Package ‘GeoexperimentsResearch’

March 23, 2017

Title Geo Experiments

Author Google Inc. `<geoexperiments-opensource@google.com>`

Maintainer Google Inc. `<geoexperiments-opensource@google.com>`

Description Functions for manipulating and analyzing geo experiment data.

Depends R (>= 3.2.2), stats

Imports assertthat (>= 0.1.0.99), ggplot2 (>= 2.0.0), reshape2 (>= 1.2.1), MASS (>= 7.3)

Suggests testthat (>= 0.10.0), knitr (>= 1.12.3)

VignetteBuilder knitr

Date 2017-03-23

Version 1.0.1

License Apache License 2.0 | file LICENSE

Copyright Copyright (C) 2017 Google, Inc.

RoxygenNote 5.0.1

R topics documented:

`aggregate.GeoTimeseries` *Aggregate the Metrics of a GeoTimeseries.*

Description

Aggregate the Metrics of a GeoTimeseries.

Usage

```
## S3 method for class 'GeoTimeseries'
aggregate(x, by = kGeo, FUN = base::sum,
  metrics = NULL, ...)
```

Arguments

`x`	a GeoTimeseries object.
`by`	(character vector) name(s) of column(s) by which to group.
`FUN`	(function) function to apply to each metric column.
`metrics`	(character vector) metrics to aggregate. Default is all metrics.
`...`	optional arguments passed to FUN.

Value

A data.frame object.

Note

Uses ’aggregate.data.frame’ to do the aggregation. This function omits rows that have missing values in the ’by’ columns.

See Also

AggregateTimeseries.

`AggregateTimeseries`

Aggregate the metrics of a time series object over specified time intervals.

Description

Aggregate the metrics of a time series object over specified time intervals.

Usage

```
AggregateTimeseries(obj, freq = c("weekly", "monthly"), ...)
```

Arguments

`obj`	an object.
`freq`	(string) 'weekly' or 'monthly' aggregation.
`...`	further arguments passed to or from other methods.

Value

An object of the same class as 'obj'.

Note

A generic S3 method.

See Also

AggregateTimeseries.GeoTimeseries.

`AggregateTimeseries.GeoTimeseries`

Convert a GeoTimeseries to a weekly or monthly GeoTimeseries. (Note: intended to transform daily data.)

Description

Convert a GeoTimeseries to a weekly or monthly GeoTimeseries. (Note: intended to transform daily data.)

Usage

```
## S3 method for class 'GeoTimeseries'
AggregateTimeseries(obj, freq, ...)
```

Arguments

`obj`	a GeoTimeseries object.
`freq`	desired frequency: 'weekly' or 'monthly'.
`...`	ignored.

Value

A GeoTimeseries object with the weekly or monthly metrics aggregated. The sums are associated with the last day of the week (which is by our definition _Sunday_), or the last day of the month.

Note

'Weekly' frequency: each day in the time series is mapped to the next Sunday, then aggregated. 'Monthly' frequency: each day in the time series is mapped to the last day of the month, then aggregated. No check is made about the current frequency. This works best on daily data. So it is possible to attempt to change a 'monthly' frequency back to 'weekly', but this simply re-maps the last day of the month to the next Sunday.

See Also

AggregateTimeseries (generic), aggregate.GeoTimeseries.

`as.GBRROASAnalysisData`

Coerces the object to a GBRROASAnalysisData object.

Description

Coerces the object to a GBRROASAnalysisData object.

Usage

```
as.GBRROASAnalysisData(obj, ...)
```

Arguments

`obj`	an object.
`...`	further arguments to be passed to or from other methods.

Value

A GBRROASAnalysisData object.

Note

A generic S3 method.

See Also

as.GBRROASAnalysisData.GeoExperimentData.

---

as.GBRROASAnalysisData.GeoExperimentData

Coerces an object to a GBRROASAnalysisData object.

Description

Coerces an object to a GBRROASAnalysisData object.

Usage

```
## S3 method for class 'GeoExperimentData'
as.GBRROASAnalysisData(obj,
  response = character(0), cost = character(0), pretest.period = 0L,
  intervention.period = 1L, cooldown.period = NULL, control.group = 1L,
  treatment.group = 2L, ...)
```

Arguments

`obj`	a GeoExperimentData object.
`response`	(string) name of the response variable column.
`cost`	(string) name of the cost variable column.
`pretest.period`	(vector of non-negative integers) number(s) of the period(s) forming the pretest period.
`intervention.period`	(vector of non-negative integers) number(s) of the period(s) forming the intervention period. All must be larger than the largest period in the pretest period.
`cooldown.period`	(NULL or vector of non-negative integers) number(s) of the period(s) forming the cooldown period. All must be larger than the largest period in the intervention period.
`control.group`	(NULL or a vector of positive integers) number(s) of geo groups forming the control group.
`treatment.group`	(NULL or a vector of positive integers) number(s) of geo groups forming the control group.
`...`	ignored.

Value

A GBRROASAnalysisData object.

See Also

as.GBRROASAnalysisData (generic).

`as.GeoExperimentData` *Coerces an object to a GeoExperimentData object.*

Description

Coerces an object to a GeoExperimentData object.

Usage

```
as.GeoExperimentData(obj, ...)
```

Arguments

`obj`	an object.
`...`	further arguments passed to methods.

Value

A GeoExperimentData object.

Note

A generic S3 method.

`as.GeoExperimentData.GeoTimeseries` *Coerces a GeoTimeseries object to a GeoExperiment object.*

Description

Coerces a GeoTimeseries object to a GeoExperiment object.

Usage

```
## S3 method for class 'GeoTimeseries'
as.GeoExperimentData(obj, strict = TRUE, ...)
```

Arguments

`obj`	a GeoTimeseries object with the additional columns 'period', 'geo.group', and 'assignment'. If any of these columns are missing, the corresponding columns in the resulting object will be 'NA'.
`strict`	(flag) if FALSE, the additional columns are optional and no error is thrown if any of them is missing;
`...`	ignored.

Value

A GeoExperimentData object.

See Also

as.GeoExperimentData (generic).

`as.GeoTimeseries` *Coerce an object to a GeoTimeseries object.*

Description

Coerce an object to a GeoTimeseries object.

Usage

```
as.GeoTimeseries(obj, ...)
```

Arguments

`obj`	an object.
`...`	additional arguments to be passed to or from methods.

Value

A GeoTimeseries object.

Note

A S3 generic method.

`as.matrix.GeoTimeseries` *Coerces a response metric of a GeoTimeseries to a matrix representation with geos in rows, dates in columns.*

Description

Coerces a response metric of a GeoTimeseries to a matrix representation with geos in rows, dates in columns.

Usage

```
## S3 method for class 'GeoTimeseries'
as.matrix(x, response, ...)
```

Arguments

`x`	a GeoTimeseries object.
`response`	(string) name of the response metric to use.
`...`	ignored.

Value

A numeric matrix of the response metric, geos in rows, dates in columns.

`as.matrix.TBRQuantiles` *Extracts the real-valued matrix of quantiles from a TBRQuantiles object.*

Description

Extracts the real-valued matrix of quantiles from a TBRQuantiles object.

Usage

```
## S3 method for class 'TBRQuantiles'
as.matrix(x, ...)
```

Arguments

`x`	a TBRQuantiles object.
`...`	ignored.

Value

A real-valued matrix of the quantiles. The column names are the same as those in the TBRQuantiles object.

`as.TBRAnalysisData` *Coerces the object to a TBRAnalysisData object.*

Description

Coerces the object to a TBRAnalysisData object.

Usage

```
as.TBRAnalysisData(obj, ...)
```

Arguments

`obj`	an object.
`...`	further arguments to be passed to or from other methods.

Value

A TBRAnalysisData object.

Note

A generic S3 method.

See Also

as.TBRAnalysisData.GeoExperimentData.

`as.TBRAnalysisData.GeoExperimentData` *Coerces a GeoExperimentData object to a TBRAnalysisData object.*

Description

Coerces a GeoExperimentData object to a TBRAnalysisData object.

Usage

```
## S3 method for class 'GeoExperimentData'
as.TBRAnalysisData(obj, response = character(0),
  control.group = 1L, treatment.group = 2L, pretest.period = 0L,
  intervention.period = 1L, cooldown.period = NULL, ...)
```

Arguments

`obj`	a GeoExperimentData object.
`response`	(string) name of the response metric to analyze.
`control.group`	(positive integer) number of the control group (matching one of the groups in the column 'geo.group'). This is typically 1.
`treatment.group`	(positive integer) number of the treatment group (matching one of the groups in the column 'geo.group'). This is typically 2.
`pretest.period`	(non-negative integers) number of the pretest period, typically 0. Can also be one or more numbers, if periods are to be collapsed.
`intervention.period`	(vector of non-negative integers) number(s) of the period(s) forming the intervention period. All must be larger than the largest period in the pretest period.
`cooldown.period`	(NULL or vector of non-negative integers) number(s) of the period(s) forming the cooldown period. All must be larger than the largest period in the intervention period.
`...`	ignored.

Value

A TBRAnalysisData object.

See Also

as.TBRAnalysisData (generic).

`as.TBRPlotData` *Coerces an object to a TBRPlotData object.*

Description

Coerces an object to a TBRPlotData object.

Usage

```
as.TBRPlotData(obj, panels, periods, quantiles = c(0.1, 0.9), ...)
```

Arguments

`obj`	an object to coerce.
`panels`	(vector of strings or NULL) names of the panels to be plotted. By default, all available panels.
`periods`	(vector of strings or NULL) names of the periods to show. By default, all default periods.
`quantiles`	(real vector of length 2) lower and upper quantiles of the credible interval to show.
`...`	further arguments passed to the methods.

Value

A TBRPlotData object.

`as.TBRPlotData.TBRAnalysisFitTbr1`
Coerces a TBRAnalysisFitTbr1 object to a TBRPlotData object.

Description

Coerces a TBRAnalysisFitTbr1 object to a TBRPlotData object.

Usage

```
## S3 method for class 'TBRAnalysisFitTbr1'
as.TBRPlotData(obj,
  panels = GetTBRPlotPanelNames(), periods = c("pretest", "prediction"),
  quantiles = c(0.1, 0.9), ...)
```

Arguments

`obj`	an object to coerce.
`panels`	(vector of strings) names of the panels to be plotted.
`periods`	(vector of strings) names of the periods to show.
`quantiles`	(real vector of length 2) lower and upper quantiles of the credible interval to show.
`...`	ignored.

Value

A TBRPlotData object.

`as.TBRPlotData.TBRROASAnalysisFit` *Coerces a TBRROASAnalysisFit object to a TBRPlotData object.*

Description

Coerces a TBRROASAnalysisFit object to a TBRPlotData object.

Usage

```
## S3 method for class 'TBRROASAnalysisFit'
as.TBRPlotData(obj, panels = "cumulative",
  periods = "prediction", quantiles = c(0.1, 0.9), ...)
```

Arguments

`obj`	an object to coerce.
`panels`	(vector of strings) names of the panels to be plotted.
`periods`	(vector of strings) names of the periods to show.
`quantiles`	(real vector of length 2) lower and upper quantiles of the credible interval to show.
`...`	ignored.

Value

A TBRPlotData object.

`CheckForAllTrue` *Check whether all(x) is true.*

Description

Check whether all(x) is true.

Usage

```
CheckForAllTrue(x, ...)
```

Arguments

`x`	(logical) vector. A 'TRUE' denotes a success, and 'FALSE' and 'NA' a failure. The names attribute must be set, with nonempty, non-NA, non-duplicated names.
`...`	one or more character vectors, passed on to `FormatText`.

Value

If all tests pass, returns TRUE invisibly. Otherwise, an ’assertError’ is thrown.

`CheckForBadValues` *Check for bad values in given columns.*

Description

Check for bad values in given columns.

Usage

```
CheckForBadValues(x, columns, CHECK, good = TRUE, what = "invalid", ...)
```

Arguments

`x`	(data frame) any data frame.
`columns`	(character) columns to check.
`CHECK`	(function) function that returns a logical vector (of length `nrow(x)`).
`good`	(logical) the value or values that indicate a ’good’ value. Other values returned by ’CHECK’ will be ’bad’ and throw an error.
`what`	(string) a string to display in the informative message.
`...`	further arguments passed on to function ’CHECK’.

Value

If all tests pass, returns NULL invisibly. Otherwise, an ’assertError’ is thrown.

`CheckForDuplicateRows` *Tests whether the values in rows of given columns are not duplicated.*

Description

Tests whether the values in rows of given columns are not duplicated.

Usage

```
CheckForDuplicateRows(x, columns)
```

Arguments

`x`	a data frame.
`columns`	(character) vector.

Value

If all tests pass, returns TRUE invisibly. Otherwise, an 'assertError' is thrown.

`CheckForMapping` *Tests whether the mapping is consistent within a data frame.*

Description

Tests whether the mapping is consistent within a data frame.

Usage

```
CheckForMapping(x, from, to)
```

Arguments

`x`	a data frame.
`from`	(character) vector of column names.
`to`	(character) vector of column names. Typically these are different from those in 'from'.

Value

If all tests pass, returns TRUE invisibly. Otherwise, an 'assertError' is thrown.

Note

Checks that no rows in 'x[from]' map to two different values of 'x[to]'.

