GeoexperimentsResearch version 1.0

Google, Inc.

2017-04-04

Contents

Geo Experiments: an introduction

What are geo experiments?

A geo experiment is a controlled online experiment, with the aim of quantifying the effects of treatments using geographical regions as experimental units.

In its simplest form, a geographical region is divided into a Control and a Treatment region; the people in the Treatment region are exposed to a certain treatment, while for those in the Control region nothing changes. The behavior (response of some given observable variable) in both these regions is observed and measured and the effect of the treatment is estimated using a statistical model.

The Control and Treatment regions are formed by aggregating smaller geographical units (called 'geos'), usually by randomization or by a matching algorithm.

Geo experiments for estimating ad effectiveness

A typical application is to investigate the effect of increased investment in online advertising. People who reside in the Treatment region are served ads at a higher intensity (spend per time unit) whenever they do a web search using prespecified keywords. Those who reside in the Control region continue to be served ads at the usual intensity. Such an experiment typically lasts for a few weeks.

The aim of a geo experiment in the online ad context is to estimate the return (in monetary terms) on the money spent: more accurately the incremental return on ad spend (iROAS), which refers to the additional return that would have not been received without the specific additional spend in ads. The "return" is usually the aggregate sales, either online or offline.

Similarly, we can estimate the effect of decreased spending. So the market intervention here may be either "increased" spending, "decreased" spending, or in general, some spend "change". In the simple one-period geo experiments, the statistical model only needs to know which geos experienced a spend change and which did not (i.e., which geos were in the Control regions).

Overview of this vignette

This vignette shows briefly how to use the R package 'GeoexperimentsResearch,'

1. to represent information as data objects;
2. to analyze geo experiment data once an experiment has finished;
3. to run a preanalysis.

For details, refer to the package manual.

For further general information on geo experiments, see [1], [2], and [3].

Attaching the package

```
library(GeoexperimentsResearch)
```

Structure of geo experiment data

For the purpose of illustration, we assume that we have run a geo experiment and wish to estimate the incremental return on ad spend.

We need to collect three pieces of information:

- The experiment periods (date ranges of the Pretest, intervention, and cooldown periods);
- The geo assignment;
- The observational data for both the response metric and the cost metric.

Experiment periods

A geo experiment consists of several, distinct time periods: the Pretest, Intervention, and Cooldown periods. The latter two periods combined make up the 'Test' period.

During the Pretest period, any ad campaigns in the Treatment and Control geos that are targeted by the experiment are in their unmodified base state. All geos operate with the same baseline level campaign settings (e.g., common bidding, keyword lists, ad targeting, etc); the difference between the Control and Treatment geos is zero in expectation.

The targeted ad campaigns are modified in the Treatment geos during the Intervention period.

Finally, these targeted ad campaigns are reset to their original state during the Cooldown period. This does not always mean their effects will cease instantly. Incremental offline sales, for example, may continue to accrue across subsequent days or even weeks. Including data from the Cooldown period in the analysis makes it possible to capture these lagged effects from the advertising change. This lagged impact may be substantial or not, depending on the advertising situation; hence it can be excluded if it is obvious in the analysis that there are no lagged effects.

By convention, we number the periods as 0 (Pretest), 1 (Intervention), 2 (Cooldown), but other numbering is allowed provided that the order of the periods is unchanged.

This information is represented by the *ExperimentPeriods* object class. The start dates of each period must be specified, and finally end date of the experiment. This example has only a pretest period and one intervention period:

```
##   Period    Name      Start        End Length
## 1      0 Pretest 2015-01-05 2015-02-15     42
## 2      1    Test 2015-02-16 2015-03-15     28
```

Geo assignment

Before the experiment is started, each of the geos is assigned either to Control or to Treatment region (*geo group*). This mapping between a geo to the geo group is called the *geo assignment*.

