

Package ‘tradeCosts’

April 19, 2009

Title Trade Cost Analysis

Version 0.3-0

Date 2008-04-17

Author Arjun Ravi Narayan <contact@arjunnarayan.com>, Aaron Schwartz <Aaron.J.Schwartz@williams.edu>, Daniel Suo <danielsuo@gmail.com> and Luyi Zhao <luyizhao@fas.harvard.edu> with contributions from Jeff Enos <jeff@kanecap.com> and David Kane <david@kanecap.com>

Maintainer Jeff Enos <jeff@kanecap.com>

LazyLoad yes

Description Performs ex-post analysis of security transaction costs.

Depends R (>= 2.4.0), methods, tools, graphics

License GPL (>= 2)

Repository CRAN

Date/Publication 2008-04-21 06:38:03

R topics documented:

tradeCosts-package	2
create.dynamic.data	2
create.trade.data	3
dynamic.mar.2007	4
static.mar.2007	5
trade.costs	5
trade.mar.2007	7
tradeCosts-class	7
tradeCostsResults-class	8

Index	10
--------------	-----------

 tradeCosts-package *Trade Cost Analysis*

Description

Performs ex-post analysis of security transaction costs.

Details

Package: tradeCosts
 Version: 0.2-1
 Date: 2007-10-01
 LazyLoad: yes
 Depends: R (>= 2.4.0), methods, tools, graphics
 License: GPL (>= 2)
 Built: R 2.6.0; ; 2007-10-02 09:09:18; unix

Index:

dynamic.mar.2007	Sample dynamic descriptive data
static.mar.2007	Sample static descriptive data
trade.costs	User-level function to perform basic trade cost analysis.
trade.mar.2007	Sample trading data
tradeCosts-class	Class "tradeCosts"
tradeCosts-package	Trade Cost Analysis
tradeCostsResults-class	Class "tradeCostsResults"

Further information is available in the following vignettes:

tradeCosts-article	Trade Costs (source, pdf)
tradeCosts	Using the tradeCosts package (source, pdf)

Author(s)

Arjun Ravi Narayan <contact@arjunnarayan.com>, Aaron Schwartz <Aaron.J.Schwartz@williams.edu>, Daniel Suo <danielsuo@gmail.com> and Luyi Zhao <luyizhao@fas.harvard.edu> with contributions from Jeff Enos <jeff@kanecap.com> and David Kane <david@kanecap.com>
 Maintainer: Jeff Enos <jeff@kanecap.com>

 create.dynamic.data

User-level function to create a dynamic dataframe for use in trade.costs

Description

This function requires two vectors: period, id. It combines these two vectors and optional vectors into a single data.frame ready for use in the trade.costs function.

Usage

```
create.dynamic.data(period, id, ...)
```

Arguments

period	The column of periods in which trades occurred.
id	The column of security identifiers traded.
...	Additional vector arguments of the same length to be incorporated into the resulting data frame

Details

This helper function makes creating a dynamic data.frame easier.

Value

A data.frame is returned for use in the trade.costs function.

Author(s)

Daniel Suo

Examples

```
data(dynamic.mar.2007)
create.dynamic.data(dynamic.mar.2007$period, dynamic.mar.2007$id,
dynamic.mar.2007$vwap)
```

create.trade.data *User-level function to create a trade data.frame for use in trade.costs*

Description

This function requires five vectors: period, id, side, exec.qty, and exec.price. It combines these five vectors and optional vectors into a single data.frame ready for use in the trade.costs function.

Usage

```
create.trade.data(period, id, side, exec.qty, exec.price, ...)
```

Arguments

<code>period</code>	The column of periods in which trades occurred.
<code>id</code>	The column of security identifiers traded.
<code>side</code>	What side the security was traded in.
<code>exec.qty</code>	How much was traded.
<code>exec.price</code>	The price per share of the trade.
<code>...</code>	Additional vector arguments of the same length to be incorporated into the resulting data frame

Details

This helper function makes creating a `trade.data.frame` easier.

Value

A `data.frame` is returned for use in the `trade.costs` function.

