Package 'estudy2'

November 15, 2021

Type Package

Title An Implementation of Parametric and Nonparametric Event Study

Version 0.10.0

Date 2021-11-15

Description An implementation of a most commonly used event study methodology, including both parametric and nonparametric tests. It contains variety aspects of the rate of return estimation (the core calculation is done in C++), as well as three classical for event study market models: mean adjusted returns, market adjusted returns and single-index market models. There are 6 parametric and 6 nonparametric tests provided, which examine cross-sectional daily abnormal return (see the documentation of the functions for more information). Parametric tests include tests proposed by Brown and Warner (1980) <DOI:10.1016/0304-405X(80)90002-1>, Brown and Warner (1985) <DOI:10.1016/0304-405X(85)90042-X>, Boehmer et al. (1991) <DOI:10.1016/0304-405X(91)90032-F>, Patell (1976) <DOI:10.2307/2490543>, and Lamb (1995) <DOI:10.2307/253695>. Nonparametric tests covered in estudy2 are tests described in Corrado and Zivney (1992) <DOI:10.2307/2331331>, McConnell and Muscarella (1985) <DOI:10.1016/0304-405X(85)90006-6>, Boehmer et al. (1991) <DOI:10.1016/0304-405X(91)90032-F>, Cowan (1992) <DOI:10.1007/BF00939016>, Corrado (1989) <DOI:10.1016/0304-405X(89)90064-0>, Campbell and Wasley (1993) <DOI:10.1016/0304-405X(93)90025-7>, Savickas (2003) <DOI:10.1111/1475-6803.00052>, Kolari and Pynnonen (2010) <DOI:10.1093/rfs/hhq072>. Furthermore, tests for the cumulative abnormal returns proposed by Brown and Warner (1985) <DOI:10.1016/0304-405X(85)90042-X> and Lamb (1995) <DOI:10.2307/253695> are included.

License GPL-3

LazyData TRUE

URL `https://github.com/irudnyts/estudy2`,
`https://irudnyts.github.io/estudy2/`

BugReports `https://github.com/irudnyts/estudy2/issues`

Depends R (>= 4.1)

Imports quantmod (>= 0.4.18), zoo (>= 1.8.9), matrixStats (>= 0.60.0), Rcpp (>= 1.0.7), curl (>= 4.3.2)

LinkingTo Rcpp

RoxygenNote 7.1.1

Suggests knitr (>= 1.33), rmarkdown (>= 2.10), purrr (>= 0.3.4), shiny (>= 1.6.0), shinyFeedback (>= 0.4.0), shinyWidgets (>= 0.6.0), DT (>= 0.19), bslib (>= 0.2.5.1), stringr (>= 1.4.0), magrittr (>= 2.0.1), formattable (>= 0.2.1), dplyr (>= 1.0.7)

VignetteBuilder knitr

Encoding UTF-8

NeedsCompilation yes

Author Iegor Rudnytskyi [aut, cre]

Maintainer Iegor Rudnytskyi <iegor.rudnytskyi@gmail.com>

Repository CRAN

Date/Publication 2021-11-15 12:50:07 UTC

R topics documented:

apply_market_model . . . 3
boehmer . . . 6
brown_warner_1980 . . . 8
brown_warner_1985 . . . 10
car_brown_warner_1985 . . . 11
car_lamb . . . 13
car_nonparametric_tests . . . 15
car_parametric_tests . . . 17
car_rank_test . . . 20
corrado_sign_test . . . 22
generalized_sign_test . . . 23
get_prices_from_tickers . . . 25
get_rates_from_prices . . . 27
lamb . . . 28
modified_rank_test . . . 30
nonparametric_tests . . . 32
parametric_tests . . . 34
patell . . . 36
prices . . . 38
prices_indx . . . 39
rank_test . . . 39
rates . . . 41
rates_indx . . . 42
returns . . . 42
run_app . . . 46
securities_returns . . . 46
sign_test . . . 47

t_test 48
wilcoxon_test 50

Index **53**

---

apply_market_model *Apply a market model and return a list of `returns` objects.*

Description

The function applies a given market model to securities' rates of returns and returns a list of `returns` objects for each security, which can be passed directly to a whole battery of tests.

Usage

```
apply_market_model(
  rates,
  regressors,
  same_regressor_for_all = TRUE,
  market_model = c("mean_adj", "mrkt_adj", "sim"),
  estimation_method = c("ols"),
  estimation_start,
  estimation_end
)
```

Arguments

`rates`	an object of `list`, `data.frame`, `zoo` containing rates of returns of securities.
`regressors`	an object of the same class as `rates` containing regressors. The argument can be omitted, if market model is `mean_adj`. `regressors` must have the same number of components as `rates` except cases when the same regressor is used for all securities.
`same_regressor_for_all`	logical. Should the same regressor be used for each security? The default value is TRUE.
`market_model`	a character indicating the market model among `mean_adj`, `mrkt_adj`, and `sim`.
`estimation_method`	a character specifying an estimation method for `sim` model.
`estimation_start`	an object of `Date` class giving the first date of the estimation period.
`estimation_end`	an object of `Date` class giving the last date of the estimation period.

Details

The generic function is dispatched for such classes as `list`, `data.frame`, and `zoo`. If `same_regressor_for_all` is TRUE, and `regressors` has the length greater than one, the first element of `regressors` will be applied for each security in `rates`.

Value

A list of `returns` objects.

References

Brown S.J., Warner J.B. *Using Daily Stock Returns, The Case of Event Studies*. Journal of Financial Economics, 14:3-31, 1985.

See Also

returns

Examples

```
## 1. Mean-adjusted-returns model
## Not run:
library("magrittr")
tickers <- c("AMZN", "ZM", "UBER", "NFLX", "SHOP", "FB", "UPWK")
securities_returns <- get_prices_from_tickers(tickers,
                                              start = as.Date("2019-04-01"),
                                              end = as.Date("2020-04-01"),
                                              quote = "Close",
                                              retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous") %>%
    apply_market_model(market_model = "mean_adj",
                       estimation_start = as.Date("2019-04-01"),
                       estimation_end = as.Date("2020-03-13"))

## End(Not run)
## The result of the code above is equivalent to:
data(rates)
securities_returns <- apply_market_model(
    rates,
    market_model = "mean_adj",
    estimation_start = as.Date("2019-04-01"),
    estimation_end = as.Date("2020-03-13")
)

## 2. Market-adjusted-returns model
## Not run:
library("magrittr")
rates_indx <- get_prices_from_tickers("^GSPC",
                                      start = as.Date("2019-04-01"),
                                      end = as.Date("2020-04-01"),
                                      quote = "Close",
                                      retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous")
tickers <- c("AMZN", "ZM", "UBER", "NFLX", "SHOP", "FB", "UPWK")
```

```
securities_returns <- get_prices_from_tickers(tickers,
                                              start = as.Date("2019-04-01"),
                                              end = as.Date("2020-04-01"),
                                              quote = "Close",
                                              retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous") %>%
    apply_market_model(regressor = rates_indx,
                       same_regressor_for_all = TRUE,
                       market_model = "mrkt_adj",
                       estimation_start = as.Date("2019-04-01"),
                       estimation_end = as.Date("2020-03-13"))

## End(Not run)
## The result of the code above is equivalent to:
data(rates, rates_indx)
securities_returns <- apply_market_model(
    rates = rates,
    regressor = rates_indx,
    same_regressor_for_all = TRUE,
    market_model = "mrkt_adj",
    estimation_start = as.Date("2019-04-01"),
    estimation_end = as.Date("2020-03-13")
)

## 3. Single-index market model
## Not run:
library("magrittr")
rates_indx <- get_prices_from_tickers("^GSPC",
                                      start = as.Date("2019-04-01"),
                                      end = as.Date("2020-04-01"),
                                      quote = "Close",
                                      retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous")
tickers <- c("AMZN", "ZM", "UBER", "NFLX", "SHOP", "FB", "UPWK")
securities_returns <- get_prices_from_tickers(tickers,
                                              start = as.Date("2019-04-01"),
                                              end = as.Date("2020-04-01"),
                                              quote = "Close",
                                              retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous") %>%
    apply_market_model(regressor = rates_indx,
                       same_regressor_for_all = TRUE,
                       market_model = "sim",
                       estimation_method = "ols",
                       estimation_start = as.Date("2019-04-01"),
                       estimation_end = as.Date("2020-03-13"))
```

```
## End(Not run)
## The result of the code above is equivalent to:
data(rates, rates_indx)
securities_returns <- apply_market_model(
    rates = rates,
    regressor = rates_indx,
    same_regressor_for_all = TRUE,
    market_model = "sim",
    estimation_method = "ols",
    estimation_start = as.Date("2019-04-01"),
    estimation_end = as.Date("2020-03-13")
)
```

boehmer *Boehmer's parametric test (1991).*

Description

An event study parametric test described in Boehmer 1991.

Usage

```
boehmer(list_of_returns, event_start, event_end)
```

Arguments

`list_of_returns`	a list of objects of S3 class `returns`, each element of which is treated as a security.
`event_start`	an object of `Date` class giving the first date of the event period.
`event_end`	an object of `Date` class giving the last date of the event period.

Details

Performs a parametric test for event study, which is described in Boehmer 1991. Also called hybrid test or standardized cross-sectional test. This test performs t-test based on Patell's standardized residuals. By combining Patell's and t-tests, this test allows for event-induced variance changes, but still assumes cross-sectional independence. The test examines the hypothesis whether the theoretical cross-sectional expected value for a given day is equal to zero. It calculates statistics even if event window and estimation period are overlapped (intersect). The critical values has Student's t-distribution. The significance levels of α are 0.1, 0.05, and 0.01 (marked respectively by *, **, and ***).