This information is represented by the *GeoAssignment* object class. Example:

```
##    geo geo.group
## 1    1         2
## 13   2         1
## 24   3         1
## 35   4         2
## 46   5         1
## 57   6         2
```

Observational data

The observational data consist of a response metric (such as sales) and cost metric (such as cost of ad clicks). These are provided broken by date and geo, including the Pretest, Intervention, and Cooldown periods. If the data is weekly data, the weekly aggregate should be associated with the same day of the month, for instance, each Sunday.

This information is represented by the *GeoTimeseries* object class. Example: a few rows shown from the example data set:

```
##         date geo   sales cost .weekday .weeknum .weekindex
## 1 2015-01-05   1 7227.32    0        1        1     201501
## 2 2015-01-05  10 1827.21    0        1        1     201501
## 3 2015-01-05 100   23.98    0        1        1     201501
## 4 2015-01-05  11 1501.10    0        1        1     201501
## 5 2015-01-05  12 1371.61    0        1        1     201501
## 6 2015-01-05  13 1366.81    0        1        1     201501
```

Analyzing geo experiment data using the GBR and TBR methods

Reading in observational data

Load the sample data set.

This is a plain *data.frame*, with the following columns:

```r
head(salesandcost)
```

```
##         date geo   sales cost
## 1 2015-01-05   1 7227.32    0
## 2 2015-01-05  10 1827.21    0
## 3 2015-01-05 100   23.98    0
## 4 2015-01-05  11 1501.10    0
## 5 2015-01-05  12 1371.61    0
## 6 2015-01-05  13 1366.81    0
```

This data frame has a date and a geo column, and two *metrics*, the sales and the cost of ad clicks, numeric values that are associated to each geo and date.

Next, we convert this data frame into a *GeoTimeseries* object and the integrity of the time series is automatically checked. We need to specify which columns are to be treated as metrics. This helps the class methods do certain operations automatically, such as aggregation over geos and time.

```r
obj.gts <- GeoTimeseries(salesandcost, metrics=c("sales", "cost"))
```

No errors occurred, so the overall structure of the data seems to be fine. The resulting object inherits from *data.frame*, with the same columns, augmented with some extra columns:

```r
head(obj.gts)
```

```
##         date geo   sales cost .weekday .weeknum .weekindex
## 1 2015-01-05   1 7227.32    0        1        1     201501
## 2 2015-01-05  10 1827.21    0        1        1     201501
## 3 2015-01-05 100   23.98    0        1        1     201501
## 4 2015-01-05  11 1501.10    0        1        1     201501
## 5 2015-01-05  12 1371.61    0        1        1     201501
## 6 2015-01-05  13 1366.81    0        1        1     201501
```

> The 'date' column must be in either 'Date', factor, or character format and is always coerced to Date. If the date format differs from 'yyyy-mm-dd', it is necessary to specify it as argument 'date.format'.
>
> The column 'geo' is of type character even though some geo IDs (such as DMAs) are represented as integers. Using character format, however, the structure of *GeoTimeseries* is also compatible with non-integer geos such as postal codes and administrative regions without remapping them to numbers.
>
> There is no checking of whether any of the metrics are negative.

There are some extra columns provided for convenience:

- *.weekday* column denotes the day of the week (1=Monday, 7=Sunday);
- *.weeknum* indicates the number of the week within a year;
- *.weekindex* indicates a unique week number.

> The data frame can have any number of other columns, although the built-in methods recognize only 'date', 'geo', '.weekday', '.weeknum', and '.weekindex', and those registered as metrics.