`CheckForMissingColumns` *Tests whether specified column names exist in a data frame.*

Description

Tests whether specified column names exist in a data frame.

Usage

```
CheckForMissingColumns(x, dataframe, what = "specified")
```

Arguments

`x`	(character) names of the columns.
`dataframe`	(data frame) data frame.
`what`	(string) string to use in the error message.

Value

If all tests pass, returns TRUE invisibly. Otherwise, an 'assertError' is thrown.

Note

Convenient for checking function arguments. Quotes the name of the variable 'x'.

`CheckForTypes` *Tests whether the vector values in the given list satisfies the specified types.*

Description

Tests whether the vector values in the given list satisfies the specified types.

Usage

```
CheckForTypes(x, checklist)
```

Arguments

`x`	a named list or a data frame.
`checklist`	(named list) mapping from names to functions that check whether the functions are correct or not.

Value

invisible NULL.

`CheckGeoGroupNumber` *Check that the specified geo group number is present in the data.*

Description

Check that the specified geo group number is present in the data.

Usage

```
CheckGeoGroupNumber(geo.group, values)
```

Arguments

`geo.group`	(an integer) geo group number (>= 1). (NA throws an error).
`values`	(vector) values to check against.

Value

TRUE, invisibly; as a side effect may throw an assertion error.

Note

For checking arguments within a function. Outputs the name of the variable in error messages.

`CheckPeriodNumbers` *Check that the specified period numbers are present in the data.*

Description

Check that the specified period numbers are present in the data.

Usage

```
CheckPeriodNumbers(period, values)
```

Arguments

`period`	(integer vector): one or more period numbers. (NA throws an error).
`values`	(vector) values to check against.

Value

TRUE invisibly if the checks pass. As a side effect will terminate with an error if the checks fail.

Note

For checking arguments within a function. Outputs the name of the variable in error messages.

`CheckThat` *An alternative version of assert_that, outputting a modified error message.*

Description

An alternative version of assert_that, outputting a modified error message.

Usage

```
CheckThat(..., env = parent.frame(), name = NULL)
```

Arguments

`...`	see 'assert_that'.
`env`	see 'assert_that'.
`name`	(string) name of the object to output in the error message.

Value

'TRUE' if tests pass; otherwise throws an error.

Note

Replaces anything up to ' is not ' in the error message with the value of 'name'.

Examples

```
# Outputs "Column 'foo' is not an integer-valued numeric vector"
# if 'x' fails the test.
## Not run:
AssertThat(is.integer.valued(x), name="Column 'foo'")
## End(Not run)
```

---

coef.GBRROASAnalysisFitGbr1 *Returns the model coefficients.*

Description

Returns the model coefficients.

Usage

```
## S3 method for class 'GBRROASAnalysisFitGbr1'
coef(object, ...)
```

Arguments

`object`	a GBRROASAnalysisFitGbr1 object.
`...`	ignored.

Value

A named numeric vector with the model coefficients from the GBR analysis.

`ComputeLinearModelWeights` *Computes the weights to be used in the weighted linear model used to estimate ROAS.*

Description

Computes the weights to be used in the weighted linear model used to estimate ROAS.

Usage

```
ComputeLinearModelWeights(response, power = 2)
```

Arguments

`response`	a vector of the response in the pre period. Length equal to the number of geos. Must be all nonnegative.
`power`	default power to which 'response' is raised to. Can be overridden by setting the global option 'geoexperiments.gbr1.weight.power'. Must be nonnegative.

Value

A vector of weights of the same length as 'response'. Data points with response == 0 have weight NA (indicating these need to be taken special care of). There is an attribute 'power' corresponding to the exponent used.

Note

If a component of 'response' tends to infinity, the corresponding weight tends to 0 (i.e., the corresponding data point is ignored).

`ConcatItems` *Concatenate items of a vector, optionally quoting each.*

Description

Concatenate items of a vector, optionally quoting each.

Usage

```
ConcatItems(x, quote = "'", collapse = ", ", max.output = Inf)
```

Arguments

`x`	(atomic vector) an atomic vector, coerced to character.
`quote`	(string) a quote character to use.
`collapse`	(string) string to use for collapsing the components of 'x'.
`max.output`	(integer) maximum number of items of 'x' to output.

Value

A string.

`confint.GBRROASAnalysisFitGbr1`

Returns the confidence intervals associated with the model parameters.

Description

Returns the confidence intervals associated with the model parameters.

Usage

```
## S3 method for class 'GBRROASAnalysisFitGbr1'
confint(object, parm = "incr.cost",
  level = 0.95, ...)
```

Arguments

`object`	a GBRROASAnalysisFitGbr1 object.
`parm`	(character vector) a specification of which parameters are to be given confidence intervals, either a vector of numbers or a vector of names.
`level`	(number between 0 and 1) the confidence level required.
`...`	ignored.

Value

A matrix with the confidence intervals from the 'confint.lm' method.

`CountRandomizations` *Counts the total numbers of randomizations in a GeoStrata object.*

Description

Counts the total numbers of randomizations in a GeoStrata object.

Usage

```
CountRandomizations(geostrata, show.warnings = TRUE, log.scale = FALSE)
```

Arguments

`geostrata`	a GeoStrata object.
`show.warnings`	(flag) if TRUE, shows a warning when a stratum is not compatible with the group.ratios.
`log.scale`	(flag) if TRUE, returns the result on the log.scale.

Value

An integer vector of the same length as the number of strata in geostrata (excluding stratum 0, if it exists). The i-th coordinate corresponds to the total number of possible randomizations for the i-th stratum.

Note

A warning is issued if one or more of the stratas are not compatible with the group.ratios.

`CountRandomizationsInAStratum` *Counts the total numbers of randomization in a single stratum.*

Description

Counts the total numbers of randomization in a single stratum.

Usage

```
CountRandomizationsInAStratum(geo.group, group.ratios, log.scale = TRUE)
```

Arguments

`geo.group`	(integer vector of length equal to number of geos) a vector of geo group numbers (may be NAs, but zeros are not allowed).
`group.ratios`	(integer vector of length equal to number of geo groups) vector of ratios of the sizes of each group.
`log.scale`	(flag) if TRUE, returns the result on the log.scale.

Value

An integer value that corresponds to the total number of possible randomizations for the stratum represented by geo.group.

`cumsum.PosteriorSimulations`

Returns simulations of the posterior cumulative joint distribution.

Description

Returns simulations of the posterior cumulative joint distribution.

Usage

```
## S3 method for class 'PosteriorSimulations'
cumsum(x)
```

Arguments

`x`	a PosteriorSimulations object.

Value

A PosteriorSimulations object.

`DefaultTreatmentAssignment`

Generates the default treatment assignment condition.

Description

Generates the default treatment assignment condition.

Usage

```
DefaultTreatmentAssignment()
```

Value

A TreatmentAssignment object.

Note

Period 1 of group 2 is assigned a change (some intervention).

`DoGBRROASAnalysis` *Performs a GBR ROAS Analysis.*

Description

Performs a GBR ROAS Analysis.

Usage

```
DoGBRROASAnalysis(obj, ...)
```

Arguments

`obj`	an object.
`...`	further arguments passed to or from other methods.

Value

A GBRROASAnalysisFit object.

Note

A generic S3 method. See also: DoGBRROASAnalysis.GBRROASAnalysisData, DoGBRROAS-Analysis.GeoExperimentData.

`DoGBRROASAnalysis.GBRROASAnalysisData` *Performs a ROAS Analysis on a GBRROASAnalysisData object.*

Description

Performs a ROAS Analysis on a GBRROASAnalysisData object.

Usage

```
## S3 method for class 'GBRROASAnalysisData'
DoGBRROASAnalysis(obj, ...)
```

Arguments

`obj`	a GBRROASAnalysisData object.
`...`	ignored.

Value

A GBRROASAnalysisFit object.

See Also

DoGBRROASAnalysis (generic), DoGBRROASAnalysis.GeoExperimentData.

`DoGBRROASAnalysis.GeoExperimentData` *Performs a ROAS Analysis on a GeoExperimentData object.*

Description

Performs a ROAS Analysis on a GeoExperimentData object.

Usage

```
## S3 method for class 'GeoExperimentData'
DoGBRROASAnalysis(obj, response = character(0),
  cost = character(0), pretest.period = 0L, intervention.period = 1L,
  cooldown.period = NULL, control.group = 1L, treatment.group = 2L, ...)
```

Arguments

`obj`	a GeoExperimentData object.
`response`	(string) name of the response variable column.
`cost`	(string) name of the cost variable column.
`pretest.period`	(non-negative integer) number of the pretest period, typically 0.
`intervention.period`	(vector of non-negative integers) number(s) of the period(s) forming the intervention period. All must be larger than the largest period in the pretest period.
`cooldown.period`	(vector of non-negative integers or NULL) number(s) of the period(s) forming the cooldown period. All must be larger than the largest period in the intervention period.
`control.group`	(a vector of positive integers) number(s) of geo groups forming the control group.

`treatment.group`	(a vector of positive integers) number(s) of geo groups forming the treatment group.
`...`	further arguments passed to as.GBRROASAnalysisData.

Value

A GBRROASAnalysisFit object.

Note

See also: DoGBRROASAnalysis (generic), DoGBRROASAnalysis.GBRROASAnalysisData.

`DoROASAnalysis` *Performs a ROAS analysis using the given model.*

Description

Performs a ROAS analysis using the given model.

Usage

```
DoROASAnalysis(obj, model, ...)
```

Arguments

`obj`	an object.
`model`	(string) id of the model to use. Available models are 'gbr1', 'tbr1'.
`...`	further arguments passed to or from other models.

Value

A GBRROASAnalysisFit (for 'gbr1') or a TBRROASAnalysisFit object (for 'tbr1').

Note

Dispatches the right model for the given model type.

`DoROASPreanalysis` *Predicts the standard deviation (standard error) of the posterior of the iROAS estimator.*

Description

Predicts the standard deviation (standard error) of the posterior of the iROAS estimator.

Usage

```
DoROASPreanalysis(obj, ...)
```

Arguments

`obj`	some object.
`...`	arguments passed on to the models.

Value

A ROASPreanalysisFit object.

`DoROASPreanalysis.GeoExperimentPreanalysisData` *Predicts the standard deviation of the posterior of the iROAS estimator by simulating experiments using historical data.*

Description

Predicts the standard deviation of the posterior of the iROAS estimator by simulating experiments using historical data.

Usage

```
## S3 method for class 'GeoExperimentPreanalysisData'
DoROASPreanalysis(obj, response, prop.to,
  n.sims = 1000, models = GetModelIds(), FUN = base::lapply, ...)
```

Arguments

`obj`	a GeoExperimentPreanalysisData object.
`response`	(string) name of the metric column to use as the response variable.
`prop.to`	(string) an existing name of the column in proportion to which the spend change will be distributed across the geos.

`n.sims`	(integer or NA) number of simulations to do. Note: if 'geos' is a GeoAssignment object, NA is allowed, and the actual number of simulations will be the number of all possible pseudo-data sets; if n.sims is given, only the first n.sims data sets will be simulated (useful for testing only).
`models`	(vector of nonempty strings) one or more analysis model ids, to apply to each simulated data set.
`FUN`	(function) a lapply-compatible function to use for generating the simulations. Use this for a parallel version of lapply to speed up the process. Note: the function only needs to be able to iterate along a vector of integers.
`...`	arguments passed to FUN.

Value

A ROASPreanalysisFit object.

Note

Simulates experiments without resorting to regenerating time series. Segments of the original time series are used to create 'pseudo-timeseries' of the length of the experiment. These simulated geo experiment data sets are produced by calls to 'SimulateGeoExperimentData'.

The object that is returned contains the simulated raw numbers. Use the 'summary' method for a more user-friendly output.

`DoROASPreanalysis.GeoTimeseries`
Predicts the standard deviation of the posterior of the iROAS estimator by simulating experiments using historical data.

Description

Predicts the standard deviation of the posterior of the iROAS estimator by simulating experiments using historical data.

Usage

```
## S3 method for class 'GeoTimeseries'
DoROASPreanalysis(obj, period.lengths, geos,
  recycle = TRUE, ...)
```

Arguments

`obj`	a GeoTimeseries object.
`period.lengths`	a vector of length 3, denoting the lengths (in days) of pre-period, test, and the cooldown periods, respectively. The test period must be at least 7 days; the pre-period must be at least as long as the test period; the cooldown period can be 0 or more days.

`geos`	(GeoStrata or GeoAssignment object) object to use for choosing the geo groups; if it is a GeoAssignment object, the geo assignment will be fixed. If it is a GeoStrata object, method 'Randomize' will be used to obtain a geo assignment.
`recycle`	(flag) if TRUE, uses an augmented data set in order to create a larger time series, reusing data from the head of the time series.
`...`	arguments passed to DoROASPreanalysis.GeoExperimentPreanalysisData.

Value

A ROASPreanalysisFit object.

Note

This is a simple wrapper, combining calls to 'GeoExperimentPreanalysisData' and 'DoROASPreanalysis'.

`DoTBRAnalysis` *Performs a TBR Causal Effect Analysis.*

Description

Performs a TBR Causal Effect Analysis.

