**Package ‘promotionImpact’**

June 5, 2019

**Type** Package

**Title** Analysis & Measurement of Promotion Effectiveness

**Version** 0.1.2

**Date** 2019-05-16

**Description** Analysis and measurement of promotion effectiveness on a given target variable (e.g. daily sales). After converting promotion schedule into dummy or smoothed predictor variables, the package estimates the effects of these variables controlled for trend/perIODICITY/structural change using prophet by Taylor and Letham (2017) <doi:10.7287/peerj.preprints.3190v2> and some prespecified variables (e.g. start of a month).

**Depends** R (>= 3.5.0), Rcpp (>= 0.12.17), dplyr (>= 0.7.6), ggplot2 (>= 3.0.0), scales (>= 1.0.0)

**Imports** KernSmooth (>= 2.23.15), data.table (>= 1.11.4), ggpubr (>= 0.1.8), reshape2 (>= 1.4.3), stringr (>= 1.3.1), strucchange (>= 1.5.1), lmtest (>= 0.9), crayon (>= 1.3.4), prophet (>= 0.3.0.1)

**License** BSD_3_clause + file LICENSE

**URL** https://github.com/ncsoft/promotionImpact

**LazyData** true

**RoxygenNote** 6.1.1

**Encoding** UTF-8

**Author** Nahyun Kim [cre, aut], Hyemin Um [aut], Eunjo Lee [aut], NCSoft Corporation [cph]

**Maintainer** Nahyun Kim <nhkim1302@ncsoft.com>

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2019-06-05 12:40:09 UTC
**R topics documented:**

- `compareModels` ................................................. 2
- `detectOutliers` ............................................. 3
- `promotionImpact` ........................................... 4
- `sim.data` ..................................................... 6
- `sim.promotion` .............................................. 6
- `sim.promotion.sales` ..................................... 7

**Description**

`compareModels`

**Usage**

```r
compareModels(data, promotion, fix = list(logged = TRUE, differencing = TRUE), time.field = "dt", target.field = "sales",
  dummy.field = NULL, trend.param = 0.05, period.param = 3,
  var.type = "smooth", smooth.except.date = NULL,
  smooth.bandwidth = 2, smooth.var.sum = TRUE, allow.missing = TRUE)
```

**Arguments**

- `data` Dataframe containing date, target variable, and some additional time dummies that the researcher wants to account for.
- `promotion` Dataframe containing promotion ID, start date, end date, promotion tag(type). Might include daily payments associated with the promotion.
- `fix` A List of constraints to find the best model. Constraints can only be in following list: `period`,`trend`,`logged`,`synergy.var`,`differencing`,`smooth.origin`,`structural.change`,`synergy.promotion`
- `time.field` Specify the date field of `data`.
- `target.field` Specify the date field of `data`.
- `dummy.field` Specify the additional time dummies of `data`.
- `trend.param` Flexibility of trend component. Default is 0.05, and as this value becomes larger, the trend component will be more flexible.
- `period.param` Flexibility of period component. Default is 3, and as this value becomes larger, the period component will be more flexible.
- `var.type` 'smooth' to use smoothed promotion variables, 'dummy' to use dummy promotion variables
- `smooth.except.date` Date value that will be excluded from the smoothing process. eg) '01' to exclude every start day of a month
**detectOutliers**

```r
smooth.bandwidth
Bandwidth of local polynomial regression used in the smoothing process. Default value is 2.

smooth.var.sum
If TRUE, the smoothing values for times when multiple promotions in a single tag overlap will be the values from the latest promotion. Otherwise, the values will be added (default).

allow.missing
TRUE to allow missing data in promotion sales during the promotion period
```

**Details**

`compareModels` compares several models under user-defined conditions and suggests the best options.

**Examples**

```r
comparison <- compareModels(data = sim.data, promotion = sim.promotion.sales,
   fix = list(logged = T, differencing = T, smooth.origin='all',
      trend = FALSE, period = NULL),
   time.field = 'dt', target.field = 'simulated_sales',
   trend.param = 0.02, period.param = 2)
```

---

**detectOutliers**

```r
detectOutliers

detect some outliers
```

**Description**

detectOutliers

**Usage**

detectOutliers(model, threshold = list(cooks.distance = 1, dfbetas = 1,
   dffits = 2), option = 2)

**Arguments**

- `model` : Execution result object : `promotionImpact`
- `threshold` : List of threshold values to be determined as outliers if greater than the written values
- `option` : The number of indicators that must be greater than the threshold values to be outliers.

**Details**

detectOutliers extracts outliers which affect the average effects of promotions.
promotionImpact

Examples

```r
pr1 <- promotionImpact(data=sim.data, promotion=sim.promotion,
    time.field = 'dt', target.field = 'simulated_sales',
    trend = FALSE, period = NULL, structural.change = FALSE,
    logged = TRUE, differencing = TRUE, synergy.promotion = FALSE,
    synergy.var = NULL, allow.missing = TRUE)
out <- detectOutliers(model = pr1,
    threshold = list(cooks.distance=1, dfbetas=1, dffits=2), option = 1)
```