Exploratory data analysis

To quickly investigate the distribution of the metrics across weeks, we can use the *aggregate* method as follows:

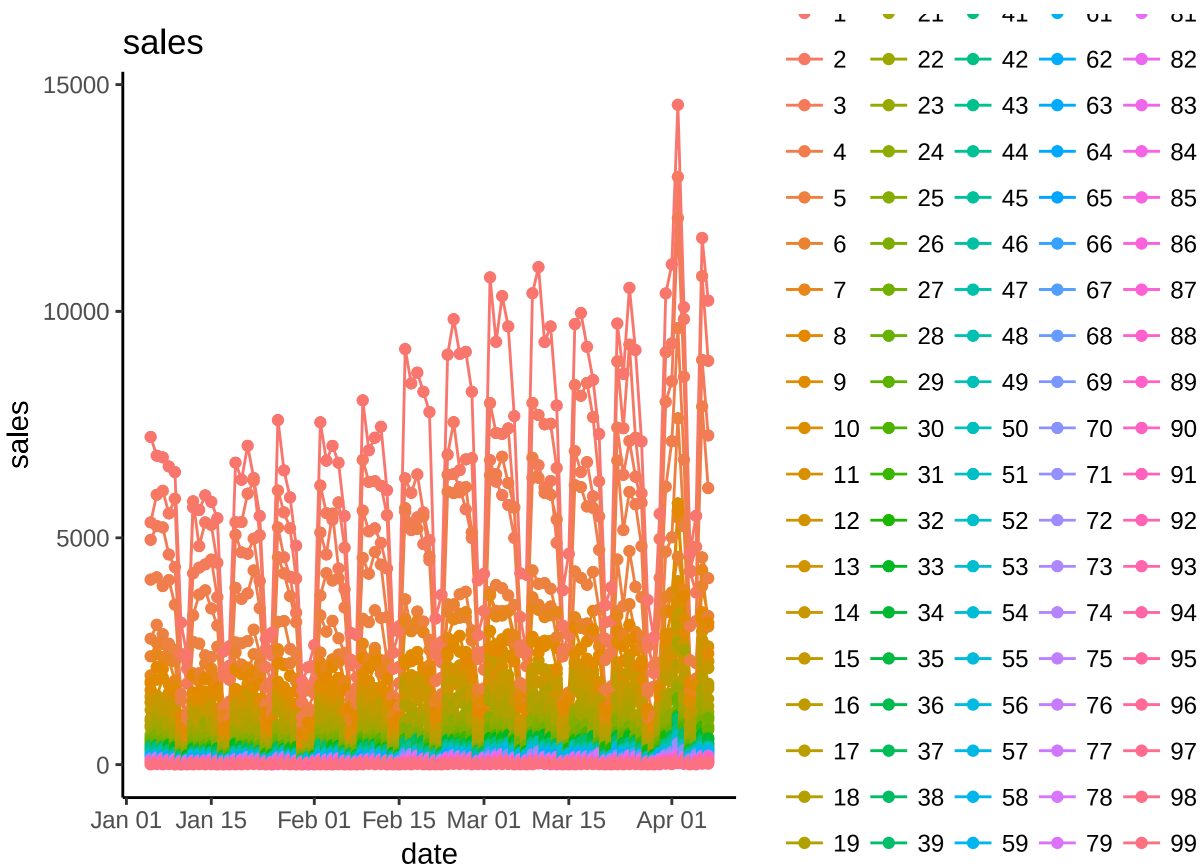
```r
aggregate(obj.gts, by='.weekindex')
```

```
##    .weekindex    sales     cost
## 1      201501 348104.2     0.00
## 2      201502 309470.3     0.00
## 3      201503 338153.1     0.00
## 4      201504 285590.4     0.00
## 5      201505 346531.2     0.00
## 6      201506 369843.2     0.00
## 7      201507 412418.4 11155.30
## 8      201508 465114.2 12468.26
## 9      201509 484845.6 13205.26
## 10     201510 490903.6 13171.18
## 11     201511 477443.1     0.00
## 12     201512 463243.3     0.00
## 13     201513 602514.1     0.00
## 14     201514 201850.0     0.00
```

We can see that normally the ad campaigns were turned off, and starting from week 6, the ad spend increased until it was turned off again on week 14.

To plot the time series, use the plot method:

```r
plot(obj.gts)
```



To hide the legend, add `legend=FALSE`. To plot the time series on log scale, add `log.scale=TRUE`. For more information of the method, type `?plot.GeoTimeseries` at the R prompt.