Usage

```
DoTBRAnalysis(obj, model, ...)
```

Arguments

`obj`	an object.
`model`	(string) model to use.
`...`	further arguments passed to or from other methods.

Value

A TBRAnalysisFit object.

Note

A generic S3 method.

See Also

DoTBRAnalysis.TBRAnalysisData, DoTBRAnalysis.GeoExperimentData.

`DoTBRAnalysis.GeoExperimentData` *Performs a TBR Causal Effect Analysis on a GeoExperimentData object.*

Description

Performs a TBR Causal Effect Analysis on a GeoExperimentData object.

Usage

```
## S3 method for class 'GeoExperimentData'
DoTBRAnalysis(obj, model, ...)
```

Arguments

`obj`	a GeoExperimentData object.
`model`	(string) model to use. See the generic method for possible values.
`...`	arguments passed on to the method performing the analysis.

Value

A TBRAnalysisFit object.

Note

Calls 'as.TBRAnalysisData' and passes the 'TBRAnalysisData' object to the analysis method.

See Also

DoTBRAnalysis (generic), DoTBRAnalysis.TBRAnalysisData.

`DoTBRAnalysis.TBRAnalysisData` *Performs a TBR Analysis on a TBRAnalysisData object.*

Description

Performs a TBR Analysis on a TBRAnalysisData object.

Usage

```
## S3 method for class 'TBRAnalysisData'
DoTBRAnalysis(obj, model, ...)
```

Arguments

`obj`	a TBRAnalysisData object.
`model`	(string) model to use. See the generic method for possible values.
`...`	Arguments passed on to the method performing the analysis.

Value

A TBRAnalysisFit object.

See Also

DoTBRAnalysis (generic), DoTBRAnalysis.GeoExperimentData.

`DoTBRAnalysisTbr1` *Performs a TBR Analysis using the 2-parameter linear model 'tbr1'.*

Description

Performs a TBR Analysis using the 2-parameter linear model 'tbr1'.

Usage

```
DoTBRAnalysisTbr1(obj, ...)
```

Arguments

`obj`	an object.
`...`	further arguments passed to or from other methods.

Value

A TBRAnalysisFit object.

Note

A generic S3 method. Users should use 'DoTBRAnalysis' instead, specifying the 'model' parameter.

See Also

DoTBRAnalysis.TBRAnalysisData, DoTBRAnalysisTbr1.TBRAnalysisData

DoTBRAnalysisTbr1.TBRAnalysisData
Performs a TBR Analysis using the 2-parameter linear model 'tbr1'.

Description

Performs a TBR Analysis using the 2-parameter linear model 'tbr1'.

Usage

```
## S3 method for class 'TBRAnalysisData'
DoTBRAnalysisTbr1(obj, ...)
```

Arguments

`obj`	a TBRAnalysisData object.
`...`	ignored.

Value

A TBRAnalysisFit object.

Note

Users should use 'DoTBRAnalysis' instead.

See Also

DoTBRAnalysis.TBRAnalysisData, DoTBRAnalysisTbr1.TBRAnalysisData

DoTBRROASAnalysis *Performs a TBR ROAS Analysis.*

Description

Performs a TBR ROAS Analysis.

Usage

```
DoTBRROASAnalysis(obj, model, response, cost, n.sims = 1e+05, ...)
```

Arguments

`obj`	an object.
`model`	(string) id of the TBR to use.
`response`	(string) name of the column of the response variable.
`cost`	(string) name of the column of the cost variable, or alternatively a real number specifying the exact incremental cost (spend change).
`n.sims`	(integer) number of simulations to draw.
`...`	arguments passed to function 'DoTBRAnalysis'.

Value

A TBRROASAnalysisFit object.

Note

A generic S3 method.

See Also

DoTBRROASAnalysis.GeoExperimentData.

`DoTBRROASAnalysis.GeoExperimentData` *Performs a TBR ROAS Analysis.*

Description

Performs a TBR ROAS Analysis.

Usage

```
## S3 method for class 'GeoExperimentData'
DoTBRROASAnalysis(obj, model, response, cost,
  n.sims = 1e+05, ...)
```

Arguments

`obj`	a GeoExperimentData object.
`model`	(string) id of the TBR to use.
`response`	(string) name of the column of the response variable.
`cost`	(string) name of the column of the cost variable, or alternatively a real number specifying the exact total incremental cost (spend change).
`n.sims`	(integer) number of simulations to draw. If cost is constant and n.sims == 1, only the point estimates will be calculated. If cost is not constant and n.sims == 1, an error is thrown.
`...`	arguments passed to function 'DoTBRAnalysis'.

Value

A TBRROASAnalysisFit object.

See Also

DoTBRROASAnalysis.GeoExperimentData.

`EstimateIncremental` *Estimates the difference of a metric between the test and counterfactual between the pre and the post periods.*

Description

Estimates the difference of a metric between the test and counterfactual between the pre and the post periods.

Usage

```
EstimateIncremental(obj, ...)
```

Arguments

`obj`	an object.
`...`	further arguments passed to or from other methods.

Value

An object representing the adjusted incremental values

Note

A generic S3 method.

See Also

EstimateIncremental.GBRROASAnalysisData.

EstimateIncremental.GBRROASAnalysisData

Estimates the incremental response or cost of the test geos during the test period.

Description

Estimates the incremental response or cost of the test geos during the test period.

Usage

```
## S3 method for class 'GBRROASAnalysisData'
EstimateIncremental(obj,
  variable = c("response", "cost"), ...)
```

Arguments

`obj`	a GBRROASAnalysisData object.
`variable`	(string) ’response’ or ’cost’.
`...`	ignored.

Value

A vector of (adjusted) incremental values, of length `nrow(obj)`.

Note

The incremental metric (response or cost) is calculated as the difference between the actual observed metric and the counterfactual during the test period. The counterfactual is defined as the prediction of the metric, using a model that uses only the control geos as the training data set. The incremental cost is defined to be zero for the control geos. In case there is not enough data to model the counterfactual (happens when all pre-test data are constant, usually zero), the counterfactual is defined to be simply the metric during the pre-test period.

See Also

EstimateIncremental (generic).

`ExperimentPeriods` *Constructs an ExperimentPeriods object.*

Description

Constructs an ExperimentPeriods object.

Usage

```
ExperimentPeriods(period.dates, period.names = NULL,
  date.format = "%Y-%m-%d")
```

Arguments

`period.dates`	(a character or Date vector); start dates of each period, plus the last date of the experiment. The first period is the pretest period, after which there must be at least one test period (there can be more than one test period); the length must be at least 3.
`period.names`	(character or NULL or a vector of nonempty strings) optional names of the periods. By default, names 'Pretest' and 'Test' (or 'Test1', 'Test2', ...) are used.
`date.format`	(string) format for the dates if provided in character format.

Value

An ExperimentPeriods object.

Note

The periods must be consecutive and each period must be at least of length 1 (day). No gaps can be specified. It is however possible to define a ('dummy') test period that is not included in the analyses.

`ExtractExperimentPeriods` *Extracts an ExperimentPeriods object.*

Description

Extracts an ExperimentPeriods object.

Usage

```
ExtractExperimentPeriods(obj, ...)
```

Arguments

`obj`	an object.
`...`	further arguments passed on to methods.

Value

An ExperimentPeriods object.

Note

A generic S3 method.

See Also

ExtractExperimentPeriods.GeoTimeseries.

`ExtractExperimentPeriods.GeoTimeseries`
Extracts a ExperimentPeriods object from a GeoTimeseries.

Description

Extracts a ExperimentPeriods object from a GeoTimeseries.

Usage

```
## S3 method for class 'GeoTimeseries'
ExtractExperimentPeriods(obj, strict = TRUE, ...)
```

Arguments

`obj`	a GeoTimeseries object with the column 'period'.
`strict`	(flag) if FALSE, the function returns NULL if the column 'period' does not exist. Otherwise, throws an error.
`...`	ignored.

Value

An ExperimentPeriods object.

See Also

ExtractExperimentPeriods (generic).

`ExtractGeoAssignment` *Extracts a GeoAssignment object.*

Description

Extracts a GeoAssignment object.

Usage

```
ExtractGeoAssignment(obj, ...)
```

Arguments

`obj`	an object.
`...`	further arguments passed on to methods.

Value

A GeoAssignment object.

Note

A generic S3 method.

See Also

ExtractGeoAssignment.GeoTimeseries.

`ExtractGeoAssignment.GeoExperimentData` *Extracts the associated GeoAssignment object.*

Description

Extracts the associated GeoAssignment object.

Usage

```
## S3 method for class 'GeoExperimentData'
ExtractGeoAssignment(obj, ...)
```

Arguments

`obj`	a GeoExperimentData object.
`...`	ignored.

Value

The GeoAssignment object associated with the GeoExperimentData object, if it exists; if it has not been set, returns NULL.

See Also

ExtractGeoAssignment.GeoTimeseries.

`ExtractGeoAssignment.GeoTimeseries` *Attempts to extract a GeoAssignment object from a GeoTimeseries.*

Description

Attempts to extract a GeoAssignment object from a GeoTimeseries.

Usage

```
## S3 method for class 'GeoTimeseries'
ExtractGeoAssignment(obj, strict = TRUE, ...)
```

Arguments

`obj`	a GeoTimeseries object with the column 'geo.group'.
`strict`	(flag) if FALSE, the function returns NULL if the column 'geo.group' does not exist. Otherwise, throws an error.
`...`	ignored.

Value

A GeoAssignment object.

Note

Finds all unique pairs of ('geo', 'geo.group') in the GeoTimeseries. 'geo.group' can have missing values.

See Also

ExtractGeoAssignment (generic).

ExtractGeos *Extract a Geos object from an object.*

Description

Extract a Geos object from an object.

Usage

```
ExtractGeos(obj, ...)
```

Arguments

`obj`	an object.
`...`	other arguments passed to the methods.

Value

A ’Geos’ object.

ExtractGeos.GeoTimeseries *Extract a Geos object from a GeoTimeseries.*

Description

Extract a Geos object from a GeoTimeseries.

Usage

```
## S3 method for class 'GeoTimeseries'
ExtractGeos(obj, volume = NULL, ...)
```

Arguments

`obj`	a GeoTimeseries object.
`volume`	(string) name of a metric in the GeoTimeseries over which to generate the ’volume’ column. If omitted, the first metric is used. The volumes have to be non-negative.
`...`	ignored.

Value

A Geos object, with the average weekly volume of all metrics in ’obj’.

`ExtractGeoStrata` *Extract a GeoStrata object from an object.*

Description

Extract a GeoStrata object from an object.

Usage

```
ExtractGeoStrata(obj, ...)
```

Arguments

`obj`	some object for which the method 'ExtractGeos' exists.
`...`	other arguments passed to the methods.

Value

A GeoStrata object.

`ExtractGeoStrata.GeoExperimentData` *Extract a GeoStrata object from a GeoExperimentData object.*

Description

Extract a GeoStrata object from a GeoExperimentData object.

Usage

```
## S3 method for class 'GeoExperimentData'
ExtractGeoStrata(obj, volume = NULL, ...)
```

Arguments

`obj`	a GeoExperimentData object.
`volume`	(string) name of a metric in the GeoExperimentData over which to generate the 'volume' column.
`...`	arguments passed to 'GeoStrata'.

Value

A GeoStrata object, obtained by first extracting a Geos object from 'obj'.

`ExtractGeoStrata.GeoTimeseries`
Extract a GeoStrata object from a GeoTimeseries.

Description

Extract a GeoStrata object from a GeoTimeseries.

Usage

```
## S3 method for class 'GeoTimeseries'
ExtractGeoStrata(obj, volume = NULL, ...)
```

Arguments

`obj`	a GeoTimeseries object.
`volume`	(string) name of a metric in the GeoTimeseries over which to generate the 'volume' column.
`...`	arguments passed to 'GeoStrata'.

Value

A GeoStrata object, obtained by first extracting a Geos object from 'obj'.

`ExtractTreatmentAssignment`
Extracts a TreatmentAssignment object.

Description

Extracts a TreatmentAssignment object.

Usage

```
ExtractTreatmentAssignment(obj, ...)
```

Arguments

`obj`	an object.
`...`	further arguments passed on to methods.

Value

A TreatmentAssignment object.

Note

A generic S3 method.

See Also

SetTreatmentAssignment<-.GeoExperimentData.

`ExtractTreatmentAssignment.GeoTimeseries`

Extracts a TreatmentAssignment object from a GeoTimeseries object.

Description

Extracts a TreatmentAssignment object from a GeoTimeseries object.

Usage

```
## S3 method for class 'GeoTimeseries'
ExtractTreatmentAssignment(obj, strict = TRUE, ...)
```

Arguments

`obj`	a GeoTimeseries object with the columns 'period', 'geo.group', and 'assignment'.
`strict`	(flag) if FALSE, the function returns NULL if either of the columns 'geo.group' or 'period' does not exist. Otherwise, throws an error.
`...`	ignored.

Value

A TreatmentAssignment object.

Note

A well-defined (period, group) pair (i.e., neither of them is missing) in the data frame implies that the corresponding (date, geo) pair is part of the experiment and *must* be therefore associated a treatment condition. Otherwise, if a date or a geo is not part of the experiment, the (date, geo) pair *must* have *no* treatment condition assignment. In other words, any (period, group) pair that maps to a non-missing treatment assignment must have no missing values. Conversely, any (period, group) pair with a missing value must map to a missing treatment assignment.

