# Package 'eventstudies'

June 29, 2017

**Version** 1.2

**Date** 2017-06-28

**Type** Package

**Title** Event Study Analysis

**Maintainer** Chirag Anand <anand.chirag@gmail.com>

**Depends** R (>= 3.4), zoo, xts

**Description** An R package for conducting event studies and a platform
for methodological research on event studies.

**VignetteBuilder** knitr

**Suggests** knitr

**Imports** boot, graphics, methods, testthat, sandwich, stats, utils

**License** GPL-2

**URL** https://github.com/nipfpmf/eventstudies

**BugReports** https://github.com/nipfpmf/eventstudies/issues/new

**LazyLoad** yes

**NeedsCompilation** no

**Author** Chirag Anand [aut, cre],
Vikram Bahure [aut],
Vimal Balasubramaniam [aut],
Shekhar Harikumar [ctb],
Sargam Jain [ctb],
Ajay Shah [aut]

**Repository** CRAN

**Date/Publication** 2017-06-29 12:14:13 UTC

## R topics documented:


eventstudies-package . . . . . 2
AggregateReturns . . . . . 3

constantMeanReturn . . . 3
eesDates . . . 4
eesInference . . . 6
eesSummary . . . 7
eventstudy . . . 9
excessReturn . . . 13
get.clusters.formatted . . . 14
IndexReturns . . . 16
inference.bootstrap . . . 16
inference.classic . . . 17
inference.wilcox . . . 18
KGEarningsDates . . . 20
KGMarketReturns . . . 20
KGStockReturns . . . 21
KGSurpriseCategory . . . 21
lmAMM . . . 22
makeX . . . 23
manyfirmssubperiod.lmAMM . . . 25
marketModel . . . 26
OtherReturns . . . 27
phys2eventtime . . . 28
RateCuts . . . 29
remap.cumprod . . . 30
remap.cumsum . . . 31
remap.event.reindex . . . 32
SplitDates . . . 33
StockPriceReturns . . . 33
subperiod.lmAMM . . . 34
TerrorAttack . . . 35
TerrorIndiceReturns . . . 36

**Index** **37**

---

`eventstudies-package` *eventstudies: R package for conducting event studies*

---

## Description

**eventstudies** is an R package for conducting event studies and a platform for methodological research on event studies.

## Details

| | |
|---|---|
| Package: | eventstudies |
| Type: | Package |
| Version: | 1.1 |
| Date: | 2014-03-28 |

License: GPL 2
LazyLoad: yes

This package allows a dataset to be studied in an event-time frame and perform parametric / non-parametric analysis using several inference procedures.

## Author(s)

Chirag Anand, Vikram Bahure, Vimal Balasubramaniam, Ajay Shah

Maintainer: Vikram Bahure <economics.vikram@gmail.com>

---

`AggregateReturns` *Intra-day stock price returns data*

---

## Description

This data set contains intra-day stock price returns (in per cent) of 7 Indian banks with highest weight in the Bank NIFTY Index. The intra-day stock returns for 35 minutes before and after the event are used to define the event window.

## Usage

```
data(AggregateReturns)
```

## Author(s)

Sargam Jain

---

`constantMeanReturn` *Extract residuals over constant mean returns*

---

## Description

This function computes constant mean return during the estimation period prior to the defined event window. If the firm return is "firm.returns", then output will be "firm.returns" during the event period less the constant mean return computed over the estimation period.

## Usage

```
constantMeanReturn(firm.returns, residuals = TRUE)
```

### Arguments

| | |
|---|---|
| firm.returns | a **zoo** timeseries of firm returns from which constant mean return is computed over the estimation period. |
| residuals | a 'logical' indicating whether to return residuals or 'constant mean'. When argument to the function includes the entire time series, returns are estimated using the entire data set and not just estimation period, value of residuals should be TRUE in such a case. |

### Value

Residual returns unexplained by constant mean returns

### Author(s)

Sargam Jain

### Examples

```
data(StockPriceReturns)
data(SplitDates)
cmr.result <- constantMeanReturn(firm.returns = StockPriceReturns,
                                 residuals = TRUE)

Comparison <- merge(meanAdjustedReturns = cmr.result$Infosys,
                    Infosys = StockPriceReturns$Infosys,
                    all = FALSE)
plot(Comparison)
```

---

## eesDates *Get event list for extreme event study analysis*

---

### Description

This function creates event list (clustered and unclustered events) for extreme event study analysis.

### Usage

```
eesDates(input)
```

### Arguments

| | |
|---|---|
| input | object returned by 'get.clusters.formatted' |

## Details

The function creates a list of interesting events extracted from the output of `get.clusters.formatted`. The event list can be directly supplied to the `eventstudy` function.

It returns extreme right tail and left tail event dates for clustered and unclustered data. The 'normal' set consists of event dates for only unclustered events and 'purged' set consists of event dates for unclustered and clustered both. Unclustered events consist of clean event window with no event occurring in the event window and clustered events are fused consecutive events which lie in the same tail.

## Value

A `list` object containing:

`events.good.normal`
: 'data.frame' containing right tail event dates of unclustered events.

`events.bad.normal`
: 'data.frame' containing left tail event dates of unclustered events.

`events.good.purged`
: 'data.frame' containing right tail event dates of unclustered events and unclustered events.

`events.bad.purged`
: 'data.frame' containing left tail event dates of unclustered and clustered events.

## Author(s)

Vikram Bahure, Chirag Anand

## References

*Ila Patnaik, Nirvikar Singh and Ajay Shah (2013). Foreign Investors under stress: Evidence from India. International Finance, 16(2), 213-244.* http://onlinelibrary.wiley.com/doi/10.1111/j.1468-2362.2013.12032.x/abstract http://macrofinance.nipfp.org.in/releases/PatnaikShahSingh2013_Foreign_Investors.html

## Examples

```
data(OtherReturns)

input <- get.clusters.formatted(event.series = OtherReturns[, "SP500"],
                        response.series = OtherReturns[, "NiftyIndex"])

eventlist <- eesDates(input)
str(eventlist, max.level = 2)
```

`eesInference` *Extreme event study inference estimation*

## Description

This function performs event study analysis on extreme event dates ('eesDates') and using formatted output ('get.clusters.formatted')

## Usage

```
eesInference(input, event.lists, event.window, to.remap = TRUE, remap = "cumsum",
  inference = "TRUE", inference.strategy = "bootstrap")
```

