Package 'MarketMatching'

July 3, 2019

Type Package

Title Market Matching and Causal Impact Inference

Version 1.1.2

Date 2019-06-24

Description For a given test market find the best control markets using time series matching and analyze the impact of an intervention. The intervention could be be a marketing event or some other local business tactic that is being tested. The workflow implemented in the Market Matching package utilizes dynamic time warping (the 'dtw' package) to do the matching and the 'CausalImpact' package to analyze the causal impact. In fact, this package can be considered a ``workflow wrapper'' for those two packages.

Depends R (>= 3.5.0)

License GPL (>= 3)

Imports data.table, ggplot2, dplyr, utils, iterators, doParallel, parallel, foreach, reshape2, CausalImpact, zoo, bsts, scales, dtw

LazyData true

VignetteBuilder knitr

Suggests knitr, rmarkdown

RoxygenNote 6.1.1

NeedsCompilation no

Author Larsen Kim [aut, cre]

Maintainer Larsen Kim <kblarsen4@gmail.com>

Repository CRAN

Date/Publication 2019-07-03 17:10:03 UTC

R topics documented:

`best_matches` *For each market, find the best matching control market*

Description

`best_matches` finds the best matching control markets for each market in the dataset using dynamic time warping (`dtw` package). The algorithm simply loops through all viable candidates for each market in a parallel fashion, and then ranks by distance and/or correlation.

Usage

```
best_matches(data=NULL,
             markets_to_be_matched=NULL,
             id_variable=NULL,
             date_variable=NULL,
             matching_variable=NULL,
             parallel=TRUE,
             warping_limit=1,
             start_match_period=NULL,
             end_match_period=NULL,
             matches=5,
             dtw_emphasis=1)
```

Arguments

`data` input data.frame for analysis. The dataset should be structured as "stacked" time series (i.e., a panel dataset). In other words, markets are rows and not columns – we have a unique row for each area/time combination.

`markets_to_be_matched` Use this parameter if you only want to get control matches for a subset of markets or a single market The default is NULL which means that all markets will be paired with matching markets

`id_variable` the name of the variable that identifies the markets

`date_variable` the time stamp variable

`matching_variable` the variable (metric) used to match the markets. For example, this could be sales or new customers

`parallel` set to TRUE for parallel processing. Default is TRUE

`warping_limit` the warping limit used for matching. Default is 1, which means that a single query value can be mapped to at most 2 reference values.

`start_match_period` the start date of the matching period (pre period). Must be a character of format "YYYY-MM-DD" – e.g., "2015-01-01"

`end_match_period`
: the end date of the matching period (pre period). Must be a character of format "YYYY-MM-DD" – e.g., "2015-10-01"

`matches`
: Number of matching markets to keep in the output (to use less markets for inference, use the control_matches parameter when calling inference)

`dtw_emphasis`
: Number from 0 to 1. The amount of emphasis placed on dtw distances, versus correlation, when ranking markets. Default is 1 (all emphasis on dtw). If emphasis is set to 0, all emphasis would be put on correlation. An emphasis of 0.5 would yield equal weighting.

Value

Returns an object of type `market_matching`. The object has the following elements:

`BestMatches`
: A data.frame that contains the best matches for each market in the input dataset

`Data`
: The raw data used to do the matching

`MarketID`
: The name of the market identifier

`MatchingMetric`
: The name of the matching variable

`DateVariable`
: The name of the date variable

Examples

```
##-----------------------------------------------------------------------
## Find the best matches for the CPH airport time series
##-----------------------------------------------------------------------
library(MarketMatching)
data(weather, package="MarketMatching")
mm <- best_matches(data=weather,
                   id="Area",
                   markets_to_be_matched=c("CPH", "SFO"),
                   date_variable="Date",
                   matching_variable="Mean_TemperatureF",
                   parallel=FALSE,
                   start_match_period="2014-01-01",
                   end_match_period="2014-10-01")
head(mm$BestMatches)
```

`inference` *Given a test market, analyze the impact of an intervention*

Description

`inference` Analyzes the causal impact of an intervention using the CausalImpact package, given a test market and a matched_market object from the best_matches function. The function returns an object of type "market_inference" which contains the estimated impact of the intervention (absolute and relative).