Experiment Periods

We specify the start of the Pretest period, the start of the test period, and the end of the experiment. If there is a Cooldown period after the actual market intervention, it must be included as a separate period (four dates in total).

```
obj.per <- ExperimentPeriods(c("2015-01-05", "2015-02-16", "2015-03-15"))
obj.per
```

```
##   Period    Name      Start        End Length
## 1      0 Pretest 2015-01-05 2015-02-15     42
## 2      1    Test 2015-02-16 2015-03-15     28
```

To introduce a cooldown period, we would specify one more date.

To learn more about the function, type `?ExperimentPeriods` at the R prompt.

Geo Assignment

We'll use the built-in sample geo assignment:

```
data(geoassignment)
head(geoassignment)
```

```
##    geo geo.group
## 1    1         2
## 13   2         1
## 24   3         1
## 35   4         2
## 46   5         1
## 57   6         2
```

From this data frame we create a *GeoAssignment* object and automatically verify its integrity:

```
obj.ga <- GeoAssignment(geoassignment)
head(obj.ga)
```

```
##    geo geo.group
## 1    1         2
## 13   2         1
## 24   3         1
## 35   4         2
## 46   5         1
## 57   6         2
```

Combining all information about the experiment into one object

The class *GeoExperimentData* combines these three pieces of information (geo time series, periods, geo assignment) into one object:

```
  obj <- GeoExperimentData(obj.gts,
                           periods=obj.per,
                           geo.assignment=obj.ga)
  head(obj)
```

```
##         date geo period geo.group assignment   sales cost .weekday .weeknum .weekindex
## 1 2015-01-05   1      0         2         NA 7227.32    0        1        1     201501
```

```
## 2 2015-01-05   10       0          1          NA 1827.21    0         1         1      201501
## 3 2015-01-05  100       0          1          NA   23.98    0         1         1      201501
## 4 2015-01-05   11       0          1          NA 1501.10    0         1         1      201501
## 5 2015-01-05   12       0          2          NA 1371.61    0         1         1      201501
## 6 2015-01-05   13       0          1          NA 1366.81    0         1         1      201501
```

The column *period* contains the indicator for the experiment periods: 0 = Pretest, 1 = test (Intervention). 'NA' marks a date that is outside of the designated experiment periods.

The column *geo.group* contains the geo group ID for each of the geos.

The column *assignment* is not used in this version of the R package. It is set to *NA* by default. It can be ignored.

Exploratory data analysis

To check how the revenue and cost metrics are distributed across periods and groups, we make use of the aggregate method again:

```
aggregate(obj, by=c('period', 'geo.group'))
```

```
##   period geo.group     sales  cost
## 1      0         1 1007853.7     0
## 2      1         1  859005.3     0
## 3      0         2  989838.8     0
## 4      1         2  994276.4 50000
```

Geo-Based Regression (GBR) Analysis

The object ('obj') that we constructed contains now all information for applying a geo experiment analysis methodology.

To perform a GBR (geo-based regression) analysis, apply method *DoGBRROASAnalysis*, specifying which of the metrics is the response and which represents the cost, along with the experiment periods and group numbers.

```
  result <- DoGBRROASAnalysis(obj, response='sales', cost='cost',
                              pretest.period=0,
                              intervention.period=1,
                              cooldown.period=NULL,
                              control.group=1,
                              treatment.group=2)
  result
```

```
##       estimate precision   lower upper level incr.resp incr.cost thres prob model
## iROAS 3.233443 0.3501623 2.88328   Inf   0.9  161672.1     50000     0    1  gbr1
```

Note that in this particular case, there is no `cooldown.period`, hence it is set to `NULL`. If there was one, we would specify the period number (for example, `cooldown.period=2`).