## Arguments

| | |
|---|---|
| `input` | a formatted cluster object, as returned by 'get.clusters.formatted' function. |
| `event.lists` | a 'list' of normal and purged events as returned by 'eesDates'. |
| `event.window` | an 'integer' of length 1 that specifies a symmetric event window around the event date. |
| `to.remap` | 'logical', indicating whether or not to remap the data in 'input'.The default setting is 'TRUE' |
| `remap` | 'character', indicating the type of remap required, "cumsum", "cumprod", or "reindex". Used when 'to.remap' is 'TRUE'. |
| `inference` | 'logical', specifying whether to undertake statistical inference and compute confidence intervals. The default setting is 'TRUE'. |
| `inference.strategy` | a 'character' scalar specifying the inference strategy to be used for estimating the confidence interval. Presently, two methods are available: "bootstrap" and "wilcox". The default setting is 'bootstrap'. |

## Details

This function performs event study analysis using `eventstudy` function on the extreme event dates of normal (unclustered events) and purged (clustered and unclustered events) sets. These interesting dates are obtained from function 'eesDates'. The function can estimate confidence interval using different inference strategies as provided by `eventstudy()`.

The function does not do market model adjustment but takes the output of `get.clusters.formatted` as it's input.

## Value

Format of event study output is a 'matrix' containing mean or median estimate with confidence interval; 'NULL' if there are no "success" "outcomes". See `phys2eventtime` for more details.

A 'list' with class attribute "ees" holding the following four event study output elements:

| | |
|---|---|
| good.normal | an event study inference 'matrix' for right tail unclustered events, termed as normal |
| bad.normal | an event study inference 'matrix' for left tail unclustered events, termed as normal |
| good.purged | an event study inference 'matrix' for right tail clustered and unclustered events, termed as purged |
| bad.purged | an event study inference 'matrix' for left tail clustered and unclustered events, termed as purged |

## Author(s)

Vikram Bahure, Chirag Anand

## References

*Ila Patnaik, Nirvikar Singh and Ajay Shah (2013). Foreign Investors under stress: Evidence from India. International Finance, 16(2), 213-244.* http://onlinelibrary.wiley.com/doi/10.1111/j.1468-2362.2013.12032.x/abstract http://macrofinance.nipfp.org.in/releases/PatnaikShahSingh2013_Foreign_Investors.html

## Examples

```
data(OtherReturns)

formattedClusters <- get.clusters.formatted(event.series = OtherReturns[, "SP500"],
                                            response.series = OtherReturns[, "NiftyIndex"])

event.lists <- eesDates(formattedClusters)

inference <- eesInference(input = formattedClusters,
                          event.lists = event.lists,
                          event.window = 5)
str(inference, max.level = 2)
```

---

# eesSummary *Summary statistics of extreme events*

---

## Description

This function generates summary statistics for identification and analysis of extreme events.

## Usage

```
eesSummary(input)
```

## Arguments

| | |
|---|---|
| input | object returned by 'get.clusters.formatted' |

## Details

This function generates summary statistics of extreme events, using the tail events as returned by the function 'get.clusters.formatted'.

Following statistics are generated for both lower and upper tail events:

- 'extreme.event.distribution' provides summary statistics on the number of consecutive events ("clustered" events) and those that are not ("unclustered" events). Consecutive events that are "mixed", i.e., with upper (lower) tail event occurring after a lower (upper) tail event, are classified separately.
- 'runlength': When events are "clustered", 'runlength' classifies such clusters into different duration bins.
- 'quantile.values': Within such events, 'quantile.values' provide the probability distribution values at 0%, 25%, 50%, 75% and 100%, alongside the mean.
- 'yearly.extreme.event': A year-wise tabulation of such extreme events, with a clustered event taken as one event.

## Value

A `list` object containing:

| | |
|---|---|
| `data.summary` | a 'data.frame' containing summary of the minimum, maximum, inter-quartile range, mean, median, standard deviation and quantile values at 5%, 25%, 75% and 95%. |
| `lower.tail` | a 'list' that contains 'extreme.event.distribution', 'runlength', 'quantile.values' and 'yearly.extreme.event' for the events on the lower tail of the distribution. See 'Details'. |
| `upper.tail` | a 'list' that contains 'extreme.event.distribution', 'runlength', 'quantile.values' and 'yearly.extreme.event' for the events on the upper tail of the distribution. See 'Details'. |

## Author(s)

Vikram Bahure, Vimal Balasubramaniam

## References

*Ila Patnaik, Nirvikar Singh and Ajay Shah (2013). Foreign Investors under stress: Evidence from India. International Finance, 16(2), 213-244.* http://onlinelibrary.wiley.com/doi/10.1111/j.1468-2362.2013.12032.x/abstract http://macrofinance.nipfp.org.in/releases/PatnaikShahSingh2013_Foreign_Investors.html

## Examples

```
data(OtherReturns)

formattedClusters <- get.clusters.formatted(event.series = OtherReturns[, "SP500"],
                                            response.series = OtherReturns[, "NiftyIndex"])
```

```
ees.summary.tables <- eesSummary(formattedClusters)
str(ees.summary.tables, max.level = 2)
```

---

`eventstudy` *Perform event study analysis*

---

## Description

'eventstudy' provides an easy interface that integrates all functionalities of package **eventstudies** to undertake event study analysis. It allows the user to specify the type of data adjustment to be done (using market model functionalities of the package) and then an inference strategy of choice.

## Usage

```
eventstudy(firm.returns,
           event.list,
           event.window = 10,
           is.levels =  FALSE,
           type = "marketModel",
           to.remap = TRUE,
           remap = "cumsum",
           inference = TRUE,
           inference.strategy = "bootstrap",
           model.args = NULL)
```

## Arguments

| | |
|---|---|
| `firm.returns` | a **zoo** matrix of 'outcome' or 'response' series. |
| `event.list` | a `data.frame` of two columns with event dates (colname: "when") and column names of the 'response' series from 'firm.returns' (colname "name"). |
| `event.window` | an 'integer' of length 1 that specifies a symmetric event window around the event time as specified in the `index` of "firm.returns". |
| `type` | a scalar of type 'character' specifying the type of data adjustment required before conducting an event study analysis. See 'Details'. |
| `to.remap` | 'logical', indicating whether or not to remap the data in 'firm.returns'. |
| `remap` | 'character', indicating the type of remap required, "cumsum", "cumprod", or "reindex". Used when 'to.remap' is 'TRUE'. |
| `is.levels` | 'logical', indicating whether data in 'firm.returns' needs to be converted into percentage returns. If 'TRUE', 'firm.returns' will be converted into percentage returns. |
| `inference` | 'logical', specifying whether to undertake statistical inference and compute confidence intervals. The default setting is 'TRUE'. |
| `inference.strategy` | a 'character' scalar specifying the inference strategy to be used for estimating the confidence interval. Presently, two methods are available: "bootstrap" and "wilcox". |

`model.args` All other arguments to be passed depends on whether 'type' is "marketModel", "excessReturn", or "lmAMM". When "None", no additional arguments will be needed. See 'Details'.