Usage

```
inference(matched_markets=NULL,
          bsts_modelargs=NULL,
          test_market=NULL,
          end_post_period=NULL,
          alpha=0.05,
          prior_level_sd=0.01,
          control_matches=5,
          analyze_betas=FALSE,
          nseasons=NULL)
```

Arguments

`matched_markets`
: A matched_market object created by the market_matching function

`bsts_modelargs`
: A list() that passes model parameters directly to bsts – such as list(niter = 1000, nseasons = 52, prior.level.sd=0.1) This parameter will overwrite the values specified in prior_level_sd and nseasons. ONLY use this if you're using intricate bsts settings For most use-cases, using the prior_level_sd and nseasons parameters should be sufficient

`test_market`
: The name of the test market (character)

`end_post_period`
: The end date of the post period. Must be a character of format "YYYY-MM-DD" – e.g., "2015-11-01"

`alpha`
: Desired tail-area probability for posterior intervals. For example, 0.05 yields 0.95 intervals

`prior_level_sd`
: Prior SD for the local level term (Gaussian random walk). Default is 0.01. The bigger this number is, the more wiggliness is allowed for the local level term. Note that more wiggly local level terms also translate into larger posterior intervals This parameter will be overwritten if you're using the bsts_modelargs parameter

`control_matches`
: Number of matching control markets to use in the analysis (default is 5)

`analyze_betas`
: Controls whether to test the model under a variety of different values for prior_level_sd.

`nseasons`
: Seasonality for the bsts model – e.g., 52 for weekly seasonality

Value

Returns an object of type `inference`. The object has the following elements:

`AbsoluteEffect`
: The estimated absolute effect of the intervention

`AbsoluteEffectLower`
: The lower limit of the estimated absolute effect of the intervention. This is based on the posterior interval of the counterfactual predictions. The width of the interval is determined by the `alpha` parameter.

`AbsoluteEffectUpper`
: The upper limit of the estimated absolute effect of the intervention. This is based on the posterior interval of the counterfactual predictions. The width of the interval is determined by the `alpha` parameter.

`RelativeEffectLower`
: Same as the above, just for relative (percentage) effects

`RelativeEffectUpper`
: Same as the above, just for relative (percentage) effects

`TailProb` Posterior probability of a non-zero effect

`PrePeriodMAPE` Pre-intervention period MAPE

`DW` Durbin-Watson statistic. Should be close to 2.

`PlotActualVersusExpected`
: Plot of actual versus expected using `ggplot2`

`PlotCumulativeEffect`
: Plot of the cumulative effect using `ggplot2`

`PlotPointEffect`
: Plot of the pointwise effect using `ggplot2`

`PlotActuals` Plot of the actual values for the test and control markets using `ggplot2`

`PlotPriorLevelSdAnalysis`
: Plot of DW and MAPE for different values of the local level SE using `ggplot2`