The resulting object (a *GBRROASAnalysisFit* object) contains the model fit: when printed, it shows its summary, which defaults to 90 percent credible intervals. To recalculate the interval with a different credibility level, we can specify this in the function call:

```
  summary(result, level=0.95, interval.type="two-sided")
```

```
##       estimate precision    lower    upper level incr.resp incr.cost thres prob model
## iROAS 3.233443 0.5385984 2.694844 3.772041  0.95  161672.1     50000     0    1  gbr1
```

To obtain the posterior probability that the true iROAS is larger than some threshold, say 3.0, we use the `summary` method as follows:

```
summary(result, threshold=3.0)
```

```
##       estimate precision   lower upper level incr.resp incr.cost thres  prob model
## iROAS 3.233443 0.3501623 2.88328   Inf   0.9  161672.1     50000     3 0.804  gbr1
```

The default threshold is 0.

Time-Based Regression (TBR) ROAS Analysis

The GeoExperimentData object can also be used for performing a TBR analysis [3], applying method *DoTBRROASAnalysis*, specifying which of the metrics is the response and which represents the cost, along with the experiment period and group numbers. The model ID is also required; currently the only available model is 'tbr1', as described in [3].

```
obj.tbr.roas <- DoTBRROASAnalysis(obj, response='sales', cost='cost',