## Details

This function is used to conduct event study analysis acting as a wrapper over the functionality provided in the **eventstudies** package. It provides an interface to select and control the process of event study analysis. It includes choice of the statistical model for doing in-sample estimation and computing coefficients, choice of cumulative returns, and selection of inference procedure. Process used to conduct a study is detailed below:

1. event.period: is defined as (-event.window, event.window].
2. estimation.period: If "type" is specified, then estimation.period is calculated for each firm-event in "event.list", starting from the start of the data span till the start of event period (inclusive).
3. For each firm-event, firm returns and other returns (as applicable) are converted to event time using 'phys2eventtime'. Data is merged using 'merge.zoo' to make sure the indexes are consistent before conversion to event time.
4. The selected model "type" is run on the series indexed by event time and abnormal returns are computed.
5. NULL values because of estimation data missing are removed from the output and "outcomes" object is updated with "edatamissing".
6. Remapping is done if "to.remap" is 'TRUE' using the function specified in "remap" argument.
7. Means of returns are computed across various events.
8. Inference is done if "inference" is 'TRUE' using the technique specified in "inference.strategy".

"firm.returns" can contain a single series also. To study a single series, use '[' with `drop = FALSE` to subset the data set. See `phys2eventtime` for more details.

'NA' values in the returns data are converted to `0`.

"type" currently supports:

- "marketModel": uses `marketModel` function to regress market returns on firms return using a linear model.
- "excessReturn": uses `excessReturn` to subtract market return from firm return.
- "lmAMM": uses `lmAMM` to perform Augmented Market Model estimation.
- "None": does not use any model.

Arguments to a model type can be sent inside 'model.args'. See 'Model arguments' section for details on accepted fields.

"remap" can take three values:

- "cumsum": cumulative sum, uses `remap.cumsum`. [Default]
- "cumprod": cumulative product, buy-hold-abnormal-return (BHAR), uses `remap.cumprod`.
- "reindex": re-indexes the event window by using `remap.event.reindex`.

For computing confidence intervals, the function can either use bootstrap or Wilcoxon signed-rank test. See `inference.bootstrap` and `inference.wilcox` for more details.

'model.args' is directly supplied to the model mentioned in the "type" argument. See section on 'Model arguments' for more details.

Note: `phys2eventtime` is called with 'width' set to 0 when called from this function.

## Value

A list with class attribute "es" holding the following elements, or 'NULL' if output from a model function is 'NULL':

- "eventstudy.output": a 'matrix' containing mean (bootstrap) or median (with wilcox) estimate with confidence interval; 'NULL' if there are no "success" "outcomes".
- "outcomes": a character vector that is the output from `phys2eventtime` containing details of the successful use of an event:
  - success: shows the successful use of event date.
  - wdatamissing: appears when width data is missing around the event. This will not appear when this function is used since the argument 'width' in `phys2eventtime` is set to 0.
  - wrongspan: if event date falls outside the range of physical date.
  - unitmissing: when the unit (firm name) is missing in the event list.
  - edatamissing: when there is insufficient data to do model estimation.

The returned object contains input information in other attributes:

- "model.residuals": a 'list' of residual series as returned by the selected model. For models which do not compute residuals, this attribute is not returned.
- "CAR": a 'zoo' object contaning Cumulative Abnormal Returns as returned by the function specified by "remap" argument.
- "inference": a 'character' providing information about which inference strategy was utilised to estimate the confidence intervals.
- "inference.strategy": a 'character' providing the name of the selected model.
- "event.window": a 'numeric' specifying the window width for event study output.
- "remap": a 'character' specifying the remapping technique used. Options are mentioned in "remap" argument description.

Function 'print.es' is provided to print the coefficients and exposures of the analysis. 'plot.es' is used to plot the model residuals and firm returns.

## Model arguments

Each model can take extra arguments (supplied as 'model.args') apart from mandatory ones for finer control over the analysis. Check the respective function documentation for definitions. The arguments from the relevant functions are listed here:

- "marketModel":
  - market.returns

- "excessReturn":
  - market.returns
- "lmAMM":
  - market.returns
  - others
  - switch.to.innov
  - market.returns.purge
  - nlag.makeX
  - nlag.lmAMM
  - dates
  - verbose

  Note: arguments (except nlag.lmAMM) are directly passed to 'makeX', see lmAMM for more details.

## Author(s)

Ajay Shah, Chirag Anand, Vikram Bahure, Vimal Balasubramaniam

## See Also

lmAMM, marketModel, excessReturn, phys2eventtime, inference.bootstrap, inference.wilcox, remap.cumsum, remap.cumprod, remap.event.reindex,

## Examples

```
data("StockPriceReturns")
data("SplitDates")
data("OtherReturns")

                # Event study without adjustment
es <- eventstudy(firm.returns = StockPriceReturns,
                 event.list = SplitDates,
                 event.window = 7,
                 type = "None",
                 to.remap = TRUE,
                 remap = "cumsum",
                 inference = TRUE,
                 inference.strategy = "bootstrap")
str(es)
plot(es)

                # Event study using Market Model
es <- eventstudy(firm.returns = StockPriceReturns,
                 event.list = SplitDates,
                 event.window = 7,
                 type = "marketModel",
                 to.remap = TRUE,
                 remap = "cumsum",
```

```
                  inference = TRUE,
                  inference.strategy = "bootstrap",
                  model.args = list(
                         market.returns = OtherReturns[, "NiftyIndex"]
                         )
                  )
str(es)
plot(es)

                 # Event study using Augmented Market Model
es <- eventstudy(firm.returns = StockPriceReturns,
                  event.list = SplitDates,
                  event.window = 7,
                  type = "lmAMM",
                  to.remap = TRUE,
                  remap = "cumsum",
                  inference = TRUE,
                  inference.strategy = "bootstrap",
                                              # model arguments
                  model.args = list(
                         market.returns = OtherReturns[, "NiftyIndex"],
                         others = OtherReturns[, "USDINR"],
                         market.returns.purge = TRUE,
                         nlag.makeX = 5,
                         nlag.lmAMM = 5
                         )
                  )
str(es)
plot(es)
```

---

`excessReturn` *Estimate excess returns over the market*

---

## Description

This function estimates excess returns over the market. If the firm return is "firm.returns" and market return is "market.returns", then output will be "firm.returns" less "market.returns".