Value

A data frame of the following columns:

- `date`: a calendar date
- `weekday`: a day of the week
- `percentage`: a share of non-missing observations for a given day
- `mean`: an average abnormal return
- `bh_stat`: a Boehmer's test statistic
- `bh_signif`: a significance of the statistic

References

- Patell J.M. *Corporate forecasts of earnings per share and stock price behavior: empirical tests*. Journal of Accounting Research, 14(2):246- 276, 1976.
- Boehmer E., Musumeci J., Poulsen A.B. *Event-study methodology under conditions of event-induced variance*. Journal of Financial Economics, 30(2):253-272, 1991.

See Also

`parametric_tests`, `brown_warner_1980`, `brown_warner_1985`, `t_test`, `patell`, and `lamb`.

Examples

```
## Not run:
library("magrittr")
rates_indx <- get_prices_from_tickers("^GSPC",
                                      start = as.Date("2019-04-01"),
                                      end = as.Date("2020-04-01"),
                                      quote = "Close",
                                      retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous")
tickers <- c("AMZN", "ZM", "UBER", "NFLX", "SHOP", "FB", "UPWK")
get_prices_from_tickers(tickers,
                        start = as.Date("2019-04-01"),
                        end = as.Date("2020-04-01"),
                        quote = "Close",
                        retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous") %>%
    apply_market_model(regressor = rates_indx,
                       same_regressor_for_all = TRUE,
                       market_model = "sim",
                       estimation_method = "ols",
                       estimation_start = as.Date("2019-04-01"),
                       estimation_end = as.Date("2020-03-13")) %>%
    boehmer(event_start = as.Date("2020-03-16"),
            event_end = as.Date("2020-03-20"))

## End(Not run)
## The result of the code above is equivalent to:
data(securities_returns)
boehmer(list_of_returns = securities_returns,
        event_start =  as.Date("2020-03-16"),
        event_end = as.Date("2020-03-20"))
```

`brown_warner_1980` *Brown and Warner parametric test (1980).*

Description

An event study parametric test described in Brown and Warner 1980.

Usage

```
brown_warner_1980(list_of_returns, event_start, event_end)
```

Arguments

`list_of_returns`	a list of objects of S3 class `returns`, each element of which is treated as a security.
`event_start`	an object of `Date` class giving the first date of the event period.
`event_end`	an object of `Date` class giving the last date of the event period.

Details

Performs a parametric test for the event study, which is described in Brown and Warner 1980. The test assumes a cross-sectional independence and an insignificance of event-induced variance. The test examines the hypothesis whether the theoretical cross-sectional expected value for a given day is equal to zero. The standard deviation in statistics is calculated as the cross-sectional mean of companies' variances, estimated on the estimation period. It calculates statistics even if the event window and the estimation period are overlapped (intersect). The critical values are Student's t-distributed (no approximation in limit). The significance levels of α are 0.1, 0.05, and 0.01 (marked respectively by *, **, and ***). It was designed to measure monthly data: for daily data look at Brown and Warner 1985 and `brown_warner_1985`.

Value

A data frame of the following columns:

- `date`: a calendar date
- `weekday`: a day of the week

- `percentage`: a share of non-missing observations for a given day
- `mean`: an average abnormal return
- `bw_1980_stat`: a Brown and Warner (1980) test statistic
- `bw_1980_signif`: a significance of the statistic

References

Brown S.J., Warner J.B. *Measuring security price performance*. Journal of Financial Economics, 8:205-258, 1980.

See Also

`parametric_tests`, `brown_warner_1985`, `t_test`, `patell`, `boehmer`, and `lamb`.

Examples

```
## Not run:
library("magrittr")
rates_indx <- get_prices_from_tickers("^GSPC",
                                      start = as.Date("2019-04-01"),
                                      end = as.Date("2020-04-01"),
                                      quote = "Close",
                                      retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous")
tickers <- c("AMZN", "ZM", "UBER", "NFLX", "SHOP", "FB", "UPWK")
get_prices_from_tickers(tickers,
                        start = as.Date("2019-04-01"),
                        end = as.Date("2020-04-01"),
                        quote = "Close",
                        retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous") %>%
    apply_market_model(regressor = rates_indx,
                       same_regressor_for_all = TRUE,
                       market_model = "sim",
                       estimation_method = "ols",
                       estimation_start = as.Date("2019-04-01"),
                       estimation_end = as.Date("2020-03-13")) %>%
    brown_warner_1980(event_start = as.Date("2020-03-16"),
                      event_end = as.Date("2020-03-20"))

## End(Not run)
## The result of the code above is equivalent to:
data(securities_returns)
brown_warner_1980(list_of_returns = securities_returns,
                  event_start = as.Date("2020-03-16"),
                  event_end = as.Date("2020-03-20"))
```

`brown_warner_1985` *Brown and Warner parametric test (1985).*

Description

An event study parametric test described in Brown and Warner 1985.

Usage

```
brown_warner_1985(list_of_returns, event_start, event_end)
```

Arguments

`list_of_returns`	a list of objects of S3 class `returns`, each element of which is treated as a security.
`event_start`	an object of `Date` class giving the first date of the event period.
`event_end`	an object of `Date` class giving the last date of the event period.

Details

Performs a parametric test for event study, which is described in Brown and Warner 1985, which is a traditional event study approach. This test does not require cross-sectional independence but is non-robust to an event-induced variance. The test examines the hypothesis whether the theoretical cross-sectional expected value for a given day is equal to zero. The standard deviation in statistics is estimated as the cross-sectional standard deviation of companies' means, estimated on the estimation period. It calculates statistics even if event window and estimation period are overlapped (intersect). The critical values are Student's t-distributed (no approximation in limit). The significance levels of α are 0.1, 0.05, and 0.01 (marked respectively by *, **, and ***).

Value

A data frame of the following columns:

- `date`: a calendar date
- `weekday`: a day of the week
- `percentage`: a share of non-missing observations for a given day
- `mean`: an average abnormal return
- `bw_1985_stat`: a Brown and Warner (1985) test statistic
- `bw_1985_signif`: a significance of the statistic

References

Brown S.J., Warner J.B. *Using Daily Stock Returns, The Case of Event Studies*. Journal of Financial Economics, 14:3-31, 1985.

See Also

parametric_tests, brown_warner_1980, t_test, patell, boehmer, and lamb.

Examples

```
## Not run:
library("magrittr")
rates_indx <- get_prices_from_tickers("^GSPC",
                                      start = as.Date("2019-04-01"),
                                      end = as.Date("2020-04-01"),
                                      quote = "Close",
                                      retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous")
tickers <- c("AMZN", "ZM", "UBER", "NFLX", "SHOP", "FB", "UPWK")
get_prices_from_tickers(tickers,
                        start = as.Date("2019-04-01"),
                        end = as.Date("2020-04-01"),
                        quote = "Close",
                        retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous") %>%
    apply_market_model(regressor = rates_indx,
                       same_regressor_for_all = TRUE,
                       market_model = "sim",
                       estimation_method = "ols",
                       estimation_start = as.Date("2019-04-01"),
                       estimation_end = as.Date("2020-03-13")) %>%
    brown_warner_1985(event_start = as.Date("2020-03-16"),
                      event_end = as.Date("2020-03-20"))

## End(Not run)
## The result of the code above is equivalent to:
data(securities_returns)
brown_warner_1985(list_of_returns = securities_returns,
                  event_start = as.Date("2020-03-16"),
                  event_end = as.Date("2020-03-20"))
```

---

car_brown_warner_1985 *Brown and Warner (1985) CAR test.*

Description

A parametric test proposed by Brown and Warner 1995 that examines whether or not cumulative abnormal return (CAR) significantly differs from zero.

Usage

```
car_brown_warner_1985(list_of_returns, car_start, car_end, percentage = 90)
```

Arguments

`list_of_returns`	a list of objects of S3 class `returns`, each element of which is treated as a security.
`car_start`	an object of `Date` class giving the first date of the CAR period.
`car_end`	an object of `Date` class giving the last date of the CAR period.
`percentage`	a lowest allowed percentage of non-missing observation for each day to be incorporated into CAR. The default value is 90 percent.

- `name`: a name of the test, i.e. `"car_brown_warner_1985"`
- `car_start`: the first date of the CAR period
- `car_end`: the last date of the CAR period
- `average_percentage`: an average share of non-missing observations over the CAR period
- `car_mean`: an average abnormal return over the CAR period
- `statistic`: a test's statistic
- `number_of_days`: the number of days in the CAR period
- `significance`: a significance of the statistic

Details

This function performs a test proposed by Brown and Warner 1985 to investigate whether CAR significantly differs from zero. This tests uses the variance, specified by Brown and Warner 1985. The advantage of this test is allowance for correlated cross-sectional returns. However, the test does not use autocorrelation adjustment. The test statistic is close enough to statistic, produced by `car_lamb`. The critical values are standard normal. The significance levels of α are 0.1, 0.05, and 0.01 (marked respectively by *, **, and ***).

References

Brown S.J., Warner J.B. *Using Daily Stock Returns, The Case of Event Studies*. Journal of Financial Economics, 14:3-31, 1985.

See Also

`car_lamb` and `car_parametric_tests`.

Examples

```
## Not run:
library("magrittr")
rates_indx <- get_prices_from_tickers("^GSPC",
                                      start = as.Date("2019-04-01"),
                                      end = as.Date("2020-04-01"),
                                    quote = "Close",
                                    retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous")
tickers <- c("AMZN", "ZM", "UBER", "NFLX", "SHOP", "FB", "UPWK")
get_prices_from_tickers(tickers,
                        start = as.Date("2019-04-01"),
                        end = as.Date("2020-04-01"),
                        quote = "Close",
                        retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous") %>%
    apply_market_model(regressor = rates_indx,
                       same_regressor_for_all = TRUE,
                       market_model = "sim",
                       estimation_method = "ols",
                       estimation_start = as.Date("2019-04-01"),
                       estimation_end = as.Date("2020-03-13")) %>%
    car_brown_warner_1985(car_start = as.Date("2020-03-16"),
                          car_end = as.Date("2020-03-20"))

## End(Not run)
## The result of the code above is equivalent to:
data(securities_returns)
car_brown_warner_1985(
    list_of_returns = securities_returns,
    car_start = as.Date("2020-03-16"),
    car_end = as.Date("2020-03-20")
)
```

car_lamb   *Lamb's CAR test (1995).*

Description

A parametric test proposed by Lamb 1995 that examines whether or not the cumulative abnormal return (CAR) significantly differs from zero.