See Also

ExtractTreatmentAssignment (generic).

`FormatText` *Compose text strings conditional on values of an object.*

Description

Compose text strings conditional on values of an object.

Usage

```
FormatText(x, ..., quote = "'")
```

Arguments

`x`	an integer-valued numeric scalar or a logical vector. In the former case, indicates a number of items; in the latter case, indicates which items in a vector were affected. May have the 'names' attribute set.
`...`	one or more character vectors that are pasted together (collapsed) after concatenating the vectors one after another.
`quote`	the quote character to use.

Value

A character string.

Note

The function replaces the following special character patterns:

- `{a|b}`: 'a' if `sum(x) == 1` and 'b' otherwise.
- `{z|a|b}`: 'z' if `sum(x) == 0`, otherwise works like `{a|b}`.
- `$N`: `sum(x)`.
- `$P`: `sprintf("%.1f", 100 * mean(x))`.
- `$L`: `length(x)`.

The following patterns are replaced with comma-separated lists of:

- `$w`: `which(x)`.
- `$W`: `which(x)`, each item quoted.
- `$x`: `names(x)[which(x)]`.
- `$X`: `names(x)[which(x)]`, each item quoted.

Exception: These four patterns output only up to the Kth item, where K = getOption('FormatTextMaxOutput', default=7L).

Examples

```
## Not run:
FormatText(n, "There {is|are} {no|one|$N} item{|s}.")
FormatText(is.na(x), "Found $N NAs ($P% of all $L) in rows $w")
## End(Not run)
```

GBRROASAnalysisData

Constructs a GBRROASAnalysisData object.

Description

Constructs a GBRROASAnalysisData object.

Usage

```
GBRROASAnalysisData(x)
```

Arguments

`x`	a _plain_ 'data.frame' or an object that has to be coercible to a _plain_ 'data.frame' with the required columns 'geo', which is a character vector; numeric columns 'resp.pre', 'resp.test', 'cost.pre', 'cost.test', and a logical (TRUE/FALSE) column 'control'. Optionally the data frame can have other columns, whose names can not start with a dot ('.'). No missing values are allowed.

Value

A GBRROASAnalysisData object, which is similarly a 'data.frame' with the same columns as described above.

Note

'GBRROASAnalysisData' is a 'data.frame', with the required columns,

- `geo` a 'character'-valued vector of Geo IDs.
- `resp.pre` numeric values of the response metric in the pre-test period.
- `resp.test` numeric values of the response metric in the test period.
- `cost.pre` numeric values of the cost metric in the pre-test period.
- `cost.test` numeric values of the response metric in the test period.
- `control` : indicator ('TRUE'/'FALSE') of which columns are in the Control group.

In addition, optionally any other user-definable columns. The column 'geo' is the primary key; it is guaranteed that no rows with duplicate geos exist. The object includes the following fields stored in the attribute 'info':

- `keys`: names of the primary key columns ('geo').
- `required`: names of the required columns.
- `metrics`: names of the columns representing metric data.

`GeoAssignment` *Constructs a GeoAssignment object.*

Description

Constructs a GeoAssignment object.

Usage

```
GeoAssignment(x)
```

Arguments

`x`	a data frame with columns 'geo' (character) and 'geo.group' (integer-valued, positive). 'geo.group' indicates the group (1, 2, ...) the corresponding geo belongs to. Other columns are allowed but not checked. If 'geo' is an integer-valued numeric or factor, it is coerced to character.

Value

An object of class 'GeoAssignment'.

Note

The group numbers must be integers (integer-valued numeric are allowed as input), starting with 1. The column 'geo.group' *can* have missing values (NA), unlike 'geo', which must have no missing values, and no duplicates. A missing value in geo.group simply indicates that the mapping from a geo is not available. When associating the geo assignment with a GeoExperimentData object, the effect will be the same as if the geo with 'geo.group'==NA was not in the mapping in the first place.

`geoassignment` *Sample Geo Assignment Data*

Description

A data frame, with columns.

geo: Geo ID.

geo.group: Geo Group ID.

Usage

```
data(geoassignment)
```

Source

Simulated data.

`GeoExperimentData` *Constructs a GeoExperimentData object.*

Description

Constructs a GeoExperimentData object.

Usage

```
GeoExperimentData(geo.timeseries, periods, geo.assignment, treat.assignment)
```

Arguments

`geo.timeseries`	a GeoTimeseries object.
`periods`	an ExperimentPeriods object, specifying the start dates of each period in the experiment. Or, 'NULL' if the dates are yet unknown. Or, leave unspecified to extract this information from the column 'periods' of the geo.timeseries object; in this case the column *must* exist in the data frame.
`geo.assignment`	a GeoAssignment object, specifying the mapping from a geo to a geo group. Or, 'NULL' if the geo assignment is yet unknown. Or, leave unspecified to extract this information from the column 'geo.group' of the geo.timeseries object; in this case the column *must* exist in the data frame.
`treat.assignment`	a TreatmentAssignment object, specifying the mapping from (period, geo) to a treatment assignment condition (treatment intervention type). Or, 'NULL' if the treatment assignment is yet unknown. Or, leave unspecified to extract this information from the column 'assignment' of the geo.timeseries object; in this case the column *must* exist in the data frame.

Value

A GeoExperimentData object.

Note

If any of the arguments 'periods', 'geo.assignment', or 'treat.assignment' is not specified, a column 'period', 'geo.group', or 'assignment', respectively, *must* be present in the 'geo.timeseries' object. If any of the arguments 'periods', 'geo.assignment', or 'treat.assignment' is NULL, the corresponding column ('period', 'geo.group', or 'assignment') in the resulting GeoExperimentData data frame will have only 'NA's. If the GeoTimeseries object has columns 'period', 'geo.group', or 'assignment', they will be overwritten by the values given by the objects 'periods', 'geo.assignment', and 'treat.assignment'. The resulting object *may* have undefined 'periods', 'geo.assignment', or 'treat.assignment'. The missing parts may be filled in by using the functions 'SetExperimentPeriods', 'SetGeoAssignment', and 'SetTreatmentAssignment'.

`GeoExperimentPreanalysisData` *Creates a GeoExperimentPreanalysisData object.*

Description

Creates a GeoExperimentPreanalysisData object.

Usage

```
GeoExperimentPreanalysisData(obj, period.lengths, geos, recycle = TRUE)
```

Arguments

`obj`	a GeoTimeseries object.
`period.lengths`	an integer-valued vector of length 2 or 3, denoting the lengths (in days) of pre-period, test, and the (optional) cooldown periods, respectively. The test period must be at least 7 days; the pre-period must be at least as long as the test period. The cooldown period can be 0 or more days; if it is zero, it will be ignored as if there were only 2 periods.
`geos`	(GeoStrata or GeoAssignment object) object to use for choosing the geo groups at the time of simulating (using SimulateGeoExperimentData); if 'geos' is a GeoAssignment object, then the geo assignment will be fixed; iteration; if 'geos' is a GeoStrata, then the geo assignment will be generated by a call to 'Randomize'.
`recycle`	(flag) if TRUE, creates an augmented data set in order to create a larger time series, reusing data from the head of the time series.

Value

A GeoExperimentPreanalysis object, which inherits from GeoExperimentData, with NAs in 'geo.group', 'period', and 'assignment'. The default treatment assignment (spend change only for period 1 and group 2) is stored into the object.

Note

A GeoExperimentPreanalysisData object stores a historical data set and the information about the length of the experiment periods. It is used as the generator of pseudo-data sets for simulation purposes. See also: 'SimulateGeoExperimentData' to generate a GeoExperimentData object.

`Geos` *Constructor for Geos objects.*

Description

Constructor for Geos objects.

Usage

```
Geos(x, volume = NULL)
```

Arguments

`x`	a data frame with column 'geo' and optionally other columns. Each 'geo' must be unique.
`volume`	name of the column that represents the 'volume' of the geo. The proportion of volume is computed. If omitted, the assumed volumes will be 1 for each geo. The volumes have to be non-negative and sum up to a positive value.

Value

An object of class 'Geos', which is a data frame with the column 'geo', the column 'proportion' and 'volume', and other optional columns. The rows are sorted by the descending order of 'volume'.

`GeoStrata` *Constructor for GeoStrata objects.*

Description

Constructor for GeoStrata objects.

Usage

```
GeoStrata(geos, n.groups = 2, group.ratios = rep(1, length.out = n.groups))
```

Arguments

`geos`	a 'Geos' object.
`n.groups`	number of groups. At least 2 and at most the number of geos.
`group.ratios`	(integer vector of length n.groups) vector of ratios of the sizes of each group. By default each group is assumed to have equal ratios. The sum of these numbers also imply the size of a stratum. For example, c(2, 1) implies that group 1 should be 2 times larger than group 2, and the stratum size is 2 + 1 = 3. Note: the ratios do not have to be normalized to have greatest common divisor 1. For example, c(4, 2) implies that the ratio of group sizes is 2:1 but the stratum size is 4 + 2 = 6.

Value

A 'GeoStrata' object that inherits from 'Geos'. There is an extra column 'stratum' that indicates the stratum number to be used in randomization, and column 'geo.group' for fixing the geo-to-group mapping.

Note

GeoStrata objects are used for (stratified) randomization of geos into groups. The geos are sorted by their 'volume' (definable by the user) and then divided into strata of size n.groups (column 'stratum'). This object has also a column 'geo.group', which offers the possibility to fix certain geos to certain groups. By default, this column is filled with NAs, indicating that none of the geos are mapped to any groups. The randomization itself is done by the method 'Randomize'.
Any individual geo -> geo.group mappings should be fixed by using the 'SetGeoGroup<-' method on a GeoStrata object.
A stratum number '0' indicates a geo that is excluded from the scheme stratification. Any geo that is mapped to group '0' will have stratum number '0'; for example if geo 2 was omitted (geo.groups were NA, 0, NA, NA, NA, ...) with group.ratios c(2, 1), the strata would be assigned as 1, 0, 1, 1, 2, 2, 2, 3, 3, 3, 4, ...
Setting 'group.ratios' to some other value than the default 1,1,... enables creating groups that have different sizes. The stratum size is then determined by the sum of the number in 'group.ratios'. For example, group.ratios=c(1, 2) implies a ratio of 1:2. Each stratum has size 3; the 3 geos in this stratum are assigned a random sample with replacement from the set 1,2,2. Similarly, c(3, 1) implies that group 1 will be on average 3 times as large as group 2.
Note: the ratios do not have to be normalized to have greatest common divisor 1. For example, c(4, 2) implies that the ratio of group sizes is 2:1 but the stratum size is 4 + 2 = 6.

See Also

'Randomize', 'SetGeoGroup<-'.

`GeoTimeseries` *Constructs a GeoTimeseries object.*

Description

Constructs a GeoTimeseries object.

Usage

```
GeoTimeseries(x, metrics = character(0), date.format = "%Y-%m-%d")
```

Arguments

`x`	a _plain_ 'data.frame' or an object that has to be coercible to a _plain_ 'data.frame' with columns 'date', 'geo'. 'date' must be a Date object or a character vector or factor coercible to Date, and 'geo' must be a character vector, factor, or integer-valued. All columns that start with a dot ('.') are removed. If 'geo' is an

	integer-valued numeric or factor, it is coerced to character silently without error. If 'date' is a character or factor, it is silently coerced to Date without error. An error is output if the date conversion fails.
`metrics`	(character) column names that point to numeric columns. At least one metric must be specified. All metrics have to be not all NA.
`date.format`	(string, optional) format of the column 'date' in the form understood by as.Date(). Used only if the 'date' column is of character type.

Value

A GeoTimeseries object.

Note

'GeoTimeseries' is a 'data.frame', with the required columns,

- `date` a 'Date' vector.
- `geo` a 'character'-valued vector of Geo IDs.
- metrics : one or more numeric metrics.

In addition, optionally any other user-definable columns. (These are ignored by the specified methods but may be convenient for the user). Three columns are added, for convenience:

- `.weekday` : day number (1 = Monday, 2 = Tuesday, ..., 7 = Sunday)
- `.weeknum` : week number in the year from 0 to 53.
- `.weekindex` : absolute index of week, year + weeknum (e.g., '201542')

These should be convenient for generating totals and averages, using the `aggregate` method. The columns 'date' and 'geo' form the primary keys: it is guaranteed that no duplicate ('date', 'geo') pairs exist. The object includes fields stored in the attribute 'info':

- `metrics`: names of the metric columns.
- `other`: names of the other user-supplied columns.

`GetGeoGroup` *Returns the geo-to-geo group mapping.*

Description

Returns the geo-to-geo group mapping.

Usage

```
GetGeoGroup(obj, geo = NULL, ...)
```

Arguments

`obj`	an object with the columns ’geo’ and ’geo.group’.
`geo`	(character vector or NULL) names of geos for which to obtain the geo group ids.
`...`	other arguments passed on to the methods.

Value

A named integer-valued vector, with the geo names in the names attribute, mapping geos to geo group ids.

`GetGeoGroup.GeoAssignment` *Returns the geo group of a geo in a GeoStrata object.*

Description

Returns the geo group of a geo in a GeoStrata object.