## Usage

```
excessReturn(firm.returns, market.returns)
```

## Arguments

| | |
|---|---|
| `firm.returns` | a **zoo** timeseries with firm returns from which excess returns from market are to be calculated. |
| `market.returns` | a **zoo** object containing market index returns. |

## Value

Excess market return

## Author(s)

Vikram Bahure

## Examples

```
data(StockPriceReturns)
data(OtherReturns)

er.result <- excessReturn(firm.returns = StockPriceReturns,
  market.returns = OtherReturns$NiftyIndex)

tail(merge(excessReturn = er.result$Infosys,
           Infosys = StockPriceReturns$Infosys,
           NiftyIndex = OtherReturns$NiftyIndex,
           all=FALSE))
```

---

# get.clusters.formatted

*Get formatted clusters to perform extreme event study analysis (ees)*

---

## Description

The functions formats extreme event dates, dealing with clusters in the event frame.

## Usage

```
get.clusters.formatted(event.series,
                       response.series,
                       probvalue = 5,
                       event.value = "nonreturns",
                       response.value = "nonreturns")
```

## Arguments

| | |
|---|---|
| event.series | a **zoo** matrix of 'event' series. |
| response.series | a **zoo** matrix of 'response' series. |
| probvalue | The value (in percent) on the probability distribution to define a tail event. |
| event.value | a 'character' scalar specifying if the 'event.series' is to be converted to 'returns' or left as 'nonreturns' |
| response.value | a 'character' scalar specifying the if the 'response.series' is to be converted to 'returns' or left as 'nonreturns' |

## Details

Tail (Rare) events are often the object of interest in finance. These events are defined as those that have a low probability of occurrence. This function identifies such events based on 'probvalue' mentioned by the user and generates summary statistics about the events. If 'probvalue' is 2.5%, events below 2.5% (lower tail) and above 97.5% (upper tail) of the distribution are identified as extreme events.

Once the extreme events are defined, this function further formats the events. The extreme event functionality is muddled if we have another event occurring in the event time frame. Following the methodology of Patnaik. Shah and Singh (2013), we handle clustered events. Clustered events are handled in following ways:

- Clustered events which are defined as consecutive events, are fused into a single event and respective returns of response series are also fused.
- Mixed clusters are the left and right tail events occurring on consecutive days. These are identified and discarded from the analysis.

## Value

A **zoo** object is returned with formatted 'event.series' and 'response.series'. It also has separate columns to identify tail events, named 'left.tail' and 'right.tail', with binary outcome (1 equals tail event). Finally, the object has column named 'cluster.pattern' which identifies the length of the cluster in the event series.

## Author(s)

Vikram Bahure, Vimal Balasubramaniam

## References

*Ila Patnaik, Nirvikar Singh and Ajay Shah (2013). Foreign Investors under stress: Evidence from India. International Finance, 16(2), 213-244.* http://onlinelibrary.wiley.com/doi/10.1111/j.1468-2362.2013.12032.x/abstract http://macrofinance.nipfp.org.in/releases/PatnaikShahSingh2013_Foreign_Investors.html

## Examples

```
data(OtherReturns)

gcf <- get.clusters.formatted(event.series = OtherReturns$SP500,
              response.series = OtherReturns$NiftyIndex)

str(gcf, max.level = 2)
```

## IndexReturns *Market indice returns data*

### Description

This data set contains intra-day CNX NIFTY 50 index returns (in per cent) for days when RBI announced key interest rate cuts. These are used to estimate abnormal returns for stock price returns of 7 Indian banks with highest weight in BANK NIFTY Index.

### Usage

```
data(IndexReturns)
```

### Author(s)

Sargam Jain

## inference.bootstrap *Bootstrap inference for event study estimator*

### Description

This function obtains a boostrapped confidence interval for estimates of magnitude over the event horizon.

### Usage

```
inference.bootstrap(es.w,
                    to.plot = TRUE,
                    boot.run = 1000,
                    xlab = "Event time",
   ylab = "Cumulative returns of response series",
   main = "Event study plot")
```

### Arguments

| | |
|---|---|
| es.w | a **zoo** object indexed by event time: the "z.e" component of the list returned by the phys2eventtime function. The object should consist of more than one series. |
| boot.run | A 'numeric', controlling the number of simulations required for the bootstrap. |
| to.plot | a 'logical' indicating whether to generate an event study plot of the inference estimated. Defaults to 'TRUE'. |
| xlab | the x-axis label of the generated plot. Used if "to.plot" is 'TRUE'. |
| ylab | the y-axis label of the generated plot. Used if "to.plot" is 'TRUE'. |
| main | main title of the plot. Used if "to.plot" is 'TRUE'. |

## Value

A 'matrix' with 3 columns, the lower confidence interval (CI), the mean, and the upper CI which are the result of bootstrap inference.

## Author(s)

Vikram Bahure, Vimal Balasubramaniam

## See Also

boot phys2eventtime inference.wilcox inference.classic

## Examples

```
data(StockPriceReturns)
data(SplitDates)

es.results <- phys2eventtime(z = StockPriceReturns,
                             events = SplitDates,
                             width = 5)
es.w <- window(es.results$z.e,
               start = -5,
               end = +5)

eventtime <- remap.cumsum(es.w, is.pc = FALSE, base = 0)
inference.bootstrap(es.w = eventtime,
                    to.plot = FALSE)
```

---

# inference.classic *Classic T-inference for event study estimator*

---

## Description

This function uses standard student's t-statistic to obtain the estimate for response of the event and the corresponding confidence intervals.

## Usage

```
inference.classic(es.w,
                  to.plot = TRUE,
                  xlab = "Event time",
  ylab = "Cumulative returns of response series",
  main = "Event study plot")
```

### Arguments

| | |
|---|---|
| es.w | a **zoo** object indexed by event time: the "z.e" component of the list returned by the phys2eventtime function. The object should consist of more than one series. |
| to.plot | a 'logical' indicating whether to generate an event study plot of the inference estimated. Defaults to 'TRUE'. |
| xlab | the x-axis label of the generated plot. Used if "to.plot" is 'TRUE'. |
| ylab | the y-axis label of the generated plot. Used if "to.plot" is 'TRUE'. |
| main | main title of the plot. Used if "to.plot" is 'TRUE'. |

### Value

A 'matrix' with 3 columns, the lower confidence interval (CI), the mean, and the upper CI which are the result of inference drawn using student's inference.

