a history.market object

See Also

For automated market history creation, see genHistory.market.

Description

issueTime returns the object issue time of a MarketSim object. matTime returns the maturity time of a MarketSim object. mktTime returns the time embedded in a MarketSim market object.

Usage

issueTime(x, ...)
## S3 method for class 'bond'
issueTime(x, ...)
## Default S3 method:
issueTime(x, ...)
## S3 method for class 'portfolio.bond'
issueTime(x, ...)
mattime(x, ...)
## S3 method for class 'bond'
mattime(x, ...)
## Default S3 method:
mattime(x, ...)
## S3 method for class 'portfolio.bond'
mattime(x, ...)
## S3 method for class 'account'
mkttime(x, ...)
## Default S3 method:
mkttime(x, ...)
## S3 method for class 'history.account'
mkttime(x, ...)
## S3 method for class 'market'
mkttime(x, ...)
Arguments

- `x` A MarketSim object
- `...` Pass-alongs.

Value

A single numeric value representing the relevant time.

market

*Create a market object*

Description

Create a market object. Inside a market object must be one or more market-specific objects. Currently this means inside a market object must be a market.bond object holding timing and interest rate information. Eventually market objects will be able to hold market.stock objects as well.

Usage

```r
market(mkts, t=0)
market.bond(i = NA, yield.curve = NA, MMrate = NA, MM.frequency = 0.5)
```

Arguments

- `mkts` A list of market-type objects. Currently only market.bond exists. If there is only one market-type object to be passed, can omit the list wrapper.
- `i` Interest rate, as a proportion (e.g. if the yield is 1
- `yield.curve` Yield curve. If NA, use `i` instead (equivalent to a flat yield curve).
- `MMrate` Money Market rate (rate on short-term debt).
- `t` Time of market object.
- `MM.frequency` Frequency of money market evaluation

market.rate

*Return a market rate for a given maturity*

Description

Parses the yield curve in a market object and returns a market rate of a given maturity/duration

Usage

```r
market.rate(mkt, mat = NA, dur = NA, f = 0.5)
```
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mkt</td>
<td>Market object</td>
</tr>
<tr>
<td>mat</td>
<td>Maturity (needed to find interest rate on yield curve). If NA, specify duration instead.</td>
</tr>
<tr>
<td>dur</td>
<td>Duration (needed to find interest rate on yield curve). If NA, specify maturity instead.</td>
</tr>
<tr>
<td>f</td>
<td>Frequency of coupon payments</td>
</tr>
</tbody>
</table>

Value

Returns a simple numeric object with the market rate.