Usage

```
## S3 method for class 'GeoAssignment'
GetGeoGroup(obj, geo = NULL, ...)
```

Arguments

`obj`	a GeoAssignment object.
`geo`	(character vector or NULL) names of geos for which to obtain the geo group ids.
`...`	ignored.

Value

A named integer-valued vector, with the geo names in the names attribute, mapping geos to geo group ids.

Note

If a specified ’geo’ does not exist in the GeoAssignment object, the corresponding geo group will be an NA.

GetGeoGroup.GeoExperimentData

Returns the geo group of a geo in a GeoExperimentData object.

Description

Returns the geo group of a geo in a GeoExperimentData object.

Usage

```
## S3 method for class 'GeoExperimentData'
GetGeoGroup(obj, geo = NULL, ...)
```

Arguments

`obj`	a GeoExperiment object.
`geo`	(character vector or NULL) names of geos for which to obtain the geo group ids.
`...`	ignored.

Value

A named integer-valued vector, with the geo names in the names attribute, mapping geos to geo group ids.

GetGeoGroup.GeoStrata

Returns the geo group of a geo in a GeoStrata object.

Description

Returns the geo group of a geo in a GeoStrata object.

Usage

```
## S3 method for class 'GeoStrata'
GetGeoGroup(obj, geo = NULL, ...)
```

Arguments

`obj`	a GeoStrata object.
`geo`	(character vector or NULL) names of geos for which to obtain the geo group ids.
`...`	ignored.

Value

A named integer-valued vector, with the geo names in the names attribute, mapping geos to geo group ids.

`GetGeoNames` *Extracts the names of the geos from the object.*

Description

Extracts the names of the geos from the object.

Usage

```
GetGeoNames(obj, groups = NULL)
```

Arguments

`obj`	an object.
`groups`	(NULL, or integer vector) id number(s) of the groups whose geos to obtain, or NULL for all geos. NA is allowed.

Value

A character vector of unique geo identifiers.

`GetGeoNames.GeoAssignment`
Extracts the names of the geos from a GeoAssignment object.

Description

Extracts the names of the geos from a GeoAssignment object.

Usage

```
## S3 method for class 'GeoAssignment'
GetGeoNames(obj, groups = NULL)
```

Arguments

`obj`	a GeoAssignment object.
`groups`	(NULL or integer vector) id number(s) of the groups whose geos to obtain, or NULL for all geos. NA is allowed.

Value

A character vector of unique geo identifiers, sorted by their name. May be empty if none of the groups specified are found in the object.

GetGeoNames.GeoExperimentData

Extracts the unique names of the geos from a GeoAssignment object.

Description

Extracts the unique names of the geos from a GeoAssignment object.

Usage

```
## S3 method for class 'GeoExperimentData'
GetGeoNames(obj, groups = NULL)
```

Arguments

`obj`	a GeoTimeseries (including GeoExperimentData) object.
`groups`	(NULL or integer vector) id number(s) of the groups whose geos to obtain, or NULL for all geos. NA is allowed.

Value

A character vector of unique geo identifiers, sorted. Can be empty if there are no geos matching 'groups'. Will be empty if the geo assignment object does not exist in the object.

GetGeoNames.Geos

Extracts the unique names of the geos from a Geos object.

Description

Extracts the unique names of the geos from a Geos object.

Usage

```
## S3 method for class 'Geos'
GetGeoNames(obj, groups = NULL)
```

Arguments

`obj`	a Geos object.
`groups`	should be NULL, otherwise an error occurs.

Value

A character vector of unique geo identifiers, sorted. Cannot be empty.

`GetGeoNames.GeoStrata`

Extracts the unique names of the geos from a GeoStrata object.

Description

Extracts the unique names of the geos from a GeoStrata object.

Usage

```
## S3 method for class 'GeoStrata'
GetGeoNames(obj, groups = NULL)
```

Arguments

`obj`	a GeoStrata object.
`groups`	(NULL, integer vector) id number(s) of the groups whose geos to obtain, or NULL for all geos. NA is allowed.

Value

A character vector of unique geo identifiers, sorted.

`GetGeoNames.GeoTimeseries`

Extracts the unique names of the geos from a GeoAssignment object.

Description

Extracts the unique names of the geos from a GeoAssignment object.

Usage

```
## S3 method for class 'GeoTimeseries'
GetGeoNames(obj, groups = NULL)
```

Arguments

`obj`	a GeoTimeseries object.
`groups`	should be NULL, otherwise an error occurs.

Value

A character vector of unique geo identifiers, sorted. Can be empty if there are no geos matching 'groups'.

`GetInfo` *Get a value of the 'info' attribute of an object.*

Description

Get a value of the 'info' attribute of an object.

Usage

```
GetInfo(obj, field)
```

Arguments

`obj`	any R object.
`field`	(string) name of the field the value of which to retrieve.

Value

The requested value (object).

`GetMessageContextString` *Get the current message context string.*

Description

Get the current message context string.

Usage

```
GetMessageContextString()
```

Value

The current message context string. If not available, returns an empty string ("").

Note

Pastes the strings together with '>'.

`GetModelIds` *Returns the list of available model identifiers.*

Description

Returns the list of available model identifiers.

Usage

```
GetModelIds()
```

Value

A character vector of the available model IDs. The 'names' attribute shows the methodology ID (gbr, tbr).

`GetTBRPlotPanelNames` *Returns the names of available TBR panels in the default order of plotting.*

Description

Returns the names of available TBR panels in the default order of plotting.

Usage

```
GetTBRPlotPanelNames()
```

Value

A character vector of the available panel IDs.

`GetWeeklyAverages` *Calculates the weekly averages of the metrics per geo.*

Description

Calculates the weekly averages of the metrics per geo.

Usage

```
GetWeeklyAverages(obj, ...)
```

Arguments

`obj`	an object.
`...`	arguments passed on to the methods.

Value

A data frame with one row per ’geo’, and the weekly averages of each metric in the columns.

Note

A generic S3 method.

See Also

GetWeeklyAverages.GeoTimeseries.

`GetWeeklyAverages.GeoTimeseries` *Calculates the weekly averages of all metrics per geo.*

Description

Calculates the weekly averages of all metrics per geo.

Usage

```
## S3 method for class 'GeoTimeseries'
GetWeeklyAverages(obj, na.rm = TRUE, ...)
```

Arguments

`obj`	a GeoTimeseries object.
`na.rm`	(flag) remove NAs?
`...`	ignored.

Value

A data frame with one row per 'geo', and the weekly averages of each metric in the columns. Sorted by the name of the geo.

Note

Does not handle incomplete weeks. Each week is supposed to have the complete total sales of the week. If for example the first and last weeks of a daily data set are incomplete, the weekly average will be underestimated. Works for both weekly and daily data. The averages are calculated by first calculating the weekly sums for each geo, and then taking the averages over the weeks. NAs are removed by default.

`is.character.vector` *Tests whether 'x' is a nonempty character vector.*

Description

Tests whether 'x' is a nonempty character vector.

Usage

```
is.character.vector(x)
```

Arguments

`x`	object to test.

Value

TRUE if and only if 'x' is a character vector of positive length, otherwise FALSE.

`is.empty.string` *Tests each component of 'x' to see whether the string is empty.*

Description

Tests each component of 'x' to see whether the string is empty.

Usage

```
is.empty.string(x)
```

Arguments

`x`	(character) a vector to test.

Value

A logical vector of the same length as 'x', with 'TRUE' if and only if the string is empty, NA if the string is NA, otherwise FALSE. If 'x' is not character, throws an error.

Note

NA maps to NA.

`is.geo.group.number` *Tests each component of the vector 'x' for valid geo group number.*

Description

Tests each component of the vector 'x' for valid geo group number.

Usage

```
is.geo.group.number(x)
```

Arguments

`x`	object to test.

Value

A logical vector of the same length as 'x', with 'TRUE' if and only if the component is nonnegative integer-valued numeric value, NA if the value is NA, otherwise FALSE. If 'x' is not integer-valued, throws an error.

Note

NA in a component returns NA. These are supposed to be tested separately. The value '0' has a special meaning: it indicates a geo that has been omitted from the experiment.

`is.inf` *Tests whether 'x' is Inf or -Inf.*

Description

Tests whether 'x' is Inf or -Inf.

Usage

```
is.inf(x)
```

Arguments

`x`	object to test.

Value

TRUE if and only if 'x' is Inf or -Inf.

`is.integer.valued` *Tests whether 'x' is an integer-valued numeric vector.*

Description

Tests whether 'x' is an integer-valued numeric vector.

Usage

```
is.integer.valued(x)
```

Arguments

`x`	object to test.

Value

TRUE if and only if 'x' is either of type integer, or numeric and all components coercible to integer, otherwise FALSE.

Note

NAs are allowed.

`is.logical.vector` *Tests whether 'x' is a nonempty logical vector.*

Description

Tests whether 'x' is a nonempty logical vector.

Usage

```
is.logical.vector(x)
```

Arguments

`x`	object to test.

Value

TRUE if and only if 'x' is a logical vector of positive length, otherwise FALSE.

`is.named.list` *Tests whether 'x' is a named list.*

Description

Tests whether 'x' is a named list.

Usage

```
is.named.list(x)
```

Arguments

`x`	object to test.

Value

TRUE if and only if 'x' is a plain list with a names attribute with non-empty strings for names, otherwise FALSE.

`is.nonempty.string` *Tests whether 'x' is a nonempty string (and not NA).*

Description

Tests whether 'x' is a nonempty string (and not NA).

Usage

```
is.nonempty.string(x)
```

Arguments

`x`	object to test.

Value

TRUE if and only if 'x' is a character vector of length 1 with a component that is nonempty and non-NA, otherwise FALSE.

`is.numeric.vector` *Tests whether 'x' is a nonempty numeric vector.*

Description

Tests whether 'x' is a nonempty numeric vector.

Usage

```
## S3 method for class 'vector'
is.numeric(x)
```

Arguments

`x`	object to test.

Value

TRUE if and only if 'x' is a numeric vector of positive length, otherwise FALSE.

`is.period.number` *Tests each component of the vector 'x' for valid experiment period number.*

Description

Tests each component of the vector 'x' for valid experiment period number.

Usage

```
is.period.number(x)
```

Arguments

`x`	object to test.

Value

A logical vector of the same length as 'x', with 'TRUE' if and only if the component is an integer-valued numeric value, NA if the value is NA, otherwise FALSE. If 'x' is not integer-valued, throws an error.

Note

NA in a component returns NA. These are supposed to be tested separately. Negative period numbers are allowed.

`is.plain.list` *Tests whether 'x' is a plain list (not an object).*

Description

Tests whether 'x' is a plain list (not an object).

Usage

```
is.plain.list(x)
```

Arguments

`x`	object to test.

Value

TRUE if and only if 'x' is a list but not an object. otherwise FALSE.

`is.real.number` *Tests whether 'x' is a real number.*

Description

Tests whether 'x' is a real number.

Usage

```
is.real.number(x)
```

Arguments

`x`	object to test.

Value

TRUE if and only if 'x' is a numeric scalar, not an NA, and not infinite, otherwise FALSE.

`is.treatment.assignment` *Tests each component of the vector 'x' for valid treatment assignment symbols.*

Description

Tests each component of the vector 'x' for valid treatment assignment symbols.

Usage

```
is.treatment.assignment(x)
```

Arguments

`x`	object to test.

Value

A logical vector of the same length as 'x', with 'TRUE' if and only if the string is a valid treatment assignment symbol, NA if the string is NA, otherwise FALSE. If 'x' is not character, throws an error.

Note

NA in a component returns NA. These are supposed to be tested separately.

`is.vector.of.nonempty.strings` *Tests whether 'x' is a character vector of nonempty strings.*

Description

Tests whether 'x' is a character vector of nonempty strings.

Usage

```
is.vector.of.nonempty.strings(x)
```

Arguments

`x`	object to test.

Value

TRUE if and only if 'x' is a vector of nonempty strings none of which is an NA, otherwise FALSE. Empty vectors yield FALSE.

`IsFixedRandomization` *Returns TRUE iff randomizing the geostrata can lead to multiple outcomes.*

Description

Returns TRUE iff randomizing the geostrata can lead to multiple outcomes.

Usage

```
IsFixedRandomization(geostrata)
```

Arguments

`geostrata`	a GeoStrata object.

Value

TRUE if randomizing the geostrata can only lead to a single GeoAssignment, FALSE otherwise.

IsInPeriod — *[internal] Returns a logical vector indicating which of the rows are in the specified period(s).*

Description

[internal] Returns a logical vector indicating which of the rows are in the specified period(s).

Usage

```
IsInPeriod(obj, periods)
```

Arguments

`obj`	some object.
`periods`	names of the periods.

Value

A logical vector of length `nrow(obj)`, TRUE for a row that is within the given period(s), FALSE otherwise. No NAs.

IsInPeriod.TBRAnalysisData — *[internal] Returns a logical vector indicating which of the rows are in the specified period(s) in the TBRAnalysisData object.*

Description

[internal] Returns a logical vector indicating which of the rows are in the specified period(s) in the TBRAnalysisData object.

Usage

```
## S3 method for class 'TBRAnalysisData'
IsInPeriod(obj, periods)
```

Arguments

`obj`	a TBRAnalysisData object.
`periods`	(vector of strings) names of the periods.