### Author(s)

Sargam Jain

### See Also

boot phys2eventtime inference.wilcox inference.bootstrap

### Examples

```
data(StockPriceReturns)
data(SplitDates)

es.results <- phys2eventtime(z = StockPriceReturns,
                             events = SplitDates,
                             width = 5)
es.w <- window(es.results$z.e,
               start = -5,
               end = +5)

eventtime <- remap.cumsum(es.w, is.pc = FALSE, base = 0)
inference.classic(es.w = eventtime,
                  to.plot = FALSE)
```

---

## inference.wilcox *Wilcox inference for event study estimator*

---

### Description

This function does wilcox inference to generate distribution of average of all the cumulative returns time-series.

## Usage

```
inference.wilcox(es.w,
                 to.plot = TRUE,
                 xlab = "Event time",
                 ylab = "Cumulative returns of response series",
         main = "Event study plot")
```

## Arguments

| | |
|---|---|
| es.w | a **zoo** object indexed by event time: the "z.e" component of the list returned by the phys2eventtime function. The object should consist of more than one series. |
| to.plot | a 'logical' indicating whether to generate an event study plot of the inference estimated. Defaults to 'TRUE'. |
| xlab | the x-axis label of the generated plot. Used if "to.plot" is 'TRUE'. |
| ylab | the y-axis label of the generated plot. Used if "to.plot" is 'TRUE'. |
| main | main title of the plot. Used if "to.plot" is 'TRUE'. |

## Value

A 'matrix' with 3 columns: the lower confidence interval (CI), the mean, and the upper CI which are the result of wilcox inference.

## Author(s)

Vikram Bahure, Vimal Balasubramaniam

## See Also

phys2eventtime inference.bootstrap inference.classic

## Examples

```
data(StockPriceReturns)
data(SplitDates)

es.results <- phys2eventtime(z = StockPriceReturns,
                             events = SplitDates,
                             width = 5)
es.w <- window(es.results$z.e, start = -5, end = +5)
eventtime <- remap.cumsum(es.w, is.pc = FALSE, base = 0)

inference.wilcox(es.w = eventtime, to.plot = FALSE)
```

## KGEarningsDates *Earnings announcement data for replication of a standard event study*

### Description

This data set contains earnings announcement records of companies belonging to the services and technology sectors between January 9, 2007 and February 28, 2007.

### Usage

```
data(KGEarningsDates)
```

### Author(s)

Sargam Jain

### References

*Doron Kliger and Gregory Gurevich (2014). Event studies for financial research – A comprehensive guide.*

## KGMarketReturns *Market indice returns data*

### Description

This data set contains S&P 500 Index returns (in per cent) between January 9, 2007 and February 28, 2007. These are used to estimate abnormal returns for stock price returns of 670 individual firms in service and technology sector.

### Usage

```
data(KGMarketReturns)
```

### Author(s)

Sargam Jain

### References

*Doron Kliger and Gregory Gurevich (2014). Event studies for financial research – A comprehensive guide.*

## KGStockReturns *Stock price returns data*

### Description

This data set contains stock price returns (in per cent) of 670 individual firms between January 9, 2007 and February 28, 2007 in service and technology sector.

### Usage

```
data(KGStockReturns)
```

### Author(s)

Sargam Jain

### References

*Doron Kliger and Gregory Gurevich (2014). Event studies for financial research – A comprehensive guide.*

## KGSurpriseCategory *Classification of stocks*

### Description

This data set classifies the 670 firms analysed into 3 categories: good, bad, and medium – on the basis of unexpected deviation of realized earnings from the forecast following the earnings' announcements.

### Usage

```
data(KGSurpriseCategory)
```

### Author(s)

Sargam Jain

### References

*Doron Kliger and Gregory Gurevich (2014). Event studies for financial research – A comprehensive guide.*

`lmAMM` *Augmented market model (AMM) estimation*

## Description

'lmAMM' (linear augmented market models) estimates exposure and residuals.

## Usage

```
lmAMM(firm.returns, X, nlags = NULL, verbose = FALSE)
```

## Arguments

| | |
|---|---|
| `firm.returns` | a univariate 'zoo' object of data for one regressor (firm). |
| `X` | a matrix of regressors obtained by using 'makeX'. See 'Details' when this is specified as a market model. |
| `nlags` | specifies a lag length required from the specified set of regressors. When unspecified, the best lag using the AIC is used. |
| `verbose` | 'logical'. If 'TRUE', prints details of piece-wise analysis. |

## Details

This function estimates a linear regression model with multiple variables using 'lm', stores coefficients as 'exposures', and HAC adjusted standard errors as 's.exposures'.

This function is the core engine for other functions that estimate AMMs. Each regression is expected in this package to have a minimum of 30 observations, a condition that translates into a month of daily data. If the total number of observations is less than 30, the function returns 'NULL'.

Function 'makeX' is used to obtain a matrix of regressors used as input for 'X'.

If "nlags" is 'NULL', then the function finds the best lag structure using the AIC(n).

'lmAMM' calls `stats::lm` to estimate the linear model. 'print' function on an object of 'class' "amm" can be used to see the call (formula) to `lm`.

## Value

The function returns an object of 'class' "amm"; 'NULL' if `nrow(firm.returns) < 30`.

Function 'summary' is provided to print a summary of results. 'print' prints the coefficients and exposures of the analysis. 'plot' plots the model residuals and firm returns.