Author(s)

Daniel Suo

Examples

```
data(trade.mar.2007)
create.trade.data(trade.mar.2007$period, trade.mar.2007$id,
  trade.mar.2007$side, trade.mar.2007$exec.qty, trade.mar.2007$exec.price)
```

`dynamic.mar.2007` *Sample dynamic descriptive data*

Description

Sample dynamic descriptive data, or data that changes over time such as price, for a set of securities.

Usage

```
data(dynamic.mar.2007)
```

Format

A data frame with 1688 observations on the following 6 variables.

period a Date

id a character vector containing a unique security id

price a numeric vector containing the period's price

vwap a numeric vector containing the volume-weighted average price for the period

prior.close a numeric vector specifying the prior closing price for the period

adjustment.factor a numeric vector

Examples

```
data(dynamic.mar.2007)
```

```
static.mar.2007
```

Sample static descriptive data

Description

Sample static descriptive data, or data that doesn't change over time, for a set of securities.

Usage

```
data(static.mar.2007)
```

Format

A data frame with 4091 observations on the following 4 variables.

id a character vector specifying a unique security id

symbol a character vector containing the security symbol

name a character vector containing company name

sector a character vector containing company sector

Examples

```
data(static.mar.2007)
```

```
trade.costs
```

User-level function to perform basic trade cost analysis.

Description

This function requires three data sets (dynamic descriptive, static descriptive, and trading data). It also requires the names of important columns related to the trade cost analysis.

Usage

```
trade.costs(trade, dynamic, static = NULL, decisions = NULL,  
            benchmark.price = "vwap", base.price = "exec.price", num.trades = 5,  
            analysis.title = "Trade Cost Analysis", batch.method = "unique",  
            start.period = NULL, end.period = NULL)
```

Arguments

<code>dynamic</code>	The <code>dynamic</code> argument is for dynamic descriptive data of securities and should include some notion of time period and price.
<code>trade</code>	The <code>trade</code> argument should be a data set with trading data including at least one type of price related to the execution of a trade as well as a notion of period.
<code>static</code>	The <code>static</code> argument is an optional argument with defaults <code>NULL</code> that provides descriptive data for securities, should include basic information for each security.
<code>decisions</code>	The <code>decisions</code> argument is an optional argument that can provide decision data for calculating implementation shortfall.
<code>benchmark.price</code>	The <code>benchmark.price</code> argument should be the name of a column in <code>dyna</code> and is the price that will be used for benchmarking and calculating trade cost statistics such as slippage.
<code>base.price</code>	The <code>base.price</code> argument should be the name of a column in <code>dyna</code> and is the price that will be used as the base price for calculating trade cost statistics such as slippage.
<code>num.trades</code>	The <code>num.trades</code> argument is a numeric argument that specifies how many trades should be shown in the resulting summaries. The default is 5 trades, which means for each category, the summary shows the 5 best and worst trades.
<code>analysis.title</code>	User-specified character string which serves as the title of the report for aesthetic purposes.
<code>batch.method</code>	Option for selecting method for grouping trades.
<code>start.period</code>	Object of the same class as the <code>period</code> column in the <code>trade</code> and <code>static</code> data frames that specifies the first period of the analysis.
<code>end.period</code>	Object of the same class as the <code>period</code> column in the <code>trade</code> and <code>static</code> data frames that specifies the last period of the analysis.

Details

This version of `trade.costs` provides a single multi-period sample batch method, `same.sided`, which groups all consecutive same-sided orders into a single batch. The default `batch.method`, `unique`, places each trade in its own batch.

Value

A `tradeCostsResults` object is returned.

Author(s)

Aaron Schwartz and Luyi Zhao

Examples

```
data(dynamic.mar.2007)
data(trade.mar.2007)
data(static.mar.2007)
trade.costs(trade.mar.2007, dynamic.mar.2007,
            static.mar.2007, num.trades = 3)
```

trade.mar.2007 *Sample trading data*

Description

Sample trading data from March, 2007.

Usage

```
data(trade.mar.2007)
```

Format

A data frame with 413 observations on the following 5 variables.

period a Date specifying the period of the trade

id a character vector containing a unique security id

side a character vector specifying type of trade. Takes on values B (buy), S (sell), C (cover) and X (short sell)

exec.qty a numeric vector specifying the size of the trade

exec.price a numeric vector specifying the execution price

Examples

```
data(trade.mar.2007)
```

tradeCosts-class *Class "tradeCosts"*

Description

This is an object of tradeCosts class constructed so trade cost analysis can be performed.