plot.history.market  

Description

plot various maRketSim objects

Usage

```r
## S3 method for class 'history.market'
plot(
  x, plot.MMrate = TRUE,
  plot.mats = c(1, 2, 5, 10),
  cols = rainbow(length(plot.mats)),
  start.t = x[[1]]$t,
  end.t = x[[length(x)]]$t + 0.5,
  xlab = "Year",
  ylab = "Interest Rate (%)",
  ...
)
## S3 method for class 'history.account'
plot(x,ylab="PV ($"),xlab="Time",...)  
## S3 method for class 'sum.account'
plot(x,which=c("pv","duration"),n.ticks=7,...)
## S3 method for class 'vasicek.discrete'
plot(x,t=ncol(x$prob.tree),...)
## S3 method for class 'sum.vasicek.discrete'
plot(x,...)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>maRketSim object</td>
</tr>
<tr>
<td>plot.MMrate</td>
<td>TRUE/FALSE whether to plot the money market rate or not</td>
</tr>
<tr>
<td>plot.mats</td>
<td>Maturities to plot</td>
</tr>
</tbody>
</table>
print.market

cols: color vector of length length(plot.mats)
start.t: Time to begin plotting at
end.t: Time to end plotting at
xlab: x label
ylab: y label
...: Pass-alongs to base graphics.
t: time vector
which: Which items of summary object to plot
n.ticks: How many ticks

Details

Defaults should be pretty sensible to give you a sense of the data embedded in an object.

Value

plot

Description

Display characteristics of marketSim object on console

Usage

## S3 method for class 'market'
print(x,...)
## S3 method for class 'sum.market.bond'
print(x,...)
## S3 method for class 'bond'
print(x,...)
## S3 method for class 'sum.bond'
print(x,...)
## S3 method for class 'cash'
print(x,...)
## S3 method for class 'account'
print(x,...)
## S3 method for class 'sum.account'
print(x,...)
## S3 method for class 'portfolio.bond'
print(x,...)
## S3 method for class 'sum.portfolio.bond'
print(x,...)
Arguments

\(x\) A MarketSim object

\[\ldots\] Pass-alongs.

---

\(\text{pv}\) \hspace{1em} \text{Return the present/future value of an object under a given market conditions}

Description

Returns the present/future value of a bond/portfolio/cash/account under specified market conditions. \(\text{pv.history.account}\) returns a vector unlike the other \(\text{pv}\) methods.

Usage

\(\text{pv}(x, \ldots)\)

---

\(\text{fv}(x, \ldots)\)

---

\(\text{pv}(x, \ldots)\)

## S3 method for class 'history.market'
print(x, ...)

## S3 method for class 'market.bond'
print(x, ...)

## S3 method for class 'cash'
\(\text{pv}(x, \ldots)\)

## S3 method for class 'bond'
\(\text{pv}(x, \text{mkt}, \ldots)\)

## S3 method for class 'account'
\(\text{pv}(x, \text{mkt}=x\$\text{orig.mkt}, \ldots)\)

## S3 method for class 'portfolio.bond'
\(\text{pv}(x, \text{mkt}=x\$\text{orig.mkt}, \ldots)\)

## S3 method for class 'history.account'
\(\text{pv}(x, \text{type}="\text{history.account"}, \ldots)\)

\(\text{fv}(x, \text{mkt}, \ldots)\)

## Default S3 method:
\(\text{fv}(x, \text{mkt}, \ldots)\)

## S3 method for class 'cash'
\(\text{fv}(x, \text{mkt}, \ldots)\)

## S3 method for class 'bond'
\(\text{fv}(x, \text{mkt}, \text{compound}="\text{continuous"}, \ldots)\)
Arguments

- **x**: A bond/portfolio/cash/account object
- **mkt**: The market under which the present value is to be evaluated
- **type**: Either "history.account", "bond", "portfolio.bond", or "cash". Determines output type.
- **compound**: Compounding type. Currently only option is "continuous".
- **...**: Pass-alongs.

---

**rebal.function.default**

*Example rebal.function for use by rebalance()*

---

**Description**

Advances time by one rebalancing period, buys/sells assets according to rebalancing rules, and returns a new account function. This example function should be adapted for your own use.

**Usage**

```r
eval.rebalance = function.default(  
  prts, start.mkt, end.mkt,  
  rebal.function.args = list(  
    min.bond.size = 1000,  
    new.bond.dur = numeric(),  
    new.bond.mat = 5,  
    sell.mat = numeric(),  
    sell.dur = numeric()  
  ),  
  ...  
)
```

**Arguments**

- **prts**: list of portfolio objects
- **start.mkt**: Starting time period’s market object
- **end.mkt**: Ending time period’s market object
- **rebal.function.args**: List of arguments passed from rebalance(). Can be whatever you like.
- **...**: Pass-alongs.

**Value**

- **account**

**See Also**

- **rebalance**
rebalance.account

Functions to rebalance accounts. Called by summary.account

Description

In general, you should use summary.account to call these functions. They rebalance an account according to rules specified in the given rebal.function, increment the time period by one, and track any interest payments and sale profits.

Usage

rebalance(obj,start.t,...)

## Default S3 method:

rebalance(obj,start.t,...)

## S3 method for class 'account'

rebalance(obj,
        start.t,
        rebal.function = rebal.function.default,
        f,
        rebal.function.args = list(),
        ...
)

Arguments

- **obj**: account object
- **rebal.function**: Function which actually handles the rebalancing. See rebal.function.default for an example, or write your own.
- **start.t**: Start time
- **f**: Frequency of coupon payments
- **rebal.function.args**: Arguments to pass to rebal.function, as a list
- **...**: Pass-alongs.

Details

Call from summary.account. These functions are at the heart of what makes this package useful.

Value

an account object

See Also

summary.account rebal.function.default
summary.bond

Evaluate maRketSim object under new market conditions

Description

The summary commands are key aspects of maRketSim, in that they evaluate objects under new market conditions.

Usage