Value

A logical vector of length `nrow(obj)`, TRUE for a row that is within the given period(s), FALSE otherwise. No NAs.

IsInPeriod.TBRAnalysisFitTbr1

[internal] Returns a logical vector indicating which of the rows are in the specified period(s) in the TBRAnalysisData object.

Description

[internal] Returns a logical vector indicating which of the rows are in the specified period(s) in the TBRAnalysisData object.

Usage

```
## S3 method for class 'TBRAnalysisFitTbr1'
IsInPeriod(obj, periods)
```

Arguments

`obj`	a TBRAnalysisFitTbr1 object.
`periods`	(vector of strings) names of the periods.

Value

A logical vector of length `nrow(obj)`, TRUE for a row that is within the given period(s), FALSE otherwise. No NAs.

IsInPeriod.TBRQuantiles

[internal] Returns a logical vector indicating which of the rows are in the specified period(s) in a TBRQuantiles object.

Description

[internal] Returns a logical vector indicating which of the rows are in the specified period(s) in a TBRQuantiles object.

Usage

```
## S3 method for class 'TBRQuantiles'
IsInPeriod(obj, periods)
```

Arguments

`obj`	a TBRQuantiles object.
`periods`	(vector of strings) names of the periods.

Value

A logical vector of length `nrow(obj)`, TRUE for a row that is within the given period(s), FALSE otherwise. No NAs.

Note

TBRQuantiles consists of simulations with rows only for the prediction period, not for the preanalysis period.

`IsInPeriod.TBRROASAnalysisFit` *[internal] Returns a logical vector indicating which of the rows are in the specified period(s) in the TBRROASAnalysisFit object.*

Description

[internal] Returns a logical vector indicating which of the rows are in the specified period(s) in the TBRROASAnalysisFit object.

Usage

```
## S3 method for class 'TBRROASAnalysisFit'
IsInPeriod(obj, periods)
```

Arguments

`obj`	a TBRROASAnalysisFit object.
`periods`	(vector of strings) names of the periods.

Value

A logical vector of length `nrow(obj)`, TRUE for a row that is within the given period(s), FALSE otherwise. No NAs.

Note

TBRROASAnalysisFit consists of simulations with rows only for the prediction period, not for the preanalysis period.

`IsStratumCompatibleWithRatios` *Checks whether strata are compatible with group.ratios.*

Description

Checks whether strata are compatible with group.ratios.

Usage

```
IsStratumCompatibleWithRatios(geo.group, group.ratios)
```

Arguments

`geo.group`	(integer vector of length equal to number of geos) a vector of geo group numbers (may be NAs, but zeros are not allowed).
`group.ratios`	(integer vector of length equal to number of geo groups) vector of ratios of the sizes of each group.

Value

TRUE if geo.group is compatible with group.ratios, i.e. no group has been assigned more geos than what group.ratios allows. FALSE otherwise.

`MapGeoGroups<-` *Map or merge geo group numbers into new group ids.*

Description

Map or merge geo group numbers into new group ids.

Usage

```
MapGeoGroups(obj) <- value
```

Arguments

`obj`	an object.
`value`	(integer vector, NA allowed, NULL or empty integer vector allowed) mapping of old group numbers to new ones. The length must be exactly equal to the number of old groups in the object. The old numbers match the positions of the index, and the new ones are the values in the vector. For example, c(2, 1) maps 1->2 and 2->1; c(3, 2, 1) maps 1->3, 3->1 and 2 stays unchanged; '1:2' has no effect; NOTE: c(1, 1) merges groups 1 and 2 into 1! If the 'geo.group' is NA in the object, its value will be NOT changed. If the data has only NAs, the only mapping that is allowed is an empty vector, resulting in no change in the original object. The special value '0' (group of excluded geos) will also be unchanged.

Value

An object identical to ’obj’ except the ’geo.groups’ column has (possibly) changed.

`MapGeoGroups<-.GeoAssignment`

Map or merge geo group numbers into new group ids.

Description

Map or merge geo group numbers into new group ids.

Usage

```
## S3 replacement method for class 'GeoAssignment'
MapGeoGroups(obj) <- value
```

Arguments

`obj`	a GeoAssignment object.
`value`	(integer vector, NA allowed, NULL or empty integer vector allowed) mapping of old group numbers to new ones. The length must be exactly equal to the number of old groups in the object. The old numbers match the positions of the index, and the new ones are the values in the vector. For example, c(2, 1) maps 1->2 and 2->1; c(3, 2, 1) maps 1->3, 3->1 and 2 stays unchanged; ’1:2’ has no effect; NOTE: c(1, 1) merges groups 1 and 2 into 1! If the ’geo.group’ is NA in the object, its value will be NOT changed. If the data has only NAs, the only mapping that is allowed is an empty vector or NULL, resulting in no change in original object. The special value ’0’ (group of excluded geos) will also be unchanged.

Value

A GeoAssignment object, identical to ’obj’ except the ’geo.groups’ column has (possibly) changed.

Note

If all group ids in ’geo.group’ are NA, nothing changes as ’NAs’ cannot be mapped. Try ’SetGeoGroup<-’ instead, to map specific geos first into group ids.
The group number ’0’ (group of excluded geos) remains unchanged. It is not possible to map this group to another. For that purpose, map geos explicitly to desired groups using the replacement method ’SetGeoGroup<-’.
It is however possible to exclude a complete group by mapping it to 0 by including 0 in the map, for example c(1, 2, 0) maps group 3 to 0, leaving group numbers 1 and 2 unchanged.

`MapGeoGroups<-.GeoExperimentData`

Map or merge geo group numbers into new group ids.

Description

Map or merge geo group numbers into new group ids.

Usage

```
## S3 replacement method for class 'GeoExperimentData'
MapGeoGroups(obj) <- value
```

Arguments

`obj`	a GeoExperimentData object.
`value`	(integer vector or NA) mapping of old group numbers to new ones. See 'MapGeoGroups<-.GeoAssignment' for details.

Value

The GeoExperimentData object, with the column 'geo.group' affected.

`merge.GeoExperimentData`

Merges two GeoExperimentData objects.

Description

Merges two GeoExperimentData objects.

Usage

```
## S3 method for class 'GeoExperimentData'
merge(x, y, ...)
```

Arguments

`x`	a GeoExperimentData object.
`y`	a GeoExperimentData object.
`...`	ignored.

Value

A GeoExperimentData object.

Note

The object created is a GeoExperimentData that includes all the columns of x, and all the columns of y that were not in x. Columns of y that have the same name as columns of x are ignored and a warning is issued whenever y and x share some column name(s). The merging is done by kDate, kGeo, kPeriod, kGeoGroup, kAssignment.

`merge.GeoTimeseries` *Merges two GeoTimeseries object.*

Description

Merges two GeoTimeseries object.

Usage

```
## S3 method for class 'GeoTimeseries'
merge(x, y, ...)
```

Arguments

`x`	a GeoTimeseries object.
`y`	a GeoTimeseries object.
`...`	ignored.

Value

A GeoTimeseries object.

Note

The object created is a GeoTimeseries that includes all the columns of x, and all the columns of y that were not in x. Columns of y that have the same name as columns of x are ignored and a warning is issued whenever y and x share some column name(s). The merging is done by kDate, kGeo.

`Message` *Form a message, prefixed with the message context string.*

Description

Form a message, prefixed with the message context string.

Usage

```
Message(...)
```

Arguments

`...`	R vectors, usually character, to be collapsed using 'paste0'.

Value

A string.

`Messagef` *Form a message, using sprintf.*

Description

Form a message, using sprintf.

Usage

```
Messagef(fmt, ...)
```

Arguments

`fmt`	a character vector of format strings (see 'sprintf' for details).
`...`	values to be passed to 'fmt'.

Value

A string.

`plot.Geos` *Generate a (gg)plot of a Geos object.*

Description

Generate a (gg)plot of a Geos object.

Usage

```
## S3 method for class 'Geos'
plot(x, y = NULL, geom = c("bar", "point", "rect"),
  sort.by = y, log.scale = FALSE, ...)
```

Arguments

`x`	a Geos object.
`y`	(string) name of the metric to plot. If omitted, plots the volume defined in the object by default.
`geom`	(string) short name of the geom ggplot2 method to be used.
`sort.by`	(string) name of the column to sort the plot by, defaults to y. If the column contains string, alphabetical order will be used unless every string can be coerced to an integer in which case numerical (ascending) order will be used. If the column contains numerical values, numerical (descending) order will be used.
`log.scale`	(flag) plot y-axis on log scale?
`...`	ignored.

Value

A ggplot object.

`plot.GeoTimeseries` *Generate a (gg)plot of a GeoTimeseries object.*

Description

Generate a (gg)plot of a GeoTimeseries object.

Usage

```
## S3 method for class 'GeoTimeseries'
plot(x, y = NULL, by = NULL, subset = NULL,
  aggregate = FALSE, title = y, log.scale = FALSE, legend = TRUE, ...)
```

Arguments

`x`	a GeoTimeseries object.
`y`	(string) name of the metric to plot. If omitted, plots the first metric in the object by default.
`by`	(string) group by which column? Default is 'geo'.
`subset`	(vector) subset of the items in column 'by' to use.
`aggregate`	(flag) aggregate over?
`title`	(string) a title string. By default, the name of the metric to plot.
`log.scale`	(flag) plot y-axis on log scale?
`legend`	(flag) plot the legend?
`...`	ignored.

Value

A ggplot object.

Note

Sets the lower y-axis limit to zero if all 'y' values are nonnegative and log.scale='FALSE'.

`plot.TBRAnalysisFitTbr1` *Creates a graph of the TBR analysis fit.*

Description

Creates a graph of the TBR analysis fit.

Usage

```
## S3 method for class 'TBRAnalysisFitTbr1'
plot(x, y, panels = GetTBRPlotPanelNames(),
  periods = c("pretest", "prediction"), quantiles = c(0.1, 0.9),
  highlight.weeks = TRUE, date.format = "%Y-%m-%d", ...)
```

Arguments

`x`	a TBRAnalysisFitTbr1 object.
`y`	ignored.
`panels`	(vector of strings) names of the panels to be plotted.
`periods`	(vector of strings) names of the periods to show.
`quantiles`	(real vector of length 2) lower and upper quantiles of the credible interval to show.

`highlight.weeks`	(flag), alternate background shading to highlight weeks?
`date.format`	(string) date format.
`...`	further arguments passed to methods 'as.TBRPlotData' and 'plot.TBRPlotData'.

Value

A ggplot2 object.

`plot.TBRPlotData` *Creates a graph of the TBRPlotData object.*

Description

Creates a graph of the TBRPlotData object.

Usage

```
## S3 method for class 'TBRPlotData'
plot(x, y, highlight.weeks = TRUE,
  date.format = "%Y-%m-%d", ...)
```

Arguments

`x`	a TBRPlotData object.
`y`	ignored.
`highlight.weeks`	(flag), alternate background shading to highlight weeks?
`date.format`	(string) date format.
`...`	ignored.

Value

A ggplot2 object.

plot.TBRROASAnalysisFit
Creates a graph of the TBR ROAS analysis fit.

Description

Creates a graph of the TBR ROAS analysis fit.

Usage

```
## S3 method for class 'TBRROASAnalysisFit'
plot(x, y, panels = "cumulative",
  periods = "prediction", quantiles = c(0.1, 0.9), highlight.weeks = TRUE,
  date.format = "%Y-%m-%d", ...)
```

Arguments

`x`	a TBRROASAnalysisFit object.
`y`	ignored.
`panels`	(vector of strings) names of the panels to be plotted.
`periods`	(vector of strings) names of the periods to show.
`quantiles`	(real vector of length 2) lower and upper quantiles of the credible interval to show.
`highlight.weeks`	(flag), alternate background shading to highlight weeks?
`date.format`	(string) date format.
`...`	further arguments passed to methods 'as.TBRPlotData' and 'plot.TBRPlotData'.

Value

A ggplot2 object.

print.GBRROASAnalysisFitGbr1
Shows a summary of GBR analysis results.

Description

Shows a summary of GBR analysis results.

Usage

```
## S3 method for class 'GBRROASAnalysisFitGbr1'
print(x, ...)
```

Arguments

`x`	a GBRROASAnalysisFitGbr1 object.
`...`	ignored.

Value

The object itself, invisibly. As a side effect, prints the default summary of 'x' on the console.

`print.ROASPreanalysisFit` *Prints a summary of the ROASPreanalysisFit method.*

Description

Prints a summary of the ROASPreanalysisFit method.

Usage

```
## S3 method for class 'ROASPreanalysisFit'
print(x, ...)
```

Arguments

`x`	a ROASPreanalysisFit object.
`...`	arguments passed to the 'summary' method.

Value

Prints the summary of 'x' and returns the summary of 'x' invisibly.

`print.TBRROASAnalysisFit` *Shows a summary of TBR analysis results.*

Description

Shows a summary of TBR analysis results.

Usage

```
## S3 method for class 'TBRROASAnalysisFit'
print(x, ...)
```

Arguments

`x`	a TBRROASAnalysisFit object.
`...`	ignored.

Value

The object itself, invisibly. As a side effect, prints the default summary of 'x' on the console.

quantile.PosteriorSimulations

Compute the quantiles for the posterior simulations.