                                  model='tbr1',
                                  pretest.period=0,
                                  intervention.period=1,
                                  cooldown.period=NULL,
                                  control.group=1,
                                  treatment.group=2)
obj.tbr.roas
```

```
##       estimate precision    lower upper level incr.resp incr.cost thres prob model
## iROAS 2.946742  0.120548 2.826194   Inf   0.9  147337.1     50000     0    1  tbr1
```

The resulting object (a *TBRROASAnalysisFit* object) contains the model fit: when printed, it shows its summary, which defaults to 90 percent one-sided credible intervals. Similarly to what we did with a *GBRROASAnalysisFit* object we can recalculate the credible interval, and the probability of exceeding a given threshold like so:

```
summary(obj.tbr.roas, level=0.95, interval.type="two-sided")
```

```
##       estimate precision    lower    upper level incr.resp incr.cost thres prob model
## iROAS 2.946742 0.1869702 2.759772 3.133713  0.95  147337.1     50000     0    1  tbr1
```

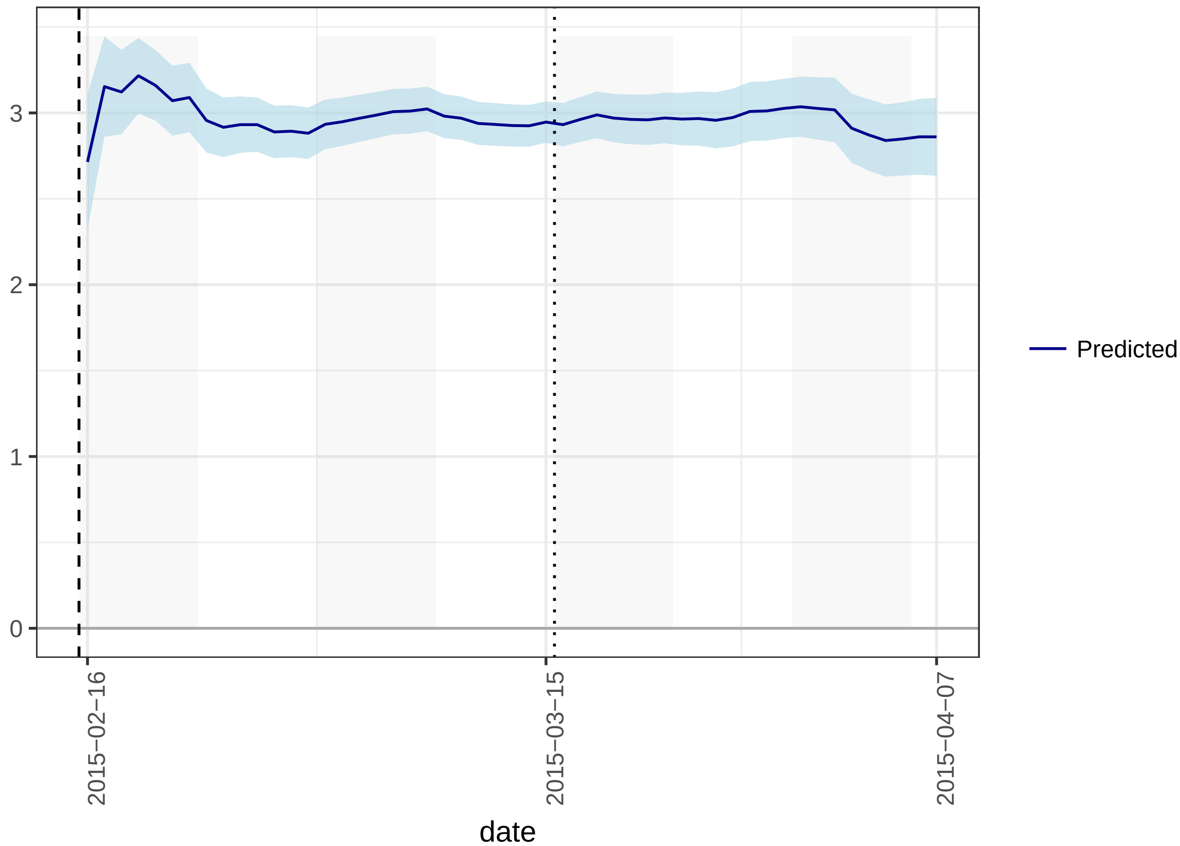
```
summary(obj.tbr.roas, threshold=3.0)
```

```
##       estimate precision    lower upper level incr.resp incr.cost thres  prob model
## iROAS 2.946742  0.120548 2.826194   Inf   0.9  147337.1     50000     3 0.284  tbr1
```