Usage

```
car_lamb(list_of_returns, car_start, car_end, percentage = 90)
```

Arguments

`list_of_returns`	a list of objects of S3 class `returns`, each element of which is treated as a security.

`car_start`	an object of `Date` class giving the first date of the CAR period.
`car_end`	an object of `Date` class giving the last date of the CAR period.
`percentage`	a lowest allowed percentage of non-missing observation for each day to be incorporated into CAR. The default value is 90 percent.

Details

This function performs a test proposed by Lamb 1995 to investigate whether CAR significantly differs from zero. This tests uses the variance, specified by Lamb 1995. The advantage of this test is allowance for correlated cross-sectional returns. The test statistic is close enough to statistic, produced by `car_brown_warner_1985`. The critical values are standard normal. The significance levels of α are 0.1, 0.05, and 0.01 (marked respectively by *, **, and ***).

Value

A data frame of the following columns:

- `name`: a name of the test, i.e. `"car_lamb"`
- `car_start`: the first date of the CAR period
- `car_end`: the last date of the CAR period
- `average_percentage`: an average share of non-missing observations over the CAR period
- `car_mean`: an average abnormal return over the CAR period
- `statistic`: a test's statistic
- `number_of_days`: the number of days in the CAR period
- `significance`: a significance of the statistic

References

Lamb R.P. *An Exposure-Based Analysis of Property-Liability Insurer Stock Values around Hurricane Andrew*. Journal of Risk and Insurance, 62(1):111-123, 1995.

See Also

`car_brown_warner_1985` and `car_parametric_tests`.

Examples

```
## Not run:
library("magrittr")
rates_indx <- get_prices_from_tickers("^GSPC",
                                      start = as.Date("2019-04-01"),
                                      end = as.Date("2020-04-01"),
                                      quote = "Close",
                                      retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous")
tickers <- c("AMZN", "ZM", "UBER", "NFLX", "SHOP", "FB", "UPWK")
```

```
get_prices_from_tickers(tickers,
                        start = as.Date("2019-04-01"),
                        end = as.Date("2020-04-01"),
                        quote = "Close",
                        retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous") %>%
    apply_market_model(regressor = rates_indx,
                       same_regressor_for_all = TRUE,
                       market_model = "sim",
                       estimation_method = "ols",
                       estimation_start = as.Date("2019-04-01"),
                       estimation_end = as.Date("2020-03-13")) %>%
    car_lamb(car_start = as.Date("2020-03-16"),
             car_end = as.Date("2020-03-20"))

## End(Not run)
## The result of the code above is equivalent to:
data(securities_returns)
car_lamb(
    list_of_returns = securities_returns,
    car_start = as.Date("2020-03-16"),
    car_end = as.Date("2020-03-20")
)
```

car_nonparametric_tests

Returns the result of given event study nonparametric CAR tests.

Description

Performs given tests to examine the statistical significance of the CAR of a given period.

Usage

```
car_nonparametric_tests(
  list_of_returns,
  car_start,
  car_end,
  percentage = 90,
  all = TRUE,
  tests
)
```

Arguments

`list_of_returns`	a list of objects of S3 class `returns`, each element of which is treated as a security.
`car_start`	an object of `Date` class giving the first date of the CAR period.
`car_end`	an object of `Date` class giving the last date of the CAR period.
`percentage`	a lowest allowed percentage of non-missing observation for each day to be incorporated into CAR. The default value is 90 percent.
`all`	a logical value indicating whether all tests should be performed. The default value is `TRUE`. Note, only `car_rank_test` will be performed.
`tests`	a list of tests' functions. Currently, only `car_rank_test` is allowed.

Details

Currently, `car_nonparametric_tests` performs only `car_rank_test` test. This function was developed for the sake of completeness and can be used for future extensions of the package.

Value

A data frame of the following columns:

- `name`: a name of the test
- `car_start`: the first date of the CAR period
- `car_end`: the last date of the CAR period
- `average_percentage`: an average share of non-missing observations over the CAR period
- `statistic`: a test's statistic
- `number_of_days`: the number of days in the CAR period
- `significance`: a significance of the statistic

References

- Corrado C.J. *A Nonparametric Test for Abnormal Security-Price Performance in Event Studies*. Journal of Financial Economics 23:385-395, 1989.
- Cowan A.R. *Nonparametric Event Study Tests*. Review of Quantitative Finance and Accounting, 2:343-358, 1992.

See Also

`car_rank_test`.

Examples

```
## Not run:
library("magrittr")
rates_indx <- get_prices_from_tickers("^GSPC",
                                      start = as.Date("2019-04-01"),
                                      end = as.Date("2020-04-01"),
                                      quote = "Close",
                                      retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous")
tickers <- c("AMZN", "ZM", "UBER", "NFLX", "SHOP", "FB", "UPWK")
car_nonparam <- get_prices_from_tickers(tickers,
                                        start = as.Date("2019-04-01"),
                                        end = as.Date("2020-04-01"),
                                        quote = "Close",
                                        retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous") %>%
    apply_market_model(regressor = rates_indx,
                       same_regressor_for_all = TRUE,
                       market_model = "sim",
                       estimation_method = "ols",
                       estimation_start = as.Date("2019-04-01"),
                       estimation_end = as.Date("2020-03-13")) %>%
    car_nonparametric_tests(car_start = as.Date("2020-03-16"),
                            car_end = as.Date("2020-03-20"))

## End(Not run)
## The result of the code above is equivalent to:
data(securities_returns)
car_nonparam <- car_nonparametric_tests(
    list_of_returns = securities_returns,
    car_start = as.Date("2020-03-16"),
    car_end = as.Date("2020-03-20")
)
```

`car_parametric_tests` *Returns the result of given event study parametric CAR tests.*

Description

Performs given tests to examine whether cumulative abnormal return (CAR) significantly differs from zero.

Usage

```
car_parametric_tests(
  list_of_returns,
  car_start,
  car_end,
  percentage = 90,
  all = TRUE,
  tests
)
```

Arguments

`list_of_returns`	a list of objects of S3 class `returns`, each element of which is treated as a security.
`car_start`	an object of `Date` class giving the first date of the CAR period.
`car_end`	an object of `Date` class giving the last date of the CAR period.
`percentage`	a lowest allowed percentage of non-missing observation for each day to be incorporated into CAR. The default value is 90 percent.
`all`	a logical value indicating whether all tests should be performed. The default value is `TRUE`.
`tests`	a list of tests' functions among `car_brown_warner_1985` and `car_lamb`.

Details

`car_parametric_tests` performs specified tests among `car_brown_warner_1985` and `lamb` and returns a list of these tests' results. If `all = TRUE` (by default), the function ignores the value of `tests`.

Value

A data frame of the following columns:

- `name`: a name of the test
- `car_start`: the first date of the CAR period
- `car_end`: the last date of the CAR period
- `average_percentage`: an average share of non-missing observations over the CAR period
- `car_mean`: an average abnormal return over the CAR period
- `statistic`: a test's statistic
- `number_of_days`: the number of days in the CAR period
- `significance`: a significance of the statistic

References

- Brown S.J., Warner J.B. *Using Daily Stock Returns, The Case of Event Studies*. Journal of Financial Economics, 14:3-31, 1985.
- Lamb R.P. *An Exposure-Based Analysis of Property-Liability Insurer Stock Values around Hurricane Andrew*. Journal of Risk and Insurance, 62(1):111-123, 1995.

See Also

`car_brown_warner_1985` and `car_lamb`.

Examples

```
## Not run:
library("magrittr")
rates_indx <- get_prices_from_tickers("^GSPC",
                                      start = as.Date("2019-04-01"),
                                      end = as.Date("2020-04-01"),
                                      quote = "Close",
                                      retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous")
tickers <- c("AMZN", "ZM", "UBER", "NFLX", "SHOP", "FB", "UPWK")
car_param <- get_prices_from_tickers(tickers,
                                                 start = as.Date("2019-04-01"),
                                                 end = as.Date("2020-04-01"),
                                                 quote = "Close",
                                                 retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous") %>%
    apply_market_model(regressor = rates_indx,
                       same_regressor_for_all = TRUE,
                       market_model = "sim",
                       estimation_method = "ols",
                       estimation_start = as.Date("2019-04-01"),
                       estimation_end = as.Date("2020-03-13")) %>%
    car_parametric_tests(car_start = as.Date("2020-03-16"),
                         car_end = as.Date("2020-03-20"))

## End(Not run)
## The result of the code above is equivalent to:
data(securities_returns)
car_param <- car_parametric_tests(
    list_of_returns = securities_returns,
    car_start = as.Date("2020-03-16"),
    car_end = as.Date("2020-03-20")
)
```

`car_rank_test` *Cowan's CAR test.*

Description

A nonparametric test proposed by Cowan 1992 as an extension of the rank test proposed by Corrado 1989.

Usage

```
car_rank_test(list_of_returns, car_start, car_end, percentage = 90)
```

Arguments

`list_of_returns`	a list of objects of S3 class `returns`, each element of which is treated as a security.
`car_start`	an object of `Date` class giving the first date of the CAR period.
`car_end`	an object of `Date` class giving the last date of the CAR period.
`percentage`	a lowest allowed percentage of non-missing observation for each day to be incorporated into CAR. The default value is 90 percent.

- `name`: a name of the test, i.e. `"car_brown_warner_1985"`
- `car_start`: the first date of the CAR period
- `car_end`: the last date of the CAR period
- `average_percentage`: an average share of non-missing observations over the CAR period
- `statistic`: a test's statistic
- `number_of_days`: the number of days in the CAR period
- `significance`: a significance of the statistic

Details

This function performs a test proposed by Cowan 1992 to investigate the significance of the CAR for a given period. In order to get ranks of corresponding abnormal returns, the procedure uses regular R function `rank` with parameter `ties.method = "average"` and `na.last = "keep"`. For this test the estimation period and the event period must not overlap, otherwise an error will be thrown. The test statistic is assumed to have a normal distribution (as an approximation). The test is well-specified for the case, when cross-sectional abnormal returns are not symmetric. The test is stable to variance increase during given period. This test ignores the dependence of abnormal returns' ranks of different days (i.e., a serial dependence). The critical values are standard normal. The significance levels of α are 0.1, 0.05, and 0.01 (marked respectively by *, **, and ***).