Description

Compute the quantiles for the posterior simulations.

Usage

```
## S3 method for class 'PosteriorSimulations'
quantile(x, probs = c(0.05, 0.1, 0.5, 0.9,
  0.95), names = TRUE, ...)
```

Arguments

`x`	a PosteriorSimulations object.
`probs`	(numeric vector) of probabilities.
`names`	(flag) if TRUE, the result has a 'names' attribute.
`...`	possibly other arguments passed on to 'quantile'.

Value

A matrix with n rows and m columns, where n = rows in 'x' and m = length of 'probs'.

`quantile.TBRAnalysisFitTbr1`
Computes quantiles of the cumulative, pointwise, or the counterfactual (posterior) distribution of the causal lift, from the beginning of pretest to the end of posttest.

Description

Computes quantiles of the cumulative, pointwise, or the counterfactual (posterior) distribution of the causal lift, from the beginning of pretest to the end of posttest.

Usage

```
## S3 method for class 'TBRAnalysisFitTbr1'
quantile(x, probs = c(0.1, 0.5, 0.9),
  distribution = c("cumulative", "pointwise", "counterfactual"), ...)
```

Arguments

`x`	a TBRAnalysisFitTbr1 object.
`probs`	(numeric vector, each >= 0 and <= 1) quantiles to calculate.
`distribution`	(string) either 'cumulative' or 'pointwise' or 'counterfactual'.
`...`	ignored.

Value

A TBRQuantiles object, which is a data frame with each row corresponding to a day in the pretest and the prediction period, with columns 'date', 'test', and the columns for the quantiles.

`quantile.TBRROASAnalysisFit`
Computes quantiles of the cumulative or pointwise distribution of the incremental ROAS, for each of the days in the pretest and the prediction period.

Description

Computes quantiles of the cumulative or pointwise distribution of the incremental ROAS, for each of the days in the pretest and the prediction period.

Usage

```
## S3 method for class 'TBRROASAnalysisFit'
quantile(x, probs = c(0.1, 0.5, 0.9),
  distribution = c("cumulative", "pointwise"), ...)
```

Arguments

`x`	a TBRROASAnalysisFit object.
`probs`	(numeric vector, each >= 0 and <= 1) quantiles to calculate.
`distribution`	(string) either 'cumulative' or 'pointwise'.
`...`	ignored.

Value

A TBRQuantiles object, which is a data frame with each row corresponding to a day in the pretest and the prediction period, with columns 'date', 'test', and the columns for the quantiles.

Note

The pretest period is included solely for the purpose of obtaining objects of conforming size from both quantile methods (that of TBRAnalysisFitTbr1 and of this one).

`Randomize` *Randomize geos to groups.*

Description

Randomize geos to groups.

Usage

```
Randomize(obj, ...)
```

Arguments

`obj`	an object.
`...`	arguments passed to other methods.

Value

A GeoAssignment object.

`Randomize.Geos` *Randomize geos, assigning geo groups by strata.*

Description

Randomize geos, assigning geo groups by strata.

Usage

```
## S3 method for class 'Geos'
Randomize(obj, n.groups = 2, group.ratios = rep(1, length.out
  = n.groups), ...)
```

Arguments

`obj`	a Geos object.
`n.groups`	(integer) number of groups.
`group.ratios`	(integer vector of length n.groups) vector of ratios of the sizes of each group. By default each group is assumed to have equal ratios.
`...`	ignored.

Value

A GeoAssignment object.

`Randomize.GeoStrata` *Randomize geos, assigning geo groups by strata.*

Description

Randomize geos, assigning geo groups by strata.

Usage

```
## S3 method for class 'GeoStrata'
Randomize(obj, ...)
```

Arguments

`obj`	a GeoStrata object.
`...`	ignored.

Value

A GeoAssignment object.

RenameColumns *Renames columns of a data frame.*

Description

Renames columns of a data frame.

Usage

```
RenameColumns(x, map = character(0))
```

Arguments

`x`	a data frame.
`map`	a named character vector, mapping old column names to new ones such that the new ones are in 'names' and the values are the old ones. For example, `c(geo='GMA', date='Week Ending')`. If 'map' has length 0, the original data frame is returned.

Value

The data frame, with the column names renamed.

ROASAnalysisResults *Constructs a ROASAnalysisResults object.*

Description

Constructs a ROASAnalysisResults object.

Usage

```
ROASAnalysisResults(estimate, lower, upper, level, incr.resp, incr.cost,
  threshold, post.prob, model)
```

Arguments

`estimate`	(number) iROAS point estimate.
`lower`	(number) lower bound of the interval estimate.
`upper`	(number or Inf) upper bound of the interval estimate.
`level`	(number between 0 and 1) confidence level.
`incr.resp`	(number) point estimate of the total incremental response.
`incr.cost`	(number) point estimate of the total incremental cost.

`threshold`	(number) threshold for calculating the posterior probability.
`post.prob`	(number) posterior probability Pr(beta2 > threshold|data).
`model`	(string) id string indicating the model used.

Value

An object of class ROASAnalysisResults, which also inherits from 'ROASAnalysisResults'. This is a data frame with the columns:

- `estimate`: point estimate of the incremental Return On Ad Spend.
- `precision`: confidence/credible interval half-width. Calculated as 0.5 times the difference between upper and lower bounds; if upper is Inf, then precision is the distance between the estimate and the lower bound.
- `lower`: lower bound of the interval estimate.
- `upper`: upper bound of the interval estimate (can be Inf).
- `level`: confidence level.
- `incr.resp`: estimated incremental response.
- `incr.cost`: estimated incremental cost.
- `thres`: threshold for the posterior tail probability.
- `prob`: posterior tail probability Pr(beta2>thres|data).
- `model`: model id.

`round.ROASAnalysisResults`

Rounds the estimates to a given number of decimal places.

Description

Rounds the estimates to a given number of decimal places.

Usage

```
## S3 method for class 'ROASAnalysisResults'
round(x, digits = 1)
```

Arguments

`x`	a ROASAnalysisResults object.
`digits`	number of digital places to round to.

Value

A ROASAnalysisResults object, with estimates appropriately rounded.

salesandcost *Sales and Cost Data*

Description

A data frame with 9225 rows, with the columns:

date: A character string representing a date in the format MM/DD/YYYY.

geo: Geo id, integer.

sales: Sales for the corresponding geo and day, numeric.

cost: Cost of advertising, numeric.

Usage

```
data(salesandcost)
```

Source

Simulated data.

SetExperimentPeriods<- *Associates the object with an ExperimentPeriods object.*

Description

Associates the object with an ExperimentPeriods object.

Usage

```
SetExperimentPeriods(obj, ...) <- value
```

Arguments

`obj`	the object to change.
`...`	further arguments passed to methods.
`value`	a ExperimentPeriods object.

Value

The object that has been modified in place.

See Also

SetExperimentPeriods<-.GeoExperimentData.

`SetExperimentPeriods<-.GeoExperimentData` *Associates the object with an ExperimentPeriods object, mapping the 'date' column to a 'period' column.*

Description

Associates the object with an ExperimentPeriods object, mapping the 'date' column to a 'period' column.

Usage

```
## S3 replacement method for class 'GeoExperimentData'
SetExperimentPeriods(obj, strict = TRUE, ...) <-
  value
```

Arguments

`obj`	the object to change.
`strict`	(flag) insist that data has all specified experiment periods?
`...`	ignored.
`value`	a ExperimentPeriods object.

Value

The same object that has been modified in place; the 'period' slot is assigned with the new value and the columns 'period' and 'assignment' are changed to reflect the new date ranges.

See Also

SetExperimentPeriods<- (generic).

`SetGeoAssignment<-` *Associates the object with a GeoAssignment object.*

Description

Associates the object with a GeoAssignment object.

Usage

```
SetGeoAssignment(obj, ...) <- value
```

Arguments

`obj`	the object to change.
`...`	further arguments passed to methods.
`value`	a GeoAssignment object.

Value

The object (that has been modified in place).

See Also

SetGeoAssignment.GeoExperimentData.

`SetGeoAssignment<-.GeoExperimentData`

Associates the object with a GeoAssignment object, mapping geos into geo group numbers.

Description

Associates the object with a GeoAssignment object, mapping geos into geo group numbers.

Usage

```
## S3 replacement method for class 'GeoExperimentData'
SetGeoAssignment(obj, strict = TRUE, ...) <- value
```

Arguments

`obj`	the GeoExperimentData object to change.
`strict`	(flag) insist that all geos in the data are mapped? Also, warn if some geos are not found in the data?
`...`	ignored.
`value`	a GeoAssignment object to associate with the GeoExperimentData object.

Value

The same object that has been modified in place; the 'geo.assignment' slot is assigned with the new value and the columns 'geo.group' and 'assignment' are changed to reflect the new geo assignment.

Note

If 'value' is 'NULL', the geo assignment and the treatment assignment are removed, and their corresponding columns in the data frame are reset to 'NA'. An error is thrown if any geo in data cannot be mapped to a group (due to the GeoAssignment object not having such a mapping). Setting 'strict' to FALSE will avert this. An error is thrown if some of the geos in the mapping were not present in the data. Setting 'strict' to FALSE will avert this.

See Also

SetGeoAssignment<- (generic).

`SetGeoGroup<-` *Modifies the geo assignment in an object.*

Description

Modifies the geo assignment in an object.

Usage

```
SetGeoGroup(obj) <- value
```

Arguments

`obj`	an object with the columns ’geo’ and ’geo.group’.
`value`	a GeoAssignment object.

Value

The object with the modfied geo-to-geo.group mapping.

Note

See also: SetGeoAssignment<-.

`SetGeoGroup<-.GeoAssignment` *Modifies a GeoAssignment object.*

Description

Modifies a GeoAssignment object.

Usage

```
## S3 replacement method for class 'GeoAssignment'
SetGeoGroup(obj) <- value
```

Arguments

`obj`	a GeoAssignment object.
`value`	a GeoAssignment object.

Value

The same object, with the geo.group column modified.

`SetGeoGroup<-.GeoExperimentData`

Modifies the GeoAssignment object in the GeoExperimentData object.

Description

Modifies the GeoAssignment object in the GeoExperimentData object.

Usage

```
## S3 replacement method for class 'GeoExperimentData'
SetGeoGroup(obj) <- value
```

Arguments

`obj`	a GeoExperimentData object.
`value`	a GeoAssignment object.

Value

The same object, with the geo assignment (and therefore also the geo.group column) modified.

Note

The GeoExperimentData object must contain a valid GeoAssignment object. Use SetGeoAssignment<- to associate a GeoAssignment object to the GeoExperimentData object.

`SetGeoGroup<-.GeoStrata`

Modifies the geo assignment in a GeoStrata object.

Description

Modifies the geo assignment in a GeoStrata object.

Usage

```
## S3 replacement method for class 'GeoStrata'
SetGeoGroup(obj) <- value
```

Arguments

`obj`	a GeoStrata object.
`value`	a GeoAssignment object.

Value

The same object, with the geo.group column modified.

SetIncrementalResponse<-

Sets an incremental response for simulation purposes.

Description

Sets an incremental response for simulation purposes.

Usage

```
SetIncrementalResponse(obj, response, periods = "all") <- value
```

Arguments

`obj`	an object with the column '.spend'.
`response`	(string) name of the column which acts as the response metric.
`periods`	numbers of the periods which are affected; if this is "all", all periods are affected.
`value`	the absolute value of the incremental return on ad spend. Can also be negative. Can also be NA, in the case of which the column '.response' will be reset to NA.

Value

The object, with the modfied spend change column '.response'.

SetIncrementalResponse<-.GeoExperimentData

Sets an incremental response for simulation purposes in a GeoExperimentData object.

Description

Sets an incremental response for simulation purposes in a GeoExperimentData object.

Usage

```
## S3 replacement method for class 'GeoExperimentData'
SetIncrementalResponse(obj, response,
  periods = "all") <- value
```

Arguments

`obj`	a GeoExperimentData object with the column '.spend'.
`response`	(string) name of the column which acts as the response metric.
`periods`	numbers of the periods which are affected; if this is "all", all periods are affected.
`value`	(a real number) the incremental return on ad spend. Can be zero or negative. Can also be NA, in the case of which the column '.response' will be reset to NA.

Value

The same object, with the column '.response' modified.

Note

Creates a column '.response' if it does not exist, and registers it as a metric. The column '.response' contains the baseline response and the incremental response, which is defined as the specified incremental ROAS ('value') times the column '.spend'.

`SetInfo` *Set the values of the 'info' attribute of an object.*

Description

Set the values of the 'info' attribute of an object.

Usage

```
SetInfo(obj, ..., info = NULL)
```

Arguments

`obj`	some object.
`...`	name-value pairs to set in the list-valued attribute 'info'.
`info`	the initial value of the 'info' attribute; if specified, will re-set the value before changing the individual name-value pairs.

Value

The object ('obj') that was changed.

Note

Used only internally in this package. Do not use for data tables.

`SetMessageContextString` *Save a string representing the current context of the program flow.*

Description

Save a string representing the current context of the program flow.

Usage

```
SetMessageContextString(context)
```

Arguments

`context`	(a string) context of the message; to be displayed in (error) messages.