**promotionImpact**  
*estimate effectiveness of promotions*

**Description**

**promotionImpact**

**Usage**

```r
promotionImpact(data, promotion, time.field = "date",
    target.field = "value", dummy.field = NULL, trend = TRUE,
    period = "auto", structural.change = FALSE, trend.param = 0.05,
    period.param = 3, var.type = "smooth", smooth.except.date = NULL,
    smooth.bandwidth = 2, smooth.origin = "all", smooth.var.sum = TRUE,
    logged = TRUE, differencing = TRUE, synergy.promotion = FALSE,
    synergy.var = NULL, allow.missing = TRUE)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>Dataframe containing date, target variable, and some additional time dummies that the researcher wants to account for.</td>
</tr>
<tr>
<td>promotion</td>
<td>Dataframe containing promotion ID, start date, end date, promotion tag(type). Might include daily payments associated with the promotion.</td>
</tr>
<tr>
<td>time.field</td>
<td>Specify the date field of 'data'.</td>
</tr>
<tr>
<td>target.field</td>
<td>Specify the target field of 'data'.</td>
</tr>
<tr>
<td>dummy.field</td>
<td>Specify the additional time dummies of 'data'.</td>
</tr>
<tr>
<td>trend</td>
<td>TRUE to incorporate trend component, FALSE to exclude the trend component.</td>
</tr>
<tr>
<td>period</td>
<td>NULL to exclude any periodicity from the model, 'auto' to automatically determine the period, certain numeric value(e.g. '30.5' for month) to manually specify the period</td>
</tr>
<tr>
<td>structural.change</td>
<td>TRUE to incorporate structural changes in the intercept(baseline)</td>
</tr>
</tbody>
</table>
**promotionImpact**

`trend.param`  Flexibility of trend component. Default is 0.05, and as this value becomes larger, the trend component will be more flexible.

`period.param` Flexibility of period component. Default is 3, and as this value becomes larger, the period component will be more flexible.

`var.type` ‘smooth’ to use smoothed promotion variables, ‘dummy’ to use dummy promotion variables

`smooth.except.date` Date value that will be excluded from the smoothing process. eg) ’01’ to exclude every start day of a month

`smooth.bandwidth` Bandwidth of local polynomial regression used in the smoothing process. Default value is 2.

`smooth.origin` ‘all’ to estimate a global smoothing function for all promotions. ‘tag’ to estimate different smoothing functions for different promotion types(tags).

`smooth.var.sum` If TRUE, the smoothing values for times when multiple promotions in a single tag overlap will be the values from the latest promotion. Otherwise, the values will be added(default).

`logged` TRUE to take logs to the target variable and the trend/period component

`differencing` TRUE to first difference the target variable, smoothed regressors, and the trend/period component values

`synergy.promotion` TRUE to incorporate synergy between promotion tags.

`synergy.var` Specify the synergy variables. ’names of fields’ between each promotion tag and other variables. eg) c(’month_start’) to incorporate synergy between each promotion tag and ’month_start’.

`allow.missing` TRUE to allow missing data in promotion sales during the promotion period

**Details**

promotionImpact is for analysis & measurement of the effectiveness of promotions, controlling for some prespecified or estimated control variables.

**Examples**

```r
pr1 <- promotionImpact(data=sim.data, promotion=sim.promotion,
  time.field = 'dt', target.field = 'simulated_sales',
  trend = FALSE, period = NULL, structural.change = FALSE,
  logged = TRUE, differencing = TRUE, synergy.promotion = FALSE,
  synergy.var = NULL, allow.missing = TRUE)
```
### sim.data  

**Daily Total Sales**

<table>
<thead>
<tr>
<th>sim.data</th>
<th>Daily Total Sales</th>
</tr>
</thead>
</table>

#### Description
This data set is simulated daily total sales data containing 958 observations of 2 variables. `dt`: date with Date format. `simulated_sales`: simulated daily sales with numeric format.

#### Usage
sim.data

#### Format
A dataset containing 958 observations of 2 variables.

#### Source
NCsoft AnalysisModeling Team <gimmesilver@ncsoft.com> <windy0126@ncsoft.com> <nhkim1302@ncsoft.com>

### sim.promotion  

**Promotion Schedule**

<table>
<thead>
<tr>
<th>sim.promotion</th>
<th>Promotion Schedule</th>
</tr>
</thead>
</table>

#### Description
This data set is promotion schedule data including promotion tag information. `pro_id`: promotion ID. `start_dt`: start date of each promotion. `end_dt`: end date of each promotion. `tag_info`: promotion tag information (promotion type).

#### Usage
sim.promotion

#### Format
A dataset containing 50 observations of 4 variables.

#### Source
NCsoft AnalysisModeling Team <gimmesilver@ncsoft.com> <windy0126@ncsoft.com> <nhkim1302@ncsoft.com>
sim.promotion.sales  Daily Promotion Sales with Promotion information

Description
This data set is simulated daily promotion sales data with promotion information. ‘pro_id’: promotion ID ‘start_dt’: start date of each promotion ‘end_dt’: end date of each promotion ‘tag_info’: promotion tag information (promotion type) ‘dt’: date ‘payment’: simulated daily promotion sales

Usage
sim.promotion.sales

Format
A dataset containing 1486 observations of 6 variables.

Source
NCsoft AnalysisModeling Team <gimmesilver@ncsoft.com> <windy0126@ncsoft.com> <nhkim1302@ncsoft.com>
Index

* Topic **datasets**
  - sim.data, 6
  - sim.promotion, 6
  - sim.promotion.sales, 7

compareModels, 2

detectOutliers, 3

promotionImpact, 4

sim.data, 6
sim.promotion, 6
sim.promotion.sales, 7