An object of class "amm" is a 'list' containing the output of `stats::lm` (which includes "residuals"), along with the following components:

| | |
|---|---|
| `exposures` | a 'numeric' containing exposure estimates for the firm. |
| `s.exposures` | a 'numeric' containing HAC adjusted standard error of the exposures estimated for the firm. |
| `nlags` | shows the lag length provided by user. |

### Author(s)

Ajay Shah, Chirag Anand, Vikram Bahure, Vimal Balasubramaniam

### See Also

makeX lm

### Examples

```
data("StockPriceReturns")
data("OtherReturns")

firm.returns <- StockPriceReturns[, "Infosys"]
market.returns <- OtherReturns[ ,"NiftyIndex"]
currency.returns <- OtherReturns[, "USDINR"]

X <- makeX(market.returns,
           others = currency.returns,
           switch.to.innov = FALSE,
           market.returns.purge = FALSE,
           nlags = 4,
           verbose = FALSE)

amm.result <- lmAMM(firm.returns, X, nlags = 3, verbose = FALSE)
plot(amm.result)

amm.residual <- residuals(amm.result)
amm.residual <- zoo(amm.residual,
                    order.by = as.Date(names(amm.residual)))

Comparison <- merge(AMMResidual = amm.residual,
                    Infosys = StockPriceReturns$Infosys,
                    NiftyIndex =  OtherReturns$NiftyIndex,
                    all = FALSE)
plot(Comparison, xlab="")
```

---

## makeX *Prepare regressors for computation of Augmented Market Models*

---

### Description

'makeX' is used to prepare regressors for computation of Augmented Market Models. It can be used to create a matrix of explanatory variables in the form specified by the user for estimation of AMM.

## Usage

```
makeX(market.returns,
      others,
      switch.to.innov = rep(TRUE, NCOL(others)),
      market.returns.purge = TRUE,
      nlags = 5,
      dates = NULL,
      verbose = FALSE)
```

## Arguments

| | |
|---|---|
| market.returns | a univariate timeseries object of 'class' **zoo**. In market models, this is normally the returns of a stock market index. |
| others | a **zoo** matrix of other regressors for the multiple regression model. |
| switch.to.innov | a 'logical' vector with an element for each column in "others" specifying whether to switch the column from raw values to auto-regressive residuals. Default is 'TRUE' for all the columns. See 'Details'. |
| market.returns.purge | a 'logical' indicating whether to purge the effects of "others" from "market.returns". See 'Details'. |
| nlags | a integer specifying the number of lags required when 'market.returns.purge' is 'TRUE'. |
| dates | a 'Date' vector, default set to 'NULL', specifying breaks. See 'Details'. |
| verbose | a 'logical' value, with the default being 'FALSE', specifying whether detailed output is required. |

## Details

This function prepares the regressors of interest for the purpose of running aumgmented market models.

The 'logical' vector 'switch.to.innov' is applicable only to the regressors in 'others' matrix and not to the time series vector in 'market.returns'.

When 'market.returns.purge' is 'TRUE', residuals from a regression of 'market.returns' on 'others' is obtained. If 'dates' are provided, this regression is done within sub-periods as identified by the 'dates' vector. When 'dates' is set to default value of 'NULL', the start and end points of 'market.returns' is taken as the period for estimation.

## Value

a time-series object of regressors is returned.

## Warning

The input data should not contain 'NA's, as is required by the "lm" function to estimate a linear regression. Please use 'na.omit' before feeding data into this function.

## Author(s)

Ajay Shah, Chirag Anand, Vikram Bahure, Vimal Balasubramaniam

## Examples

```
data("OtherReturns")
market.returns <- OtherReturns$NiftyIndex
currency.returns <- OtherReturns$USDINR

X <- makeX(market.returns,
           others = currency.returns,
           switch.to.innov = FALSE,
           market.returns.purge = FALSE,
           verbose = FALSE)
head(na.omit(X))
```

---

`manyfirmssubperiod.lmAMM` *Estimate exposure for many regressands over multiple periods*

---

## Description

`manyfirmssubperiod.lmAMM` estimates exposure for many regressands over a set of regressors obtained by using 'makeX' over multiple periods.

## Usage

```
manyfirmssubperiod.lmAMM(firm.returns,
                         X,
                         lags,
                         dates = NULL,
                         periodnames = NULL,
                         verbose = FALSE)
```

## Arguments

| | |
|---|---|
| `firm.returns` | a 'zoo' matrix of data for multiple regressands (firms). |
| `X` | a matrix of regressors obtained by using 'makeX'. |
| `lags` | an integer specifying the number of lags to be used in the market model. |
| `dates` | a 'Date' class vector, specifying break points in the time series to be used for sub-period identification. The default value is 'NULL' resulting in estimates identical to 'lmAMM' used over multiple regressands. |
| `periodnames` | a 'character' vector of names for each subperiod that has been marked by the "dates" argument. |
| `verbose` | 'logical', indicating whether the function should print detailed results. |

## Details

This function computes the exposure, and HAC adjusted standard errors to linear augmented market models estimated for several regressands across multiple periods.

## Warning

Do not have any space between names provided under "periodnames".

## Author(s)

Chirag Anand, Vikram Bahure, Vimal Balasubramaniam

## See Also

lmAMM

## Examples

```
data("StockPriceReturns", package = "eventstudies")
data("OtherReturns", package = "eventstudies")

firm.returns <- StockPriceReturns[, c("Infosys","TCS")]
market.returns <- OtherReturns$NiftyIndex
currency.returns <- OtherReturns$USDINR

X <- makeX(market.returns,
           others = currency.returns,
           nlags = 1,
           switch.to.innov = FALSE,
           market.returns.purge = FALSE,
           verbose = FALSE,
   dates = as.Date(c("2010-07-01", "2011-11-17", "2013-03-28")))

res <- manyfirmssubperiod.lmAMM(firm.returns = firm.returns,
                                X = X,
                                lags = 1,
                           dates = as.Date(c("2010-07-01", "2011-11-17", "2013-03-28")),
                                periodnames = c("P1", "P2"),
                                verbose = FALSE)
print(res)
```

---

`marketModel` *Extract residuals from a market model*

---

## Description

This function extracts residuals from a market model using function `stats:lm`. 'na.exclude' is passed as 'na.action' for 'lm'. For more than one firm, the function merges "market.returns" with each element of "firm.returns" before passing to 'lm' so that rows of firms without 'NA's are not removed from the zoo object.

## Usage

```
marketModel(firm.returns, market.returns, residuals = TRUE)
```

## Arguments

| | |
|---|---|
| firm.returns | a **zoo** time series object (univariate or otherwise) with firm returns. |
| market.returns | a **zoo** time series of market index returns. |
| residuals | a 'logical' indicating whether to return residuals or 'lm' object. When argument to the function includes the entire time series, returns are estimated using the entire data set and not just estimation period, value of residuals should be TRUE in such a case. |

## Value

Residual returns unexplained by market index returns.