```r
## S3 method for class 'bond'
summary(object, mkt, ...)
## S3 method for class 'portfolio.bond'
summary(object, mkt=object$orig.mkt, ...)
## S3 method for class 'market'
summary(object, ...)
## S3 method for class 'market.bond'
summary(object, i=NA, ...)
## S3 method for class 'account'
summary(
  object,
  t,
  rebal.function=rebal.function.default,
  return.history.account=TRUE,
  f=.5,
  rebal.function.args=list()
,...)
```

Arguments

- `object` A maRketSim object
- `mkt` A market object under which you want to evaluate object
- `t` Time under which to evaluate changes
- `i` the coupon rate of a non-mentioned bond you wish to analyze max duration under
- `rebal.function` Rebalancing function
- `return.history.account` If TRUE, return the entire history object (potentially quite large).
- `rebal.function.args` List of arguments to pass to rebal.function
- `f` Coupon frequency (in fraction of a unit of time, typically a year)
- `...` Pass alongs

See Also

rebal.function
sum.\textit{vasicek.discrete}  
\textit{Summarize a Vasicek model}

### Description

Summarize a Vasicek model

### Usage

```r
## S3 method for class 'vasicek.discrete'
summary(object, lambda, ...)
```

### Arguments

- \textit{object} \hspace{1cm} \textit{vasicek.discrete} object
- \textit{lambda} \hspace{1cm} Lambda is a measure of excess return per unit of standard deviation (risk-reward relationship), and is constant across all terms
- ... \hspace{1cm} Pass-alongs

### Value

- \textit{sum.\textit{vasicek.discrete}} object

---

\textit{\textbf{vasicek\_discrete}}  
\textit{Vasicek yield model in discrete time}

### Description

Classic Vasicek model

### Usage

```r
\textit{vasicek\_discrete}(mu, sigma, h, phi, r\_initial, N)
```

### Arguments

- \textit{mu} \hspace{1cm} what the short term rate converges to
- \textit{sigma} \hspace{1cm} SD of the forecast (determines step size)
- \textit{h} \hspace{1cm} time per period
- \textit{phi} \hspace{1cm} determines mean reversion
- \textit{r\_initial} \hspace{1cm} initial rate
- \textit{N} \hspace{1cm} number of periods total to run the simulation

### Value

- \textit{\textit{vasicek.discrete}} object
VFITX_div

VFITX divindends

Usage

data(VFITX_div)

Format

A data frame with 177 observations on the following 2 variables.

Date   Date, as a factor
Dividends  a numeric vector

Details

Dividend history only. See demo file for details.

Source

Yahoo

Examples

data(VFITX_div)
div.df <- VFITX_div
div.df$Date <- as.Date(div.df$Date)
div.df$month <- as.numeric(format(div.df$Date,format="%m"))
div.df$year <- as.numeric(format(div.df$Date,format="%Y")) # Aggregate into 6-month dividends
div.df$month[div.df$month>=1&div.df$month<=6] <- 6
div.df$month[div.df$month>=7&div.df$month<=12] <- 12
by.res <- by(div.df,list(div.df$month,div.df$year),
function(x) {
  return(data.frame(div=sum(x$Dividends),N=length(x$Dividends)))
})
div.df <- by.res[[1]]
for(i in seq(2,length(by.res))) {
  if(!is.null(by.res[[i]])) {
    div.df <- rbind(div.df,by.res[[i]])
  }
}
div.df$month <- as.numeric(rep(dimnames(by.res)[[1]],length(dimnames(by.res)[[2]])))
div.df$year <- as.numeric(rep(dimnames(by.res)[[2]],each=length(dimnames(by.res)[[1]])))
div.df <- subset(div.df,N>6,select=c(-N)) # Exclude partial half-years
VFITX_prices

Vanguard Intermediate Treasury Data, as downloaded from Yahoo

Description
VFITX prices

Usage
data(VFITX_prices)

Format
A data frame with 3626 observations on the following 7 variables.

Date  date, as a factor
Open  a numeric vector
High  a numeric vector
Low   a numeric vector
Close  a numeric vector
Volume a numeric vector
Adj.Close a numeric vector

Details
Price history only. See demo file for details.

Source
Yahoo

Examples
data(VFITX_prices)
pr.df <- VFITX_prices
pr.df$Date <- as.Date(pr.df$Date)
pr.df$month <- format(pr.df$Date, format="%m")
pr.df <- subset(pr.df, month="06", select=c("Date","Close","month"))
pr.df$month <- as.numeric(pr.df$month)
pr.df$year <- as.numeric(format(pr.df$Date, format="%Y"))
pr.df$day <- as.numeric(format(pr.df$Date, format="%d"))
# Select last available day of each month
by.res <- by(pr.df, list(pr.df$month, pr.df$year), function(x) x[x$day==max(x$day),])
pr.df <- rbind(pr.df, by.res[[1]])
for(i in seq(2, length(by.res))) {
  if(!is.null(by.res[[i]])) {
    pr.df <- rbind(pr.df, by.res[[i]])
  }
VFITX_prices