Value

The previous message context strings (a character vector).

`SetSpendChange<-` *Sets the spend change in a geo experiment for simulation purposes.*

Description

Sets the spend change in a geo experiment for simulation purposes.

Usage

```
SetSpendChange(obj, prop.to, periods = "all") <- value
```

Arguments

`obj`	an object with the column 'assignment'.
`prop.to`	(existing) name of the column in proportion to which the spend change will be spread across the geos.
`periods`	numbers of the periods which are affected; if this is "all", all periods are affected and the spend change will be spread across all periods that have spend change (determined by the column 'assignment').
`value`	the total absolute value of the spend change. Can also be NA, in the case of which the spend change column will be reset to NA.

Value

The object, with the modified spend change column `.spend`.

SetSpendChange<-.GeoExperimentData

Sets the spend change column in a GeoExperimentData object.

Description

Sets the spend change column in a GeoExperimentData object.

Usage

```
## S3 replacement method for class 'GeoExperimentData'
SetSpendChange(obj, prop.to, periods = "all") <-
  value
```

Arguments

`obj`	a GeoExperimentData object. The values in the 'assignment' column must not be missing.
`prop.to`	(existing) name of the column in proportion to which the spend change will be spread across the geos.
`periods`	numbers of the periods which are affected; if this is "all", all periods are affected and the spend change will be spread across all periods that have spend change (determined by the column 'assignment').
`value`	the total absolute value of the spend change. Can also be NA, in the case of which the spend change column will be reset to NA.

Value

The same object, with the column specified as the spend change column '.spend' modified.

Note

Creates a column '.spend'. The spend change type is determined by the 'assignment' column. The absolute value of the spend change will be spread across geos in the proportion of the given column ('prop.to'); the sign of the change is determined by the type of change: for treatment assignments set to 'decrease', the sign will be negative, otherwise positive.

SetTreatmentAssignment<-

Associates the object with an TreatmentAssignment object.

Description

Associates the object with an TreatmentAssignment object.

Usage

```
SetTreatmentAssignment(obj, ...) <- value
```

Arguments

`obj`	the object to change.
`...`	further arguments passed to methods.
`value`	a TreatmentAssignment object.

Value

The object (that has been modified in place).

See Also

SetTreatmentAssignment<-.GeoExperimentData.

SetTreatmentAssignment<-.GeoExperimentData

Associates the object with an TreatmentAssignment object, mapping the ('period', 'geo.group') pairs to a (treatment) 'assignment' column.

Description

Associates the object with an TreatmentAssignment object, mapping the ('period', 'geo.group') pairs to a (treatment) 'assignment' column.

Usage

```
## S3 replacement method for class 'GeoExperimentData'
SetTreatmentAssignment(obj, strict = TRUE, ...) <-
  value
```

Arguments

`obj`	the object (a GeoExperimentData object) to change.
`strict`	(flag) insist that data has all treatment combinations?
`...`	ignored.
`value`	a TreatmentAssignment object.

Value

The object, that has been modified in place.

Note

If 'strict' is TRUE, throws an error if some treatment assignments are not found in the data. If 'strict' is FALSE, no warnings or errors are thrown.

See Also

SetTreatmentAssignment<- (treatment).

`signif.ROASAnalysisResults`

Rounds the estimates to a given number of significant digits.

Description

Rounds the estimates to a given number of significant digits.

Usage

```
## S3 method for class 'ROASAnalysisResults'
signif(x, digits = 1)
```

Arguments

`x`	a ROASAnalysisResults object.
`digits`	number of significant digits to round to.

Value

A ROASAnalysisResults object, with estimates appropriately rounded.

`SimulateGeoExperimentData`
Returns a geo experiment data set.

Description

Returns a geo experiment data set.

Usage

```
SimulateGeoExperimentData(obj, ...)
```

Arguments

`obj`	a GeoExperimentPreanalysisData object.
`...`	arguments passed on to the methods.

Value

A GeoExperimentData object.

`SimulateGeoExperimentData.GeoExperimentPreanalysisData`
Returns a set number i from the GeoExperimentPreanalysisData object.

Description

Returns a set number i from the GeoExperimentPreanalysisData object.

Usage

```
## S3 method for class 'GeoExperimentPreanalysisData'
SimulateGeoExperimentData(obj,
  i = NA_integer_, ...)
```

Arguments

`obj`	a GeoExperimentPreanalysisData object.
`i`	(positive integer or NA) number of the pseudo data set; if NA, the number will be drawn from a uniform discrete distribution.
`...`	ignored.

Value

A GeoExperimentData object with the experiment periods, geo assignment, and the treatment assignment set.

Note

If the embedded 'geos' object is GeoStrata, a randomization is done and the geo assignment applied.

`SimulatePosterior` *Draws a sample from a posterior distribution.*

Description

Draws a sample from a posterior distribution.

Usage

```
SimulatePosterior(obj, n.sims = 1e+05, ...)
```

Arguments

`obj`	an object.
`n.sims`	(integer >= 2) number of simulations to draw.
`...`	other arguments passed on to the methods.

Value

A PosteriorSimulations object, which is a matrix with 'n.sims' columns. Each row corresponds to a variable.

`SimulatePosterior.TBRAnalysisFitTbr1` *Draws from the posterior distribution of the pointwise incremental effect over the test period.*

Description

Draws from the posterior distribution of the pointwise incremental effect over the test period.

Usage

```
## S3 method for class 'TBRAnalysisFitTbr1'
SimulatePosterior(obj, n.sims = 1e+05, ...)
```

Arguments

`obj`	A TBRAnalysisFitTbr1 object.
`n.sims`	(integer >= 2) number of simulations to draw.
`...`	ignored.

Value

A PosteriorSimulations object, which is a matrix with 'n.sims' columns and as many rows as there are time points in the test period. Each row corresponds to one time point in the test period, with the draws from the pointwise distribution of the incremental effect.

Note

The 'standard' noninformative prior is assumed: uniform on (beta, log sigma). Reference: Gelman et al. Bayesian Data Analysis (2nd ed.), section 14.2, formula (14.8). To obtain the simulations for the cumulative distribution, use the `cumsum` method.

`Subset` *Returns a subset of the object, ensuring that the resulting object is of the same class.*

Description

Returns a subset of the object, ensuring that the resulting object is of the same class.

Usage

```
Subset(obj, ...)
```

Arguments

`obj`	an object.
`...`	further arguments passed to or from other methods.

Value

An object of the same class.

Note

A generic S3 method.

See Also

Subset.GeoTimeseries, Subset.GeoExperimentData.

`Subset.GeoExperimentData` *Extracts a subset of a GeoExperimentData.*

Description

Extracts a subset of a GeoExperimentData.

Usage

```
## S3 method for class 'GeoExperimentData'
Subset(obj, rows, columns, ...)
```

Arguments

`obj`	a GeoExperimentData object.
`rows`	logical vector indicating rows to keep. No missing values are allowed.
`columns`	character vector indicating columns of the data frame to keep. Columns such as 'date', 'geo', 'period', 'geo.group' and 'assignment', if they exists, are always included.
`...`	further arguments to be passed to or from other methods.

Value

A GeoExperimentData object (a subset of 'x').

Note

The arguments 'rows' and 'columns' are *not* evaluated in the environment of the columns of 'obj'; the expressions must be evaluable in the enclosing environment. This behavior is deliberately different from that of 'subset'. The original experiment configuration (periods, geo assignment, treatment assignment) is preserved.

See Also

Subset (generic), Subset.GeoTimeseries.

`Subset.GeoTimeseries` *Extract a subset of GeoTimeseries.*

Description

Extract a subset of GeoTimeseries.

Usage

```
## S3 method for class 'GeoTimeseries'
Subset(obj, rows, columns, ...)
```

Arguments

`obj`	a GeoTimeseries object.
`rows`	logical vector indicating rows to keep. No missing values are allowed.
`columns`	character vector indicating columns of the data frame to keep. Columns such as 'date', 'geo', 'period', 'geo.group' and 'assignment', if they exist, are always included.
`...`	further arguments to be passed to or from other methods.

Value

A GeoTimeseries object (a subset of 'obj').

Note

The arguments 'rows' and 'columns' are not evaluated in the environment of the columns of 'obj'; the expressions must be evaluable in the enclosing environment.

See Also

Subset (generic), Subset.GeoExperimentData.

`summary.GBRROASAnalysisFitGbr1` *Returns a concise summary of the ROAS estimate and confidence interval and the total incremental cost and incremental response.*

Description

Returns a concise summary of the ROAS estimate and confidence interval and the total incremental cost and incremental response.

Usage

```
## S3 method for class 'GBRROASAnalysisFitGbr1'
summary(object, level = 0.9,
  interval.type = c("one-sided", "two-sided"), threshold = 0, ...)
```

Arguments

`object`	a GBRROASAnalysisFitGbr1 object.
`level`	(number between 0 and 1) confidence level.
`interval.type`	(string) 'one-sided', or 'two-sided' (interval).
`threshold`	(numeric vector) threshold(s) for the right-tail posterior probabilities of beta2.
`...`	ignored.

Value

A ROASAnalysisResults object.

`summary.ROASPreanalysisFit`
Computes a summary of the predicted ROAS estimate and associated incremental cost.

Description

Computes a summary of the predicted ROAS estimate and associated incremental cost.

Usage

```
## S3 method for class 'ROASPreanalysisFit'
summary(object, level = 0.9,
  interval.type = c("one-sided", "two-sided"), precision = 1,
  cost = NA_real_, ...)
```

Arguments

`object`	a ROASPreanalysisFit object.
`level`	(number between 0 and 1) confidence level.
`interval.type`	(string) 'one-sided', 'two-sided'.
`precision`	(number) target precision of the estimate; the distance between the lower bound of the confidence interval and the point estimate. Note: if 'cost' is given, this will be ignored and the CI half-width will be computed based on 'cost'.
`cost`	(number) cost difference (ad spend difference); if missing, will be computed based on 'precision'.
`...`	ignored.

Value

A ROASPreanalysisResults object.

---

summary.TBRAnalysisFitTbr1

Returns a concise summary of the incremental response estimate and its confidence interval for the model 'tbr1'.

Description

Returns a concise summary of the incremental response estimate and its confidence interval for the model 'tbr1'.

Usage

```
## S3 method for class 'TBRAnalysisFitTbr1'
summary(object, level = 0.9,
  interval.type = c("one-sided", "two-sided"), threshold = 0, ...)
```

Arguments

`object`	a TBRAnalysisFit object.
`level`	(number between 0 and 1) confidence level.
`interval.type`	(string) 'one-sided', 'two-sided' (interval).
`threshold`	(numeric vector) threshold(s) for the right-tail posterior.
`...`	ignored.

Value

A TBRAnalysisResults object.

---

summary.TBRROASAnalysisFit

Returns a concise summary of the incremental response estimate and its confidence interval for a TBR ROAS analysis.

Description

Returns a concise summary of the incremental response estimate and its confidence interval for a TBR ROAS analysis.

Usage

```
## S3 method for class 'TBRROASAnalysisFit'
summary(object, level = 0.9,
  interval.type = c("one-sided", "two-sided"), threshold = 0, ...)
```

Arguments

`object`	a TBRROASAnalysisFit object.
`level`	(number between 0 and 1) confidence level.
`interval.type`	(string) 'one-sided', 'two-sided' (interval).
`threshold`	(numeric vector) threshold(s) for the right-tail posterior probabilities of the total cumulative iROAS.
`...`	ignored.

Value

A ROASAnalysisResults object.

`TBRAnalysisData` *Constructs a TBRAnalysisData object.*

Description

Constructs a TBRAnalysisData object.

Usage

```
TBRAnalysisData(x)
```

Arguments

`x`	a _plain_ 'data.frame' or an object that has to be coercible to a _plain_ 'data.frame' with the required columns 'data', which _must_ be a 'Date' vector; integer-valued column 'period' indicating the pre-experiment, pretest, intervention, cooldown, and posttest periods; numeric columns 'y' (response), 'x' (covariate). Optionally the data frame can have other columns, which are ignored but whose names cannot start with a dot ('.'). No missing values are allowed in the pretest and test periods (but allowed outside of either period).

Value

A TBRAnalysisData object, which is similarly a 'data.frame' with the same columns as described above.

Note

'TBRAnalysisData' is a 'data.frame', with the required columns,

- `date` a 'Date'-valued vector.
- `period` numeric indicator of various periods. 0 indicates the pretest period. NA indicates a date that has been excluded from all analyses.
- `y` numeric values of the metric of the Treatment group.
- `x` numeric values of the metric in the Control group. In addition, optionally any other user-definable columns.

The column 'date' is the primary key; it is guaranteed that no rows with duplicate geos exist and that the rows are in temporal order.

`TreatmentAssignment` *Constructs a TreatmentAssignment object.*

Description

Constructs a TreatmentAssignment object.

Usage

```
TreatmentAssignment(x)
```

Arguments

`x`	(data frame) a mapping from (period, group) to treatment assignment condition. The data frame must have the columns 'period' (integer-valued), 'geo.group' (integer-valued, positive), and 'assignment' (integer-valued, one of 0, 1, -1), Each row specifies the mapping of one (period, group) pair. No missing values are allowed.

Value

An object of class 'TreatmentAssignment'.