References

- Corrado C.J. *A Nonparametric Test for Abnormal Security-Price Performance in Event Studies*. Journal of Financial Economics 23:385-395, 1989.
- Cowan A.R. *Nonparametric Event Study Tests*. Review of Quantitative Finance and Accounting, 2:343-358, 1992.

See Also

`car_nonparametric_tests`.

Examples

```
## Not run:
library("magrittr")
rates_indx <- get_prices_from_tickers("^GSPC",
                                      start = as.Date("2019-04-01"),
                                      end = as.Date("2020-04-01"),
                                      quote = "Close",
                                      retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous")
tickers <- c("AMZN", "ZM", "UBER", "NFLX", "SHOP", "FB", "UPWK")
get_prices_from_tickers(tickers,
                        start = as.Date("2019-04-01"),
                        end = as.Date("2020-04-01"),
                        quote = "Close",
                        retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous") %>%
    apply_market_model(regressor = rates_indx,
                       same_regressor_for_all = TRUE,
                       market_model = "sim",
                       estimation_method = "ols",
                       estimation_start = as.Date("2019-04-01"),
                       estimation_end = as.Date("2020-03-13")) %>%
    car_rank_test(car_start = as.Date("2020-03-16"),
                  car_end = as.Date("2020-03-20"))

## End(Not run)
## The result of the code above is equivalent to:
data(securities_returns)
car_rank_test(
    list_of_returns = securities_returns,
    car_start = as.Date("2020-03-16"),
    car_end = as.Date("2020-03-20")
)
```

`corrado_sign_test` *Corrado's sign test (1992).*

Description

An event study nonparametric test described in Corrado and Zivney 1992.

Usage

```
corrado_sign_test(list_of_returns, event_start, event_end)
```

Arguments

`list_of_returns`	a list of objects of S3 class `returns`, each element of which is treated as a security.
`event_start`	an object of `Date` class giving the first date of the event period.
`event_end`	an object of `Date` class giving the last date of the event period.

Details

Performs a nonparametric test for the event study, which is described in Corrado and Zivney 1992. This test is similar to procedure, described in Brown and Warner 1985 (t-ratio), but instead of using abnormal returns, the test uses $G_{i,t} = sign(A_{i,t} - median(A_i))$. `sign` and `median` are ones, which have the same definition as R functions. For this test the estimation period and the event period must not overlap, otherwise an error will be thrown. The sign test procedure avoids the misspecification of tests, which assumes symmetry around zero of abnormal returns (the median equals to zero). For a single day the performance of this test is proven to be better than classical Brown and Warner's test (without event-induced variance). This test is dominated by rank test. The significance levels of α are 0.1, 0.05, and 0.01 (marked respectively by *, **, and ***).

Value

A data frame of the following columns:

- `date`: a calendar date
- `weekday`: a day of the week
- `percentage`: a share of non-missing observations for a given day
- `csign_stat`: a Corrado's sign test statistic
- `csign_signif`: a significance of the statistic

References

Corrado C.J., Zivney T.L. *The Specification and Power of the Sign Test in Event Study Hypothesis Tests Using Daily Stock Returns*. Journal of Financial and Quantitative Analysis, 27(3):465-478, 1992.

See Also

nonparametric_tests, sign_test, generalized_sign_test, rank_test, modified_rank_test, and wilcoxon_test.

Examples

```
## Not run:
library("magrittr")
rates_indx <- get_prices_from_tickers("^GSPC",
                                      start = as.Date("2019-04-01"),
                                      end = as.Date("2020-04-01"),
                                      quote = "Close",
                                      retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous")
tickers <- c("AMZN", "ZM", "UBER", "NFLX", "SHOP", "FB", "UPWK")
get_prices_from_tickers(tickers,
                        start = as.Date("2019-04-01"),
                        end = as.Date("2020-04-01"),
                        quote = "Close",
                        retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous") %>%
    apply_market_model(regressor = rates_indx,
                       same_regressor_for_all = TRUE,
                       market_model = "sim",
                       estimation_method = "ols",
                       estimation_start = as.Date("2019-04-01"),
                       estimation_end = as.Date("2020-03-13")) %>%
    corrado_sign_test(event_start = as.Date("2020-03-16"),
                      event_end = as.Date("2020-03-20"))

## End(Not run)
## The result of the code above is equivalent to:
data(securities_returns)
corrado_sign_test(list_of_returns = securities_returns,
                  event_start =  as.Date("2020-03-16"),
                  event_end = as.Date("2020-03-20"))
```

generalized_sign_test *An event study binomial sign test.*

Description

A binomial sign test which determines whether the frequency of positive abnormal returns in the event period is significantly different from the frequency in the estimation period.

Usage

```
generalized_sign_test(list_of_returns, event_start, event_end)
```

Arguments

`list_of_returns`	a list of objects of S3 class `returns`, each element of which is treated as a security.
`event_start`	an object of `Date` class giving the first date of the event period.
`event_end`	an object of `Date` class giving the last date of the event period.

Details

This test is application of the binomial test to the event study, which indicates whether the cross-sectional frequency of positive abnormal returns is significantly different from the expected. This test is stable to outliers, in other words allows for checking if the result is driven by few companies with extremely large abnormal performance. For this test the estimation period and the event period must not overlap, otherwise an error will be thrown. This test uses an estimate from the estimation period instead of using naive value of expected frequency 0.5. The test statistic is assumed to have a normal distribution. Typically the test is used together with parametric tests. The test is well-specified for the case, when cross-sectional abnormal returns are not symmetric. Also this procedure is less sensitive to extreme returns than the rank test. The significance levels of α are 0.1, 0.05, and 0.01 (marked respectively by *, **, and ***).

Value

A data frame of the following columns:

- `date`: a calendar date
- `weekday`: a day of the week
- `percentage`: a share of non-missing observations for a given day
- `gsign_stat`: a generalized sign test statistic
- `gsign_signif`: a significance of the statistic

References

- McConnell J.J., Muscarella C.J. *Capital expenditure plans and firm value* Journal of Financial Economics, 14:399-422, 1985.
- Cowan A.R. *Nonparametric Event Study Tests*. Review of Quantitative Finance and Accounting, 2:343-358, 1992.

See Also

`nonparametric_tests`, `sign_test`, `corrado_sign_test`, `rank_test`, `modified_rank_test`, and `wilcoxon_test`.

Examples

```
## Not run:
library("magrittr")
rates_indx <- get_prices_from_tickers("^GSPC",
                                      start = as.Date("2019-04-01"),
                                      end = as.Date("2020-04-01"),
                                      quote = "Close",
                                      retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous")
tickers <- c("AMZN", "ZM", "UBER", "NFLX", "SHOP", "FB", "UPWK")
get_prices_from_tickers(tickers,
                        start = as.Date("2019-04-01"),
                        end = as.Date("2020-04-01"),
                        quote = "Close",
                        retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous") %>%
    apply_market_model(regressor = rates_indx,
                       same_regressor_for_all = TRUE,
                       market_model = "sim",
                       estimation_method = "ols",
                       estimation_start = as.Date("2019-04-01"),
                       estimation_end = as.Date("2020-03-13")) %>%
    generalized_sign_test(event_start = as.Date("2020-03-16"),
                          event_end = as.Date("2020-03-20"))

## End(Not run)
## The result of the code above is equivalent to:
data(securities_returns)
generalized_sign_test(list_of_returns = securities_returns,
                      event_start =  as.Date("2020-03-16"),
                      event_end = as.Date("2020-03-20"))
```

`get_prices_from_tickers` *Get daily prices of securities.*

Description

Returns daily Open or Close prices between `start` and end date for given tickers.

Usage

```
get_prices_from_tickers(
  ...,
  start,
  end,
  quote = c("Open", "Close"),
  retclass = c("list", "zoo", "data.frame")
)
```

Arguments

`...`	character vectors indicating tickers (should be valid in Yahoo Finance).
`start`	an object of `Date` class specifying the first date of the observed time period.
`end`	an object of `Date` class specifying the last date of the observed time period.
`quote`	a character indicating the type of the price: `"Open"` (default) or `"Close"`.
`retclass`	a character specifying the return class: `"list"` (default), `"zoo"` or `"data.frame"`.

Details

This function uses the function `getSymbols` form the `quantmod` package. The provider is set automatically to Yahoo Finance. The function returns the data in different class-containers: list of `zoo`'s, `zoo`, or `data.frame`.

Value

Prices of securities as `"list"`, `"zoo"`, or `"data.frame"`.

See Also

`getSymbols`

Examples

```
## Download historical prices of seven companies' stocks:
## Not run:
library("magrittr")
tickers <- c("AMZN", "ZM", "UBER", "NFLX", "SHOP", "FB", "UPWK")
prices <- tickers %>%
    get_prices_from_tickers(start = as.Date("2019-04-01"),
                            end = as.Date("2020-04-01"),
                            quote = "Close",
                            retclass = "zoo")

## End(Not run)
## The result of the above code is stored in:
data(prices)

## Download historical prices of S&P 500 index:
## Not run:
prices_indx <- get_prices_from_tickers("^GSPC",
                                       start = as.Date("2019-04-01"),
                                       end = as.Date("2020-04-01"),
                                       quote = "Close",
                                              retclass = "zoo")

## End(Not run)
## The result of the above code is stored in:
data(prices_indx)
```

`get_rates_from_prices` *Calculate rates of return for given prices.*

Description

`get_rates_from_prices` is used for computing rates of return from prices for different classes.