```r
pr.df <- subset(pr.df, select=c("Close","month","year"))
names(pr.df)[names(pr.df)=="Close"] <- "p"
```
Index

*Topic datasets
  VFITX_div, 23
  VFITX_prices, 24

*Topic package
  marKetSim-package, 2

aa, 3
account, 4
as.data.frame.bond, 5
as.data.frame.history.account
  (history.account), 13
as.data.frame.portfolio.bond
  (as.data.frame.bond), 5
as.history.account (history.account), 13
bond (account), 4
cash (account), 4
cashflow, 5
connectTheDots, 6
count (count.history.account), 7
count.history.account, 7
current.market, 7
current.portfolio.bond, 8
duration, 8
findDur, 9
findDur_ClosedForm (findDur), 9
findFV (findPV), 10
findMat (findDur), 9
findMaxDur (findDur), 9
findMaxMat (findDur), 9
findPV, 10
fund, 11
fv (pv), 18
genHistory.market, 11, 14
genPortfolio.bond, 12
history.account, 13
history.market, 12, 13
issueTime, 14
market, 15
market.rate, 15
marKetSim (marKetSim-package), 2
marKetSim-package, 2
matTime (issueTime), 14
mktTime (issueTime), 14
plot.history.account
  (plot.history.market), 16
plot.history.market, 16
plot.sum.account (plot.history.market),
  16
plot.sum.vasicek.discrete
  (plot.history.market), 16
plot.vasicek.discrete
  (plot.history.market), 16
portfolio (account), 4
print.account (print.market), 17
print.bond (print.market), 17
print.cash (print.market), 17
print.history.account
  (history.account), 13
print.history.market (print.market), 17
print.market, 17
print.portfolio.bond (print.market), 17
print.sum.account (print.market), 17
print.sum.bond (print.market), 17
print.sum.market.bond (print.market), 17
print.sum.portfolio.bond
  (print.market), 17
pv, 18
read.yield.curve (connectTheDots), 6
rebal.function.default, 19, 20
rebalance, 13, 19
rebalance (rebalance.account), 20
INDEX

rebalance.account, 20
summary.account, 13, 20
summary.account (summary.bond), 21
summary.bond, 21
summary.market (summary.bond), 21
summary.portfolio.bond (summary.bond), 21
summary.vasicek.discrete, 22
vasicek.discrete, 22
VFITX_div, 23
VFITX_prices, 24