The *plot* method shows the evolution of the iROAS estimate across the Test period:

```
plot(obj.tbr.roas)
```

```
## Warning: Removed 51 rows containing missing values (geom_vline).
```



For more information on the method, type `?plot.TBRROASAnalysisFit` at the R prompt.

Time-Based Regression (TBR) Causal Effect Analysis

Unlike the TBR ROAS Analysis, which estimates the ratio of the incremental response and incremental cost, the TBR Causal Effect Analysis applies only to one single variable, such as revenue.

```
obj.tbr <- DoTBRAnalysis(obj, response='sales',
                         model='tbr1',
                         pretest.period=0,
                         intervention.period=1,
                         cooldown.period=NULL,
                         control.group=1,
                         treatment.group=2)
```

The resulting object (a *TBRAnalysisFitTbr1* object) contains the model fit for each time point, which can be seen when printed. To show the summary of the effect, we use the *summary* method:

```
summary(obj.tbr)
```

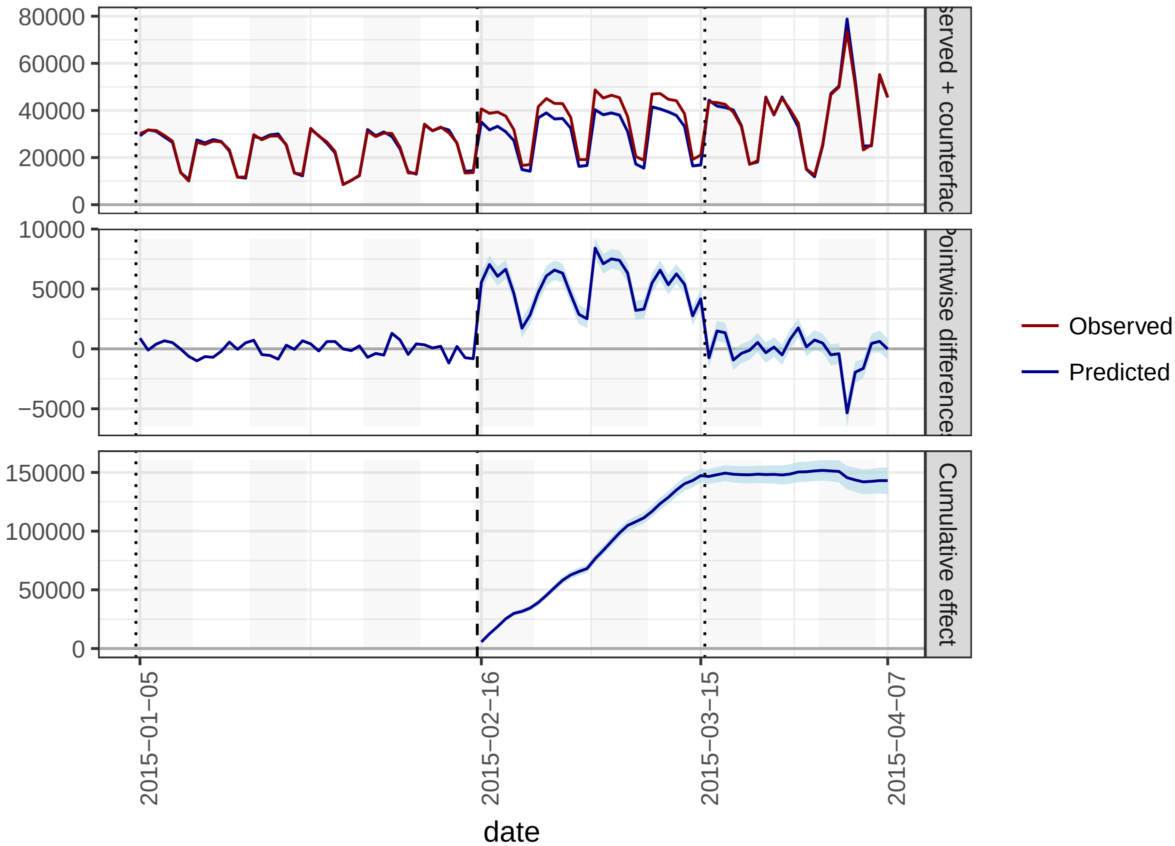
```
##             estimate precision    lower upper       se level thres prob model
## incremental 147337.1  6027.401 141309.7   Inf 4625.514   0.9     0    1  tbr1
```

which defaults to the 90% one-sided interval.

The *plot* method illustrates the results of the analysis.

```
plot(obj.tbr)
```

```
## Warning: Removed 186 rows containing missing values (geom_path).
```



For more information on the method, type `?plot.TBRAnalysisFitTbr1` at the R prompt.

Preanalysis

Before running an experiment, we need to understand how the design parameters affect the uncertainty of the iROAS estimate. One of the most important parameters is the ad spend change, which affects the estimate uncertainty directly: doubling the ad spend halves the width of the 2-sided confidence interval (in terms of one-sided intervals, this is the distance from the lower bound and the point estimate). We refer to this confidence interval half-width by *precision* (which gets better as the confidence interval gets shorter).

The function *DoROASPreanalysis* predicts the precision of the iROAS estimate based on historical data provided. It simulates experiments (by resampling) with given period lengths and records the precision from each simulated experiment. We can then use the *summary* method to compute the precision given an ad spend change, or find the ad spend change associated with a given precision.

For each simulated geo experiment data set, ROAS and its precision is estimated. The process yields a distribution of the these estimates of precision. The summary method takes the empirical median as the point estimate. If the data set does not have strong seasonalities, the variation of this estimate should be fairly small.