Usage

```
get_rates_from_prices(
  prices,
  quote = c("Open", "Close"),
  multi_day = TRUE,
  compounding = c("discrete", "continuous")
)
```

Arguments

`prices`	an object containing prices of securities. Three classes are allowed: `list`, `data.frame`, and `zoo`.
`quote`	a character vector specifying the type of the quote: `"Open"` (default) or `"Close"`.
`multi_day`	logical, is a rate of return between more than 1 day is allowed?
`compounding`	a character vector defining the type of compounding: `"continuous"` (default) or `"discrete"`.

Details

This is a generic function, dispatched for such classes as `list`, `data.frame`, and `zoo` that represent prices.

The calculation is made in C++ (`Rcpp`) in favor of speed.

If `prices` is a data frame, than the first column should be of the class `Date` and contain ordered dates of prices.

The correspondence between dates and values of the rates depends on the quote, which can be either Open or Close. If the quote is Open, than the value of rate belongs to the former date. Otherwise, to the latter one. This is also applied for the algorithm, if multiday is allowed: the value of the rate of return is assigned to the latter day in case of Close price, and to the former day in in case of Open quote.

The `multi_day` parameter specifies how to handle missing values and weekends. If the value is TRUE, the function ignores missing values and the rates are calculated between non-missing prices.

If it is FALSE, then only one-day period rates of return are computed (between two consecutive calendar dates).

The function uses either continuous (by default) or discrete (periodic) compounding.

Value

Rates of returns of the same class as prices.

Examples

```
## Download historical prices of seven companies' stocks and estimate rates
## of returns form prices:
## Not run:
library("magrittr")
tickers <- c("AMZN", "ZM", "UBER", "NFLX", "SHOP", "FB", "UPWK")
rates <- tickers %>%
    get_prices_from_tickers(start = as.Date("2019-04-01"),
                            end = as.Date("2020-04-01"),
                            quote = "Close",
                            retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous")

## End(Not run)
## The result of the above code is stored in:
data(rates)

## Download historical prices of S&P 500 index and estimate rates of
## returns from prices:
## Not run:
library("magrittr")
rates_indx <- get_prices_from_tickers("^GSPC",
                                      start = as.Date("2019-04-01"),
                                      end = as.Date("2020-04-01"),
                                      quote = "Close",
                                      retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous")

## End(Not run)
## The result of the above code is stored in:
data(rates_indx)
```

`lamb` *Lamb's parametric test (1995).*

Description

An event study parametric test described in Lamb 1995.

Usage

```
lamb(list_of_returns, event_start, event_end)
```

Arguments

`list_of_returns`	a list of objects of S3 class `returns`, each element of which is treated as a security.
`event_start`	an object of `Date` class giving the first date of the event period.
`event_end`	an object of `Date` class giving the last date of the event period.

Details

Performs a parametric test for the event study, which is described in Lamb 1995. The author refers to Warner and Brown 1985 and Henderson 1990. However, this test was not observed in neither papers. The test statistics are very close to the statistics produced by `brown_warner_1985` and typically has the same significance. The test examines the hypothesis whether the theoretical cross-sectional expected value for a given day is equal to zero. It calculates statistics even if event window and estimation period are overlapped (intersect). The critical values are standard normal. The significance levels of α are 0.1, 0.05, and 0.01 (marked respectively by *, **, and ***).

Value

A data frame of the following columns:

- `date`: a calendar date
- `weekday`: a day of the week
- `percentage`: a share of non-missing observations for a given day
- `mean`: an average abnormal return
- `lmb_stat`: a Lamb's test statistic
- `lmb_signif`: a significance of the statistic

References

Lamb R.P. *An Exposure-Based Analysis of Property-Liability Insurer Stock Values around Hurricane Andrew*. Journal of Risk and Insurance, 62(1):111-123, 1995.

See Also

`parametric_tests`, `brown_warner_1980`, `brown_warner_1985`, `t_test`,`patell` and `boehmer`.

Examples

```
## Not run:
library("magrittr")
rates_indx <- get_prices_from_tickers("^GSPC",
                                      start = as.Date("2019-04-01"),
                                      end = as.Date("2020-04-01"),
                                      quote = "Close",
                                      retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous")
tickers <- c("AMZN", "ZM", "UBER", "NFLX", "SHOP", "FB", "UPWK")
get_prices_from_tickers(tickers,
                        start = as.Date("2019-04-01"),
                        end = as.Date("2020-04-01"),
                        quote = "Close",
                        retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous") %>%
    apply_market_model(regressor = rates_indx,
                       same_regressor_for_all = TRUE,
                       market_model = "sim",
                       estimation_method = "ols",
                       estimation_start = as.Date("2019-04-01"),
                       estimation_end = as.Date("2020-03-13")) %>%
    lamb(event_start = as.Date("2020-03-16"),
         event_end = as.Date("2020-03-20"))

## End(Not run)
## The result of the code above is equivalent to:
data(securities_returns)
lamb(list_of_returns = securities_returns,
     event_start =  as.Date("2020-03-16"),
     event_end = as.Date("2020-03-20"))
```

modified_rank_test *An event study modified rank test.*

Description

The test is the modification of the original rank test, proposed by Corrado 1989. This test is adapted to missing values in abnormal returns.

Usage

```
modified_rank_test(list_of_returns, event_start, event_end)
```

Arguments

`list_of_returns`	a list of objects of S3 class `returns`, each element of which is treated as a security.
`event_start`	an object of `Date` class giving the first date of the event period.
`event_end`	an object of `Date` class giving the last date of the event period.

Details

In addition to the original rank test, the procedure divides corresponding ranks by the number of nonmissing returns plus one for each security. This leads to order statistics with uniform distribution. In limit overall statistics under a null hypothesis is approximately normally distributed. For this test the estimation period and the event period must not overlap, otherwise an error will be thrown. The test is well-specified for the case, when cross-sectional abnormal returns are not symmetric. The test is stable to variance increase during the event window. This test is more sensitive to extreme values than the sign test. The significance levels of α are 0.1, 0.05, and 0.01 (marked respectively by *, **, and ***).

Value

A data frame of the following columns:

- `date`: a calendar date
- `weekday`: a day of the week
- `percentage`: a share of non-missing observations for a given day
- `mrank_stat`: a modified rank test statistic
- `mrank_signif`: a significance of the statistic

References

- Corrado C.J., Zivney T.L. *The Specification and Power of the Sign Test in Event Study Hypothesis Tests Using Daily Stock Returns*. Journal of Financial and Quantitative Analysis, 27(3):465-478, 1992.
- Kolari J.W., Pynnonen S. *Event Study Testing with Cross-sectional Correlation of Abnormal Returns*. The Review of Financial Studies, 23(11):3996-4025, 2010.

See Also

`nonparametric_tests`,`sign_test`, `generalized_sign_test`, `corrado_sign_test`, `rank_test`, and `wilcoxon_test`.

Examples

```
## Not run:
library("magrittr")
rates_indx <- get_prices_from_tickers("^GSPC",
                                      start = as.Date("2019-04-01"),
                                      end = as.Date("2020-04-01"),
                                      quote = "Close",
                                      retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous")
tickers <- c("AMZN", "ZM", "UBER", "NFLX", "SHOP", "FB", "UPWK")
get_prices_from_tickers(tickers,
                        start = as.Date("2019-04-01"),
                        end = as.Date("2020-04-01"),
                        quote = "Close",
                        retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous") %>%
    apply_market_model(regressor = rates_indx,
                       same_regressor_for_all = TRUE,
                       market_model = "sim",
                       estimation_method = "ols",
                       estimation_start = as.Date("2019-04-01"),
                       estimation_end = as.Date("2020-03-13")) %>%
    modified_rank_test(event_start = as.Date("2020-03-16"),
                       event_end = as.Date("2020-03-20"))

## End(Not run)
## The result of the code above is equivalent to:
data(securities_returns)
modified_rank_test(list_of_returns = securities_returns,
                   event_start =  as.Date("2020-03-16"),
                   event_end = as.Date("2020-03-20"))
```

`nonparametric_tests` *Returns the result of given event study nonparametric tests.*

Description

Performs main nonparametric tests for each date in the event window and returns a data frame of their statistics and significance.

Usage

```
nonparametric_tests(list_of_returns, event_start, event_end, all = TRUE, tests)
```

Arguments

`list_of_returns`	a list of objects of S3 class `returns`, each element of which is treated as a security.
`event_start`	an object of `Date` class giving the first date of the event period.

`event_end`	an object of `Date` class giving the last date of the event period.
`all`	a logical vector of length one indicating whether all tests should be performed. The default value is `TRUE`.
`tests`	a list of tests' functions among `sign_test`, `generalized_sign_test`, `corrado_sign_test`, `rank_test`, `modified_rank_test`, and `wilcoxon_test`.

Details

`nonparametric_tests` performs given tests among `sign_test`, `generalized_sign_test`, `corrado_sign_test`, `rank_test`, `modified_rank_test`, `wilcoxon_test`, and merge result to a single data frame. If `all = TRUE` (the default value), the function ignores the value of `tests`.

Value

A data frame of the following columns:

- `date`: a calendar date
- `weekday`: a day of the week
- `percentage`: a share of non-missing observations for a given day
- Various tests' statistics and significance

References

- Corrado C.J., Zivney T.L. *The Specification and Power of the Sign Test in Event Study Hypothesis Tests Using Daily Stock Returns*. Journal of Financial and Quantitative Analysis, 27(3):465-478, 1992.
- McConnell J.J., Muscarella C.J. *Capital expenditure plans and firm value* Journal of Financial Economics, 14:399-422, 1985.
- Boehmer E., Musumeci J., Poulsen A.B. *Event-study methodology under conditions of event-induced variance*. Journal of Financial Economics, 30(2):253-272, 1991.
- Cowan A.R. *Nonparametric Event Study Tests*. Review of Quantitative Finance and Accounting, 2:343-358, 1992.
- Corrado C.J. *A Nonparametric Test for Abnormal Security-Price Performance in Event Studies*. Journal of Financial Economics 23:385-395, 1989.
- Campbell C.J., Wasley C.E. *Measuring Security Price Performance Using Daily NASDAQ Returns*. Journal of Financial Economics 33:73-92, 1993.
- Savickas R. *Event-Induced Volatility and Tests for Abnormal Performance*. The Journal of Financial Research, 26(2):156-178, 2003.
- Kolari J.W., Pynnonen S. *Event Study Testing with Cross-sectional Correlation of Abnormal Returns*. The Review of Financial Studies, 23(11):3996-4025, 2010.
- Wilcoxon F. *Individual Comparisons by Ranking Methods*. Biometrics Bulletin 1(6):80-83, 1945.
- Lehmann E.L, *Nonparametrics: Statistical Methods Based on Ranks*. San Francisco: Holden-Day, 1975.
- Hollander M., Wolfe D.A. *Nonparametric Statistical Methods*. New York: John Wiley & Sons, 1973.