Slots

name: Object of class "character" containing the name of this "tradeCosts" object.

trade.data: Object of class "data.frame" with trade information

static.desc: Object of class "data.frame" with static descriptive security information.

dynamic.desc: Object of class "data.frame" with dynamic descriptive security information.

decisions.data: Object of class "data.frame" with decisions data.

Methods

analyzeData signature(object = "tradeCosts"). Merges the data, does NA reporting, and calls private functions that calculate trade costs.

Author(s)

Aaron Schwartz and Luyi Zhao and Arjun Ravi Narayan

```
tradeCostsResults-class
      Class "tradeCostsResults"
```

Description

This function constructs a basic trade cost analysis results object.

Slots

name: Object of class "character" containing the name of this trade cost analysis

results: Object of class "data.frame" that contains the raw trade cost results

na.counts: Objects of class "data.frame" that contains the raw trade cost results

base.price: Object of class "character" that explains what price is being used as the base price.

benchmark.price: Object of class "character" that explains what price is being used as the benchmark price.

batch.method: Object of class "character" that explains in what way to group the executions for analysis.

Methods

summary signature(object = "tradeCostsResults"): Provides a printout summarizing the tradeCostsResults object.

pdfReport signature(object = "tradeCostsResults"): Provides a pdf summary of the tradeCostsResults object.

- calcAll** signature(results = "data.frame", num.trades = "numeric"): Calculates best/worst trades for all id/date pairs.
- calcID** signature(results = "data.frame", num.trades = "numeric"): Calculates best/worst trades for ID's.
- calcPeriod** signature(results = "data.frame", num.trades = "numeric"): Calculates best/worst trades over periods.
- calcStats** signature(results = "data.frame"): Calculates summary statistics for the tradeCostsResults object.
- plot** signature(x = "tradeCostsResults"): Plots the tradeCostsResults object with the argument 'type' as one of "time.series" (this is the default), "time.series.bps", "cumulative", or any column name in the static data frame

Author(s)

Aaron Schwartz and Luyi Zhao and Arjun Ravi Narayan

Index

*Topic **classes**

tradeCosts-class, 7
tradeCostsResults-class, 8

*Topic **datasets**

dynamic.mar.2007, 4
static.mar.2007, 5
trade.mar.2007, 7

*Topic **file**

create.dynamic.data, 2
create.trade.data, 3
trade.costs, 5

*Topic **package**

tradeCosts-package, 1

analyzeData (tradeCosts-class), 7
analyzeData, tradeCosts-method
(tradeCosts-class), 7

calc (tradeCostsResults-class), 8
calc, tradeCostsResults, character-method
(tradeCostsResults-class),
8

calcAll
(tradeCostsResults-class),
8

calcAll, data.frame, numeric-method
(tradeCostsResults-class),
8

calcID (tradeCostsResults-class),
8

calcID, data.frame, numeric-method
(tradeCostsResults-class),
8

calcPeriod
(tradeCostsResults-class),
8

calcPeriod, data.frame, numeric-method
(tradeCostsResults-class),
8

calcStats
(tradeCostsResults-class),
8

calcStats, data.frame-method
(tradeCostsResults-class),
8

create.dynamic.data, 2
create.trade.data, 3

dynamic.mar.2007, 4

extremeTrades, tradeCostsResults, numeric, character
(tradeCostsResults-class),
8

pdfReport
(tradeCostsResults-class),
8

pdfReport, tradeCostsResults-method
(tradeCostsResults-class),
8

plot, tradeCostsResults, character-method
(tradeCostsResults-class),
8

show, tradeCostsResults-method
(tradeCostsResults-class),
8

static.mar.2007, 5

summary, tradeCostsResults-method
(tradeCostsResults-class),
8

trade.costs, 5

trade.mar.2007, 7

tradeCosts (tradeCosts-package), 1
tradeCosts-class, 7

tradeCosts-package, 1

tradeCostsResults-class, 8