## Author(s)

Chirag Anand, Vikram Bahure

## Examples

```
data("StockPriceReturns")
data("OtherReturns")

mm.result <- marketModel(firm.returns = StockPriceReturns,
                         market.returns = OtherReturns$NiftyIndex,
                         residuals = TRUE)

comparison <- merge(MarketModel = mm.result$Infosys,
                    Infosys = StockPriceReturns$Infosys,
                    NiftyIndex = OtherReturns$NiftyIndex,
                    all = FALSE)
plot(comparison)
```

---

# OtherReturns *Data set containing daily returns of Nifty index, USD INR, call momey rate, and S&P 500 index*

---

## Description

This data set consists of daily time series of market returns (Nifty index and S&P 500 index), currency returns (USD/INR), and call money rate.

The data series is a daily time-series zoo object. All series are in per cent.

### Usage

```
data(OtherReturns)
```

### Author(s)

Chirag Anand

---

`phys2eventtime` *Convert data from physical to event time*

---

### Description

'phys2eventtime' is used to convert data from physical to event time using information on events identified by the user.

### Usage

```
phys2eventtime(z, events, width = 10)
```

### Arguments

| | |
|---|---|
| `z` | an object of class **zoo** or **xts** containing data to be converted into event time. |
| `events` | 'data.frame' containing event identifiers. See 'Details'. |
| `width` | an 'integer' specifying the event window within which data should be available to consider the outcome a 'success'. See 'Details'. |

### Details

"events" object contains two columns: "name" consists of names of the event, and "when" is the respective event time. 'class' of 'index' of "z" and "when" should be same and one of the 'date-time' or 'Date' classes.

If an event date does not lie within the index of "z", the function approximates to the nearest previous time using `findInterval`. Note that 'findInterval' assumes the index of 'z' is non-decreasing.

The argument "width" provides the user with an option to define successful events as those that have data within a window around the event. Window is defined as (-width, +width]. If "width" is 10 periods, those events with 'NA' within 10 periods before and after the event will be classified as "wdatamissing", otherwise, the event "outcome" will be classified as "success".

Currently this function requires "z" to have at least one *column*. It relies on 'colnames' of "z" for the series names, and matches it with the "events" object. One can use `drop = FALSE` with '[' to achieve a single-column object.

## Value

Returns a 'list' of two elements:

| | |
|---|---|
| z.e | a **zoo** object containing data of successful events indexed with event time; "NULL" if there are no "success" in "outcomes". 'colnames' of z.e are event numbers: row numbers of "events". |
| outcomes | a character vector with outcome definition for each event. |

- success: the successful use of an event.
- wdatamissing: when there are NAs within the event window.
- wrongspan: when event time cannot be mapped to physical time in "z" because event window is outside 'index(z)'.
- unitmissing: when the unit (a column) is missing in "z".

## Author(s)

Ajay Shah, Chirag Anand, Vikram Bahure, Vimal Balasubramaniam

## See Also

findInterval

## Examples

```
data(StockPriceReturns)
data(SplitDates)

result <- phys2eventtime(z = StockPriceReturns,
 events = SplitDates,
 width = 5)
print(result$z.e[as.character(-4:5)])
print(SplitDates[result$outcomes == "success", ])
```

---

# RateCuts *Dates of RBI key interest rate cuts*

---

## Description

This data set contains dates of announcements of key interest rate cuts by Reserve Bank of India (RBI). For each announcement date, each company is considered a different entity. For example, for two different rate cuts say, on April 17, 2012 and January 29, 2013, ICICI Bank is considered a separate entity.

## Usage

```
data(RateCuts)
```

### Author(s)

Sargam Jain

---

`remap.cumprod` *Cumulative geometric values*

---

### Description

This function computes the cumulative geometric values for a given **zoo** object.

### Usage

```
remap.cumprod(z, is.pc = FALSE, is.returns = TRUE, base = 100)
```

### Arguments

| | |
|---|---|
| `z` | a **zoo** object indexed by event time, typically by the "z.e" component obtained from "phys2eventtime" function. |
| `is.pc` | 'logical', whether input is a percentage. Default value set to 'FALSE'. |
| `is.returns` | 'logical', whether input is a returns series. |
| `base` | an integer specifying the base for cumulative product. |

### Value

A **zoo** object with the cumulative product for each series, representing a buy-hold return estimate.

### Author(s)

Chirag Anand, Vikram Bahure, Vimal Balasubramaniam

### See Also

`phys2eventtime`

### Examples

```
data(StockPriceReturns)
data(SplitDates)

es.results <- phys2eventtime(z = StockPriceReturns,
                             events = SplitDates,
                             width = 5)
es.w <- window(es.results$z.e, start = -5, end = +5)

eventtime <- remap.cumprod(es.w,
                           is.pc = TRUE,
                           is.returns = TRUE,
                           base = 100)
```

```
print(eventtime[as.character(-3:3), ])
```

---

`remap.cumsum` *Cumulative values*

---

## Description

This function remaps a time series into its cumulative summation.

## Usage

```
remap.cumsum(z, is.pc = FALSE, base = 0)
```

## Arguments

| | |
|---|---|
| `z` | a **zoo** object indexed by event time, typically by the "z.e" component obtained from "phys2eventtime" function. |
| `is.pc` | 'logical', whether input is a percentage. Default value set to 'FALSE'. |
| `base` | an integer specifying the base for cumulative sum. |

## Details

This function remaps a time series into ints cumulative summation. Function assigns first value as zero in the event window (-width) before cumulating the values.

## Value

A **zoo** object with the cumulative summation for each series.

## Author(s)

Chirag Anand, Vikram Bahure, Vimal Balasubramaniam

## See Also

`phys2eventtime`

## Examples

```
data(StockPriceReturns)
data(SplitDates)

es.results <- phys2eventtime(z = StockPriceReturns,
                             events = SplitDates,
                             width = 5)
es.w <- window(es.results$z.e, start = -5, end = +5)
eventtime <- remap.cumsum(es.w, is.pc = FALSE, base = 0)

print(eventtime[as.character(-3:3), ])
```

`remap.event.reindex` *Re-index value within event window*

## Description

Reset value at the beginning of the event window to a 100 and reindex thereon.

## Usage

```
remap.event.reindex(z)
```

## Arguments

| | |
|---|---|
| z | z is a zoo object obtained from phys2eventtime. |

## Value

Rescaled returns value

## Author(s)

Chirag Anand, Vikram Bahure, Vimal Balasubramaniam

## See Also

phys2eventtime

## Examples

```
data(StockPriceReturns)
data(SplitDates)

es.results <- phys2eventtime(z = StockPriceReturns,
                             events = SplitDates,
                             width = 5)
es.w <- window(es.results$z.e, start = -5, end = +5)

eventtime <- remap.event.reindex(es.w)

eventtime[as.character(-3:3), ]
```

## SplitDates *Data set of events used to perform event study analysis*

### Description

This data set contains stock split event dates for the index constituents of the Bombay Stock Exchange index (SENSEX). The data format follows the required format in the function `phys2eventtime`, with two columns 'outcome.unit' (firm name) and 'event.when' (stock split date).