See Also

sign_test, generalized_sign_test, corrado_sign_test, rank_test, modified_rank_test, and wilcoxon_test.

Examples

```
## Not run:
library("magrittr")
rates_indx <- get_prices_from_tickers("^GSPC",
                                      start = as.Date("2019-04-01"),
                                      end = as.Date("2020-04-01"),
                                      quote = "Close",
                                      retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous")
tickers <- c("AMZN", "ZM", "UBER", "NFLX", "SHOP", "FB", "UPWK")
nparam <- get_prices_from_tickers(tickers,
                                  start = as.Date("2019-04-01"),
                                  end = as.Date("2020-04-01"),
                                  quote = "Close",
                                  retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous") %>%
    apply_market_model(regressor = rates_indx,
                       same_regressor_for_all = TRUE,
                       market_model = "sim",
                       estimation_method = "ols",
                       estimation_start = as.Date("2019-04-01"),
                       estimation_end = as.Date("2020-03-13")) %>%
    nonparametric_tests(event_start = as.Date("2020-03-16"),
                        event_end = as.Date("2020-03-20"))

## End(Not run)
## The result of the code above is equivalent to:
data(securities_returns)
nparam <- nonparametric_tests(list_of_returns = securities_returns,
                              event_start = as.Date("2020-03-16"),
                              event_end = as.Date("2020-03-20"))
```

parametric_tests *Returns the result of given event study parametric tests.*

Description

Performs main parametric tests for each date in the event window and returns a data frame of their statistics and significance.

Usage

```
parametric_tests(list_of_returns, event_start, event_end, all = TRUE, tests)
```

Arguments

`list_of_returns`	a list of objects of S3 class `returns`, each element of which is treated as a security.
`event_start`	an object of `Date` class giving the first date of the event period.
`event_end`	an object of `Date` class giving the last date of the event period.
`all`	a logical vector of length one indicating whether all tests should be performed. The default value is `TRUE`.
`tests`	a list of tests' functions among `brown_warner_1980`, `brown_warner_1985`, `t_test`, `patell`, `boehmer`, and `lamb`.

Details

`parametric_tests` performs given tests among `brown_warner_1980`, `brown_warner_1985`, `t_test`, `patell`, `boehmer`, `lamb` and merge result to a single data frame. If `all = TRUE` (the default value), the function ignores the value of `tests`.

Value

A data frame of the following columns:

- `date`: a calendar date
- `weekday`: a day of the week
- `percentage`: a share of non-missing observations for a given day
- `mean`: an average abnormal return
- Various tests' statistics and significance

References

- Brown S.J., Warner J.B. *Measuring security price performance*. Journal of Financial Economics, 8:205-258, 1980.
- Brown S.J., Warner J.B. *Using Daily Stock Returns, The Case of Event Studies*. Journal of Financial Economics, 14:3-31, 1985.
- Boehmer E., Musumeci J., Poulsen A.B. *Event-study methodology under conditions of event-induced variance*. Journal of Financial Economics, 30(2):253-272, 1991.
- Patell J.M. *Corporate forecasts of earnings per share and stock price behavior: empirical tests*. Journal of Accounting Research, 14(2):246- 276, 1976.
- Lamb R.P. *An Exposure-Based Analysis of Property-Liability Insurer Stock Values around Hurricane Andrew*. Journal of Risk and Insurance, 62(1):111-123, 1995.

See Also

`brown_warner_1980`, `brown_warner_1985`, `t_test`, `patell`, `boehmer`, and `lamb`.

Examples

```
## Not run:
library("magrittr")
rates_indx <- get_prices_from_tickers("^GSPC",
                                      start = as.Date("2019-04-01"),
                                      end = as.Date("2020-04-01"),
                                      quote = "Close",
                                      retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous")
tickers <- c("AMZN", "ZM", "UBER", "NFLX", "SHOP", "FB", "UPWK")
param <- get_prices_from_tickers(tickers,
                                 start = as.Date("2019-04-01"),
                                 end = as.Date("2020-04-01"),
                                 quote = "Close",
                                 retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous") %>%
    apply_market_model(regressor = rates_indx,
                       same_regressor_for_all = TRUE,
                       market_model = "sim",
                       estimation_method = "ols",
                       estimation_start = as.Date("2019-04-01"),
                       estimation_end = as.Date("2020-03-13")) %>%
    parametric_tests(event_start = as.Date("2020-03-16"),
                     event_end = as.Date("2020-03-20"))

## End(Not run)
## The result of the code above is equivalent to:
data(securities_returns)
param <- parametric_tests(list_of_returns = securities_returns,
                          event_start = as.Date("2020-03-16"),
                          event_end = as.Date("2020-03-20"))
```

`patell` *Patell's parametric test (1976).*

Description

An event study parametric test described in Patell 1976.

Usage

```
patell(list_of_returns, event_start, event_end)
```

Arguments

`list_of_returns`	a list of objects of S3 class `returns`, each element of which is treated as a security.
`event_start`	an object of `Date` class giving the first date of the event period.
`event_end`	an object of `Date` class giving the last date of the event period.

Details

Performs a parametric test for event study, which is described in Patell 1976, which is called standardized-residuals method in Boehmer 1991. Test's assumptions are a cross-sectional independence and an insignificance of an event-induced variance. The standardization smooths the effect of the event-induced variance comparing to Brown and Warner tests. Also standardization incorporates the situation, when a highly volatile security dominates the test. The test examines the hypothesis whether the theoretical cross-sectional expected value for a given day is equal to zero. It calculates statistics even if event window and estimation period are overlapped (intersect). The critical values are standard normal. The significance levels of α are 0.1, 0.05, and 0.01 (marked respectively by *, **, and ***).

Value

A data frame of the following columns:

- `date`: a calendar date
- `weekday`: a day of the week
- `percentage`: a share of non-missing observations for a given day
- `mean`: an average abnormal return
- `pt_stat`: a Patell's test statistic
- `pt_signif`: a significance of the statistic

References

- Patell J.M. *Corporate forecasts of earnings per share and stock price behavior: empirical tests*. Journal of Accounting Research, 14(2):246- 276, 1976.
- Boehmer E., Musumeci J., Poulsen A.B. *Event-study methodology under conditions of event-induced variance*. Journal of Financial Economics, 30(2):253-272, 1991.

See Also

`parametric_tests`, `brown_warner_1980`, `brown_warner_1985`, `t_test`, and `boehmer`, and `lamb`.

Examples

```
## Not run:
library("magrittr")
rates_indx <- get_prices_from_tickers("^GSPC",
                                      start = as.Date("2019-04-01"),
                                        end = as.Date("2020-04-01"),
                                        quote = "Close",
                                        retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous")
tickers <- c("AMZN", "ZM", "UBER", "NFLX", "SHOP", "FB", "UPWK")
get_prices_from_tickers(tickers,
                        start = as.Date("2019-04-01"),
                        end = as.Date("2020-04-01"),
                        quote = "Close",
                        retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous") %>%
    apply_market_model(regressor = rates_indx,
                       same_regressor_for_all = TRUE,
                       market_model = "sim",
                       estimation_method = "ols",
                       estimation_start = as.Date("2019-04-01"),
                       estimation_end = as.Date("2020-03-13")) %>%
    patell(event_start = as.Date("2020-03-16"),
           event_end = as.Date("2020-03-20"))

## End(Not run)
## The result of the code above is equivalent to:
data(securities_returns)
patell(list_of_returns = securities_returns,
       event_start =  as.Date("2020-03-16"),
       event_end = as.Date("2020-03-20"))
```

`prices` *Stock prices of seven companies from 2019-04-01 to 2020-04-01*

Description

A zoo object of seven columns containing daily Close stock prices from 2019-04-01 to 2020-04-01 of seven companies, which could profit from COVID-19 lockdown. See examples of `get_prices_from_tickers` for the dataset generation.

Usage

```
prices
```

Format

A zoo object of eight columns:

- AMZN

- ZM
- UBER
- NFLX
- SHOP
- FB
- UPWK

`prices_indx` *Prices of S&P 500 index from 2019-04-01 to 2020-04-01*

Description

A `zoo` object containing daily prices of S&P 500 index from 2019-04-01 to 2020-04-01. See examples of `get_prices_from_tickers` for the dataset generation.

Usage

```
prices_indx
```

Format

A `zoo` object.

`rank_test` *An event study rank test.*

Description

An original rank test applied to an event study, which is based on Wilcoxon (1945) rank test.

Usage

```
rank_test(list_of_returns, event_start, event_end)
```

Arguments

`list_of_returns`	a list of objects of S3 class `returns`, each element of which is treated as a security.
`event_start`	an object of `Date` class giving the first date of the event period.
`event_end`	an object of `Date` class giving the last date of the event period.

Details

This procedure uses ranks of abnormal returns to examine significance of each day in the event window. In order to get ranks of corresponding abnormal returns, the procedure uses regular R function `rank` with parameter `ties.method = "average"` and `na.last = "keep"`. For this test the estimation period and the event period must not overlap, otherwise an error will be thrown. The test statistic is assumed to have a normal distribution (as an approximation). The test is well-specified for the case, when cross-sectional abnormal returns are not symmetric. The test is stable to variance increase during event window. This test is more sensitive to extreme values than sign test. For data with missing data see the `modified_rank_test`. The significance levels of α are 0.1, 0.05, and 0.01 (marked respectively by *, **, and ***).