`PlotLocalLevel` Plot of the local level term using `ggplot2`

`TestData` A `data.frame` with the test market data

`ControlData` A `data.frame` with the data for the control markets

`PlotResiduals` Plot of the residuals using `ggplot2`

`TestName` The name of the test market

`TestName` The name of the control market

`zooData` A `zoo` time series object with the test and control data

`Predictions` Actual versus predicted values

`CausalImpactObject`
: The CausalImpact object created

`Coefficients` The average posterior coefficients

Examples

```
library(MarketMatching)
##-----------------------------------------------------------------------
## Analyze causal impact of a made-up weather intervention in Copenhagen
## Since this is weather data it is a not a very meaningful example.
## This is merely to demonstrate the function.
##-----------------------------------------------------------------------
data(weather, package="MarketMatching")
mm <- best_matches(data=weather,
                   id="Area",
                   markets_to_be_matched=c("CPH", "SFO"),
```

```
                   date_variable="Date",
                   matching_variable="Mean_TemperatureF",
                   parallel=FALSE,
                   warping_limit=1, # warping limit=1
                   dtw_emphasis=1, # rely only on dtw for pre-screening
                   matches=5, # request 5 matches
                   start_match_period="2014-01-01",
                   end_match_period="2014-10-01")
library(CausalImpact)
results <- inference(matched_markets=mm,
                     test_market="CPH",
                     analyze_betas=FALSE,
                     control_matches=5, # use all 5 matches for inference
                     end_post_period="2015-12-15",
                     prior_level_sd=0.002)
```

`MarketMatching` *Market Matching and Causal Impact Inference*

Description

For a given test market find the best matching control markets using time series matching and analyze the impact of an intervention. The intervention could be be a marketing event or some other local business tactic that is being tested. The package utilizes dynamic time warping to do the matching and the CausalImpact package to analyze the causal impact. In fact, MarketMatching is simply a wrapper and worfflow for those two packages. MarketMatching does not provide any functionality that cannot be found in these packages but simplifies the workflow of using dtw and CausalImpact together and provides charts and data that are easy to manipulate.

Details

The MarketMatching package can be used to perform the following analyses:

- For all markets in the input dataset, find the best control markets using time series matching.

- Given a test market and a matching control market (from above), analyze the causal impact of an intervention

The package utilizes the dtw package in CRAN to do the time series matching, and the CausalImpact package to do the inference. (Created by Kay Brodersen at Google). For more information about the CausualImpact package, see the following reference:

CausalImpact version 1.0.3, Brodersen et al., Annals of Applied Statistics (2015). http://google.github.io/CausalImpact/

The MarketMatching has two separate functions to perform the tasks described above:

- best_matches(): This function finds the best matching control markets for all markets in the input dataset.

- inference(): Given an object from best_matches(), this function analyzes the causal impact of an intervention.

For more details, check out the vignette: browseVignettes("MarketMatching")

Author(s)

Kim Larsen (kblarsen4 at gmail.com)

Examples

```
##-----------------------------------------------------------------------
## Find best matches for CPH
## If we leave test_market as NULL, best matches are found for all markets
##-----------------------------------------------------------------------
library(MarketMatching)
data(weather, package="MarketMatching")
mm <- best_matches(data=weather,
                   id="Area",
                   date_variable="Date",
                   matching_variable="Mean_TemperatureF",
                   parallel=FALSE,
                   markets_to_be_matched="CPH",
                   warping_limit=1, # warping limit=1
                   dtw_emphasis=1, # rely only on dtw for pre-screening
                   matches=5, # request 5 matches
                   start_match_period="2014-01-01",
                   end_match_period="2014-10-01")
head(mm$Distances)

##-----------------------------------------------------------------------
## Analyze causal impact of a made-up weather intervention in Copenhagen
## Since this is weather data it is a not a very meaningful example.
## This is merely to demonstrate the functionality.
##-----------------------------------------------------------------------
results <- MarketMatching::inference(matched_markets = mm,
                                     test_market = "CPH",
                                     analyze_betas=FALSE,
                                     end_post_period = "2015-10-01",
                                     prior_level_sd = 0.002)

## Plot the impact
results$PlotCumulativeEffect

## Plot actual observations for test market (CPH) versus the expectation (based on the control)
results$PlotActualVersusExpected
```

---

weather *Weather dataset*

Description

The data was extracted using the weatherData package It contains average temperature readings for 19 airports for 2014.

Usage

```
weather
```

Format

A time series dataset with 6,935 rows and 3 variables (19 airports and 365 days):

- Area: Airport code
- Date: Date
- Mean_TemperatureF: Average temperature

Index