### Usage

```
data(SplitDates)
```

### Author(s)

Vikram Bahure

## StockPriceReturns *Stock price returns data*

### Description

This data set contains stock price returns (in per cent) of 30 major stocks on the National Stock Exchange (NSE) of India.

### Usage

```
data(StockPriceReturns)
```

### Author(s)

Vikram Bahure

`subperiod.lmAMM` *Estimate exposure for a single regressor over multiple periods*

## Description

This function estimates exposure for a single regressand over a set of regressors obtained by using 'makeX' over multiple periods.

## Usage

```
subperiod.lmAMM(firm.returns,
                X,
                nlags = 1,
                verbose = FALSE,
                dates = NULL,
                residual = TRUE)
```

## Arguments

| | |
|---|---|
| `firm.returns` | a 'zoo' vector of data for one regressand (firm). |
| `X` | a matrix of regressors obtained by using 'makeX'. See 'Details' when this is specified as a market model. |
| `nlags` | specifies a lag length required from the specified set of regressors. When unspecified, the best lag using the AIC is computed for the market model. |
| `verbose` | 'logical', indicating whether the function should print detailed results. |
| `dates` | a 'Date' class vector, specifying break points in the time series to be used for sub-period identification. The default value is 'NULL' resulting in estimates identical to 'lmAMM'. |
| `residual` | 'logical', returns AMM residuals if TRUE, AMM exposure otherwise. Defaults to 'TRUE'. |

## Details

When 'dates' is 'NULL', resulting estimates from this function is identical to 'lmAMM'.

## Value

A 'list' object of length 3 is returned with:

- "exposures": A matrix of exposures by sub-period and regressands.
- "sds": HAC adjusted standard errors for all exposures.
- "residuals": Contain residuals of class **xts** from the fitted model for all sub-periods.

## Author(s)

Chirag Anand, Vikram Bahure, Vimal Balasubramaniam

## See Also

lmAMM

## Examples

```
data("StockPriceReturns")
data("OtherReturns")

firm.returns <- StockPriceReturns$Infosys
market.returns <- OtherReturns$NiftyIndex
currency.returns <- OtherReturns$USDINR

regressors <- makeX(market.returns,
                    others = currency.returns,
                    switch.to.innov = TRUE,
                    market.returns.purge = TRUE,
                    nlags = 1,
                    dates = as.Date(c("2010-07-01", "2011-11-17", "2013-03-28")),
                    verbose = FALSE)

res <- subperiod.lmAMM(firm.returns,
                       X = regressors,
                       nlags = 1,
                       verbose = FALSE,
                       dates = as.Date(c("2010-07-01", "2011-11-17", "2013-03-28")))
str(res)
```

---

# TerrorAttack *Dates of terrorist attack*

---

## Description

Accounting for different time zones across 33 global capital markets, this data set provides different dates for each capital market on the day of the terrorist attack of September 11, 2001 EST.

## Usage

```
data(TerrorAttack)
```

## Author(s)

Sargam Jain

## References

*Andrew H Chen and Thomas F Siems (2004). The effects of terrorism on global capital markets. European Journal of Political Economy, 20(1), 349-366. doi: 10.1016/j.ejpoleco.2003.12.005*

`TerrorIndiceReturns` *Stock indice returns data*

## Description

This data set contains daily stock indice returns (in per cent) of 33 global capital markets around September 11, 2009. The daily stock indice returns for 30 days pre and post the terrorist attack are used in the event studies analysis.

## Usage

```
data(TerrorIndiceReturns)
```

## Author(s)

Sargam Jain

## References

*Andrew H Chen and Thomas F Siems (2004). The effects of terrorism on global capital markets. European Journal of Political Economy, 20(1), 349-366. doi: 10.1016/j.ejpoleco.2003.12.005*

# Index


∗Topic **OtherReturns**
OtherReturns, 27
∗Topic **constantMeanReturn**
constantMeanReturn, 3
∗Topic **datasets**
AggregateReturns, 3
IndexReturns, 16
KGEarningsDates, 20
KGMarketReturns, 20
KGStockReturns, 21
KGSurpriseCategory, 21
RateCuts, 29
SplitDates, 33
StockPriceReturns, 33
TerrorAttack, 35
TerrorIndiceReturns, 36
∗Topic **eventstudies**
eventstudies-package, 2
∗Topic **eventstudy**
eventstudy, 9
∗Topic **excessReturn**
excessReturn, 13
∗Topic **lmAMM**
lmAMM, 22
∗Topic **makeX**
makeX, 23
∗Topic **manyfirmssubperiod.lmAMM**
manyfirmssubperiod.lmAMM, 25
∗Topic **marketModel**
marketModel, 26
∗Topic **phys2eventime**
phys2eventtime, 28
∗Topic **subperiod.lmAMM**
subperiod.lmAMM, 34

AggregateReturns, 3

boot, *17, 18*

constantMeanReturn, 3

eesDates, 4
eesInference, 6
eesSummary, 7
eventstudies-package, 2
eventstudy, 9
excessReturn, *10*, *12*, 13

findInterval, *28, 29*

get.clusters.formatted, 14

IndexReturns, 16
inference.bootstrap, *11, 12*, 16, *18, 19*
inference.classic, *17*, 17, *19*
inference.wilcox, *11, 12*, *17, 18*, 18

KGEarningsDates, 20
KGMarketReturns, 20
KGStockReturns, 21
KGSurpriseCategory, 21

lm, *23*
lmAMM, *10*, *12*, 22, *26*, *35*

makeX, *23*, 23
manyfirmssubperiod.lmAMM, *25*, 25
marketModel, *10*, *12*, 26

OtherReturns, 27

phys2eventtime, *6*, *10–12*, *16–19*, 28, *30–32*

RateCuts, 29
remap.cumprod, *10*, *12*, 30
remap.cumsum, *10*, *12*, 31
remap.event.reindex, *10*, *12*, 32

SplitDates, 33
StockPriceReturns, 33
subperiod.lmAMM, 34

TerrorAttack, 35
TerrorIndiceReturns, 36