Value

A data frame of the following columns:

- `date`: a calendar date
- `weekday`: a day of the week
- `percentage`: a share of non-missing observations for a given day
- `rank_stat`: a rank test statistic
- `rank_signif`: a significance of the statistic

References

- Corrado C.J. *A Nonparametric Test for Abnormal Security-Price Performance in Event Studies*. Journal of Financial Economics 23:385-395, 1989.
- Cowan A.R. *Nonparametric Event Study Tests*. Review of Quantitative Finance and Accounting, 2:343-358, 1992.
- Campbell C.J., Wasley C.E. *Measuring Security Price Performance Using Daily NASDAQ Returns*. Journal of Financial Economics 33:73-92, 1993.
- Savickas R. *Event-Induced Volatility and Tests for Abnormal Performance*. The Journal of Financial Research, 26(2):156-178, 2003.

See Also

`nonparametric_tests`,`sign_test`, `generalized_sign_test`, `corrado_sign_test`, `modified_rank_test`, and `wilcoxon_test`.

Examples

```
## Not run:
library("magrittr")
rates_indx <- get_prices_from_tickers("^GSPC",
                                      start = as.Date("2019-04-01"),
                                      end = as.Date("2020-04-01"),
                                      quote = "Close",
                                      retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                              compounding = "continuous")
tickers <- c("AMZN", "ZM", "UBER", "NFLX", "SHOP", "FB", "UPWK")
get_prices_from_tickers(tickers,
                        start = as.Date("2019-04-01"),
                        end = as.Date("2020-04-01"),
                        quote = "Close",
                        retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous") %>%
    apply_market_model(regressor = rates_indx,
                       same_regressor_for_all = TRUE,
                       market_model = "sim",
                       estimation_method = "ols",
                       estimation_start = as.Date("2019-04-01"),
                       estimation_end = as.Date("2020-03-13")) %>%
    rank_test(event_start = as.Date("2020-03-16"),
              event_end = as.Date("2020-03-20"))

## End(Not run)
## The result of the code above is equivalent to:
data(securities_returns)
rank_test(list_of_returns = securities_returns,
          event_start =  as.Date("2020-03-16"),
          event_end = as.Date("2020-03-20"))
```

`rates` *Rates of returns of seven companies from 2019-04-01 to 2020-04-01*

Description

A zoo object of seven columns containing daily rates of returns from 2019-04-01 to 2020-04-01 of seven companies, which could profit from COVID-19 lockdown. See examples of `get_rates_from_prices` for the dataset generation.

Usage

```
rates
```

Format

A zoo object of eight columns:

- AMZN
- ZM
- UBER
- NFLX

- SHOP
- FB
- UPWK

`rates_indx` *Rates of returns of S&P 500 index from 2019-04-01 to 2020-04-01*

Description

A zoo object containing daily rates of returns of S&P 500 index from 2019-04-01 to 2020-04-01. See examples of `get_rates_from_prices` for the dataset generation.

Usage

```
rates_indx
```

Format

A zoo object.

`returns` *Constructor of an object of S3 class* `returns`.

Description

Constructs an object of S3 class `returns`.

Usage

```
returns(
  rates,
  regressor,
  market_model = c("mean_adj", "mrkt_adj", "sim"),
  estimation_method = c("ols"),
  estimation_start,
  estimation_end
)
```

Arguments

`rates`	an object of class either `zoo` or `data.frame` giving observed rates of returns of security.
`regressor`	an object of the same class as `rates` representing rates of returns of the market model, if needed.
`market_model`	a character indicating the market model among `mean_adj`, `mrkt_adj`, and `sim`.
`estimation_method`	a character specifying an estimation method for `sim` model.
`estimation_start`	an object of `Date` class giving the first date of the estimation period.
`estimation_end`	an object of `Date` class giving the last date of the estimation period.

Details

The constructor is a generic function, dispatched for classes `zoo` `data.frame`. Parameters `rates` and `regressor` should be objects of the same class (`zoo` or `data.frame`). There are three market model implemented. `mean_adj` stands for mean-adjusted-returns model, which is the average of returns during the estimation period. `mrkt_adj` represents market-adjusted-returns model: the securities' rates of returns are simply market index rates of returns (in terms of parameters - `regressor`). Finally, `sim` stands for single-index market model For this model only Ordinary Least Squares `estimation_method` is currently implemented. All models are described in Brown and Warner (1985).

Value

An object of S3 class `returns`, which contains following fields:

- observed: an object of `zoo` class containing observed rates of returns.
- predicted: an object of `zoo` class containing predicted by a market model rates of returns.
- lower95CI: a lower bound of the 95% Confidence Interval for predicted rates of returns.
- upper95CI: an upper bound of the 95% Confidence Interval for predicted rates of returns.
- abnormal: an object of `zoo` class containing abnormal returns.
- regressor: an object of `zoo` class containing rates of regressor (typically market index).
- market_model: a code name of the market model.
- full_name_market_model: a full name of the market model.
- estimation_method: a code name of the estimation method (applied only for SIM).
- full_name_estimation_method: a full name of the estimation method (applied only for SIM).
- coefficients: coefficients α and β for SIM market model (applied only for SIM).
- estimation_start: a start date of the estimation period.
- estimation_end: an end date of the estimation period.
- estimation_length: a length of the estimation period.

References

Brown S.J., Warner J.B. *Using Daily Stock Returns, The Case of Event Studies*. Journal of Financial Economics, 14:3-31, 1985.

See Also

apply_market_model

Examples

```
library("zoo")
## 1. Mean-adjusted-returns model
## Not run:
library("magrittr")
single_return <- get_prices_from_tickers("AMZN",
                                         start = as.Date("2019-04-01"),
                                         end = as.Date("2020-04-01"),
                                         quote = "Close",
                                         retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous") %>%
    returns(market_model = "mean_adj",
            estimation_start = as.Date("2019-04-01"),
            estimation_end = as.Date("2020-03-13"))

## End(Not run)
## The result of the code above is equivalent to:
data(rates)
single_return <- returns(rates[, "AMZN"],
                         market_model = "mean_adj",
                         estimation_start = as.Date("2019-04-01"),
                         estimation_end = as.Date("2020-03-13"))

## 2. Market-adjusted-returns model
## Not run:
library("magrittr")
rates_indx <- get_prices_from_tickers("^GSPC",
                                      start = as.Date("2019-04-01"),
                                      end = as.Date("2020-04-01"),
                                      quote = "Close",
                                      retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous")

single_return <- get_prices_from_tickers("AMZN",
                                         start = as.Date("2019-04-01"),
                                         end = as.Date("2020-04-01"),
                                         quote = "Close",
                                         retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous") %>%
    returns(regressor = rates_indx,
            market_model = "mrkt_adj",
            estimation_start = as.Date("2019-04-01"),
            estimation_end = as.Date("2020-03-13"))

## End(Not run)
## The result of the code above is equivalent to:
data(rates, rates_indx)
single_return <- returns(rates = rates[, "AMZN", drop = FALSE],
                         regressor = rates_indx,
                         market_model = "mrkt_adj",
                         estimation_method = "ols",
                         estimation_start = as.Date("2019-04-01"),
                         estimation_end = as.Date("2020-03-13"))

## 3. Single-index market model
## Not run:
library("magrittr")
rates_indx <- get_prices_from_tickers("^GSPC",
                                      start = as.Date("2019-04-01"),
                                      end = as.Date("2020-04-01"),
                                      quote = "Close",
                                      retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous")

single_return <- get_prices_from_tickers("AMZN",
                                         start = as.Date("2019-04-01"),
                                         end = as.Date("2020-04-01"),
                                         quote = "Close",
                                         retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous") %>%
    returns(regressor = rates_indx,
            market_model = "sim",
            estimation_method = "ols",
            estimation_start = as.Date("2019-04-01"),
            estimation_end = as.Date("2020-03-13"))

## End(Not run)
## The result of the code above is equivalent to:
data(rates, rates_indx)
single_return <- returns(rates = rates[, "AMZN", drop = FALSE],
                         regressor = rates_indx,
                         market_model = "sim",
                         estimation_method = "ols",
                         estimation_start = as.Date("2019-04-01"),
                         estimation_end = as.Date("2020-03-13"))
```

`run_app` *Run Shiny demo app*

Description

The function `run_app()` launches a Shiny app, which is a GUI wrapper of crippled version of `{estudy2}`. This app is developed exclusively for demonstration purposes.

Usage

```
run_app()
```

Details

The app is run locally.

`securities_returns` *Returns of seven companies from 2019-04-01 to 2020-04-01*

Description

A list of length seven, elements of which are objects of the class `returns`. The list contains all necessary returns from 2019-04-01 to 2020-04-01 of seven companies, which could profit from COVID-19 lockdown. See examples of `apply_market_model` for the dataset generation.

Usage

```
securities_returns
```

Format

A list of eight `zoo` elements:

- AMZN
- ZM
- UBER
- NFLX
- SHOP
- FB
- UPWK

`sign_test` *An event study simple binomial sign test.*

Description

A binomial sign test which determines whether the frequency of positive abnormal returns in the event period is significantly different from one-half.

Usage

```
sign_test(list_of_returns, event_start, event_end)
```

Arguments

`list_of_returns`	a list of objects of S3 class `returns`, each element of which is treated as a security.
`event_start`	an object of `Date` class giving the first date of the event period.
`event_end`	an object of `Date` class giving the last date of the event period.

Details

This test is application of the simple binomial test to the event study, which indicates whether the cross-sectional frequency of positive abnormal returns is significantly different from 0.5. This test is stable to outliers, in other words allows for checking if the result is driven by few companies with extremely large abnormal performance. For this test the estimation period and the event period must not overlap, otherwise an error will be thrown. The test statistic is assumed to have a normal distribution in approximation under a null hypothesis, if the number of securities is large. Typically the test is used together with parametric tests. The test is well-specified for the case, when cross-sectional abnormal returns are not symmetric. Also this procedure is less sensitive to extreme returns than the rank test. The significance levels of α are 0.1, 0.05, and 0.01 (marked respectively by *, **, and ***).