The process runs as follows:

1. Assign geos to treatment groups.
2. Run preanalysis to predict the precision.

A randomized geo assignment

Randomized geo assignments can be done using 'GeoStrata' objects. This object includes a mapping from each geo to a *stratum* (or block), so a stratified randomization can be performed. This can be generated automatically using the *ExtractGeoStrata* function:

```
obj.geo.strata <- ExtractGeoStrata(obj.gts, volume="sales", n.groups=2)
head(obj.geo.strata)
```

```
##   geo stratum geo.group proportion   volume    sales     cost
## 1   1       1        NA 0.11180683 44690.98 44690.98 774.9707
## 2   2       1        NA 0.09195421 36755.57 36755.57   0.0000
## 3   3       2        NA 0.07783086 31110.24 31110.24   0.0000
## 4   4       2        NA 0.06877094 27488.85 27488.85 485.7571
## 5   5       3        NA 0.04748058 18978.75 18978.75   0.0000
## 6   6       3        NA 0.03869424 15466.71 15466.71 272.3836
```

The argument 'volume' specifies the name of the metric that is used for stratification: the geos are sorted by their volume and divided into strata of 2 each.

To generate a randomized geo assignment, we use the 'Randomize' method:

```
obj.geo.assignment <- Randomize(obj.geo.strata)
head(obj.geo.assignment)
```

```
##   geo stratum geo.group proportion   volume    sales     cost
## 1   1       1         2 0.11180683 44690.98 44690.98 774.9707
## 2   2       1         1 0.09195421 36755.57 36755.57   0.0000
## 3   3       2         1 0.07783086 31110.24 31110.24   0.0000
## 4   4       2         2 0.06877094 27488.85 27488.85 485.7571
## 5   5       3         1 0.04748058 18978.75 18978.75   0.0000
## 6   6       3         2 0.03869424 15466.71 15466.71 272.3836
```

Predicting the precision

We pass this object to the method *DoGBRPreanalysis* along with the GeoTimeseries, the length of the Pretest, Intervention, and Cooldown periods, and specify a metric:

```
obj.pre <-  DoROASPreanalysis(obj.gts, response="sales",
                              geos=obj.geo.assignment,
                              prop.to="sales",
                              period.lengths=c(42, 21, 7))
```

The resulting object 'obj.pre' is of class *ROASPreanalysisFit*, which only contains the raw simulated numbers. To compute the required spend for precision +/- 1.0, we call the *summary* method:

```
  results <- summary(obj.pre,
               level=0.90,
               type="one-sided",
               precision=1.0)
  print(results)
```

```
##   model precision total.cost level  interval cfrac gratio n.geos pretest test cooldown fixed
## 1  gbr1         1  20302.235   0.9 one-sided   0.5    1:1    100      42   21        7  TRUE
## 2  tbr1         1   9496.645   0.9 one-sided   0.5    1:1    100      42   21        7  TRUE
```

This function takes the median of the simulated precisions as the point estimate.

For convenience, applying the 'print' method on the "GBRPreanalysisFit" prints the default summary (one-sided confidence interval at level 0.90, precision 1.0):

```
print(obj.pre)
```

```
##   model precision total.cost level  interval cfrac gratio n.geos pretest test cooldown fixed
## 1  gbr1         1  20302.235   0.9 one-sided   0.5    1:1    100      42   21        7  TRUE
## 2  tbr1         1   9496.645   0.9 one-sided   0.5    1:1    100      42   21        7  TRUE
```

The function can be also used to predict the precision given a (total) spend change over the test period:

```
results2 <- summary(obj.pre,
              level=0.90,
              type="one-sided",
              cost=10000)
print(results2)
```

```
##   model precision total.cost level  interval cfrac gratio n.geos pretest test cooldown fixed
## 1  gbr1 2.0302235      10000   0.9 one-sided   0.5    1:1    100      42   21        7  TRUE
## 2  tbr1 0.9496645      10000   0.9 one-sided   0.5    1:1    100      42   21        7  TRUE
```

The results apply to the given geo assignment only; for a different geo assignment, the results are likely to be different.

References

[1] Kerman, J., Vaver, J. and Koehler, J. (2011) Estimating causal effects using geo experiments

[2] Vaver, J. and Koehler, J. (2011) Measuring Ad Effectiveness Using Geo Experiments

[3] Kerman, J. and Wang, P., and Vaver, J. (2017) Estimating Ad Effectiveness using Geo Experiments in a Time-Based Regression Framework

Disclaimer