Value

A data frame of the following columns:

- `date`: a calendar date
- `weekday`: a day of the week
- `percentage`: a share of non-missing observations for a given day
- `sign_stat`: a sign test statistic
- `sign_signif`: a significance of the statistic

References

Boehmer E., Musumeci J., Poulsen A.B. *Event-study methodology under conditions of event-induced variance*. Journal of Financial Economics, 30(2):253-272, 1991.

See Also

nonparametric_tests, generalized_sign_test, corrado_sign_test, rank_test, modified_rank_test, and wilcoxon_test.

Examples

```
## Not run:
library("magrittr")
rates_indx <- get_prices_from_tickers("^GSPC",
                                      start = as.Date("2019-04-01"),
                                      end = as.Date("2020-04-01"),
                                      quote = "Close",
                                      retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous")
tickers <- c("AMZN", "ZM", "UBER", "NFLX", "SHOP", "FB", "UPWK")
get_prices_from_tickers(tickers,
                        start = as.Date("2019-04-01"),
                        end = as.Date("2020-04-01"),
                        quote = "Close",
                        retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous") %>%
    apply_market_model(regressor = rates_indx,
                       same_regressor_for_all = TRUE,
                       market_model = "sim",
                       estimation_method = "ols",
                       estimation_start = as.Date("2019-04-01"),
                       estimation_end = as.Date("2020-03-13")) %>%
    sign_test(event_start = as.Date("2020-03-16"),
              event_end = as.Date("2020-03-20"))

## End(Not run)
## The result of the code above is equivalent to:
data(securities_returns)
sign_test(list_of_returns = securities_returns,
          event_start =  as.Date("2020-03-16"),
          event_end = as.Date("2020-03-20"))
```

t_test *An event study t-test.*

Description

A classical t-test that examines each date in the event window.

Usage

```
t_test(list_of_returns, event_start, event_end)
```

Arguments

`list_of_returns`	a list of objects of S3 class `returns`, each element of which is treated as a security.
`event_start`	an object of `Date` class giving the first date of the event period.
`event_end`	an object of `Date` class giving the last date of the event period.

Details

Performs a t-test for the event study. The procedure of this test is described in Boehmer et al. 1991, sometimes is called a cross-sectional test. Assumes independence of securities, however is stable to event-induced variance. This test examines the equality of the cross-sectional expected value to zero. The standard deviation, which is used in this test, is simply a cross-sectional standard deviation for a given day in the event window. It calculates statistics even if event window and estimation period are overlapped (intersect). The critical values are Student's t-distributed (no approximation in limit). The significance levels of α are 0.1, 0.05, and 0.01 (marked respectively by *, **, and ***).

Value

A data frame of the following columns:

- `date`: a calendar date
- `weekday`: a day of the week
- `percentage`: a share of non-missing observations for a given day
- `mean`: an average abnormal return
- `t_test_stat`: a t-test statistic
- `t_test_signif`: a significance of the statistic

Warning

This test strongly requires cross-sectional independence and sensitive to the size of the sample.

References

Boehmer E., Musumeci J., Poulsen A.B. *Event-study methodology under conditions of event-induced variance*. Journal of Financial Economics, 30(2):253-272, 1991.

See Also

`parametric_tests`, `brown_warner_1980`, `brown_warner_1985`, `patell`, `boehmer`, and `lamb`.

Examples

```
## Not run:
library("magrittr")
rates_indx <- get_prices_from_tickers("^GSPC",
                                      start = as.Date("2019-04-01"),
                                      end = as.Date("2020-04-01"),
                                      quote = "Close",
                                      retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous")
tickers <- c("AMZN", "ZM", "UBER", "NFLX", "SHOP", "FB", "UPWK")
get_prices_from_tickers(tickers,
                        start = as.Date("2019-04-01"),
                        end = as.Date("2020-04-01"),
                        quote = "Close",
                        retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous") %>%
    apply_market_model(regressor = rates_indx,
                       same_regressor_for_all = TRUE,
                       market_model = "sim",
                       estimation_method = "ols",
                       estimation_start = as.Date("2019-04-01"),
                       estimation_end = as.Date("2020-03-13")) %>%
    t_test(event_start = as.Date("2020-03-16"),
           event_end = as.Date("2020-03-20"))

## End(Not run)
## The result of the code above is equivalent to:
data(securities_returns)
t_test(list_of_returns = securities_returns,
       event_start = as.Date("2020-03-16"),
       event_end = as.Date("2020-03-20"))
```

`wilcoxon_test` *An event study Wilcoxon signed rank test.*

Description

Performs Wilcoxon test on the event period for abnormal returns (abnormal returns are considered as differences).

Usage

```
wilcoxon_test(list_of_returns, event_start, event_end)
```

Arguments

`list_of_returns`	a list of objects of S3 class `returns`, each element of which is treated as a security.
`event_start`	an object of `Date` class giving the first date of the event period.
`event_end`	an object of `Date` class giving the last date of the event period.

Details

The estimation periods can overlap with event windows, because the procedure takes into account only abnormal returns from the event window. The test has the same algorithm as built-in `R` `wilcox.test`. The critical values are exact values, which are obtained from `qsignrank`. The algorithm is the following: for each day in event window the cross-sectional abnormal returns treated as sample of differences. Firstly the absolute value of these differences are computed, and corresponding ranks of non-zero values are calculated. The test statistic is the sum of ranks, corresponding to positive abnormal returns. The significance levels of α are 0.1, 0.05, and 0.01 (marked respectively by *, **, and ***).

Value

A data frame of the following columns:

- `date`: a calendar date
- `weekday`: a day of the week
- `percentage`: a share of non-missing observations for a given day
- `wlcx_stat`: a Wilcoxon signed rank test statistic
- `wlcx_signif`: a significance of the statistic

References

- Wilcoxon F. *Individual Comparisons by Ranking Methods*. Biometrics Bulletin 1(6):80-83, 1945.
- Kolari J.W., Pynnonen S. *Event Study Testing with Cross-sectional Correlation of Abnormal Returns*. The Review of Financial Studies, 23(11):3996-4025, 2010.
- Lehmann E.L, *Nonparametrics: Statistical Methods Based on Ranks*. San Francisco: Holden-Day, 1975.
- Hollander M., Wolfe D.A. *Nonparametric Statistical Methods*. New York: John Wiley & Sons, 1973.

See Also

`nonparametric_tests`, `sign_test`, `generalized_sign_test`, `corrado_sign_test`, `rank_test`, and `modified_rank_test`.

Examples

```
## Not run:
library("magrittr")
rates_indx <- get_prices_from_tickers("^GSPC",
                                      start = as.Date("2019-04-01"),
                                      end = as.Date("2020-04-01"),
                                      quote = "Close",
                                      retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous")
tickers <- c("AMZN", "ZM", "UBER", "NFLX", "SHOP", "FB", "UPWK")
get_prices_from_tickers(tickers,
                        start = as.Date("2019-04-01"),
                        end = as.Date("2020-04-01"),
                        quote = "Close",
                        retclass = "zoo") %>%
    get_rates_from_prices(quote = "Close",
                          multi_day = TRUE,
                          compounding = "continuous") %>%
    apply_market_model(regressor = rates_indx,
                       same_regressor_for_all = TRUE,
                       market_model = "sim",
                       estimation_method = "ols",
                       estimation_start = as.Date("2019-04-01"),
                       estimation_end = as.Date("2020-03-13")) %>%
    wilcoxon_test(event_start = as.Date("2020-03-16"),
                  event_end = as.Date("2020-03-20"))

## End(Not run)
## The result of the code above is equivalent to:
data(securities_returns)
wilcoxon_test(list_of_returns = securities_returns,
              event_start =  as.Date("2020-03-16"),
              event_end = as.Date("2020-03-20"))
```

Index

∗ **datasets**
- prices, 38
- prices_indx, 39
- rates, 41
- rates_indx, 42
- securities_returns, 46

apply_market_model, 3, *44*, *46*

boehmer, 6, *9*, *11*, *29*, *35*, *37*, *49*
brown_warner_1980, *7*, 8, *11*, *29*, *35*, *37*, *49*
brown_warner_1985, *7*, *9*, 10, *29*, *35*, *37*, *49*

car_brown_warner_1985, 11, *14*, *19*
car_lamb, *12*, 13, *19*
car_nonparametric_tests, 15, *21*
car_parametric_tests, *12*, *14*, 17
car_rank_test, *16*, 20
corrado_sign_test, 22, *24*, *31*, *33*, *34*, *40*, *48*, *51*

generalized_sign_test, *23*, 23, *31*, *33*, *34*, *40*, *48*, *51*
get_prices_from_tickers, 25, *38*, *39*
get_rates_from_prices, 27, *41*, *42*
getSymbols, *26*

lamb, *7*, *9*, *11*, 28, *35*, *37*, *49*

modified_rank_test, *23*, *24*, 30, *33*, *34*, *40*, *48*, *51*

nonparametric_tests, *23*, *24*, *31*, 32, *40*, *48*, *51*

parametric_tests, *7*, *9*, *11*, *29*, 34, *37*, *49*
patell, *7*, *9*, *11*, *29*, *35*, 36, *49*
prices, 38
prices_indx, 39

qsignrank, *51*

rank, *20*, *40*
rank_test, *23*, *24*, *31*, *33*, *34*, 39, *48*, *51*
rates, 41
rates_indx, 42
returns, *4*, 42
run_app, 46

securities_returns, 46
sign_test, *23*, *24*, *31*, *33*, *34*, *40*, 47, *51*

t_test, *7*, *9*, *11*, *29*, *35*, *37*, 48

wilcox.test, *51*
wilcoxon_test, *23*, *24*, *31*, *33*, *34*, *40*, *48*, 50