Package 'CausalImpact'

September 15, 2017

Title Inferring Causal Effects using Bayesian Structural Time-Series Models

Date 2017-08-16

Author Kay H. Brodersen <kbrodersen@google.com>, Alain Hauser <alhauser@google.com>

Maintainer Alain Hauser <alhauser@google.com>

URL https://google.github.io/CausalImpact/

Description Implements a Bayesian approach to causal impact estimation in time series, as described in Brodersen et al. (2015) <DOI:10.1214/14-AOAS788>. See the package documentation on GitHub <https://google.github.io/CausalImpact/> to get started.

Version 1.2.3

VignetteBuilder knitr

License Apache License 2.0 | file LICENSE

Imports assertthat (>= 0.2.0), Boom, dplyr, ggplot2, zoo

Depends bsts (>= 0.7.0)

Suggests knitr, testthat

NeedsCompilation no

Repository CRAN

Date/Publication 2017-09-14 22:22:48 UTC

R topics documented:

as.CausalImpact *Coercion to a* CausalImpact *object*

Description

Method for coercing objects to class CausalImpact.

This function serves as a template to implement S3 methods for coercing other objects to CausalImpact objects in other packages. The function itself just dispatches a call to the appropriate S3 method.

Usage

```
as.CausalImpact(x, ...)
```

Arguments

x	Any R object that should be coerced to a CausalImpact object.
...	Optional additional arguments (not currently used).

Author(s)

Kay H. Brodersen <kbrodersen@google.com>

CausalImpact *Inferring causal impact using structural time-series models*

Description

CausalImpact() performs causal inference through counterfactual predictions using a Bayesian structural time-series model.

See the package documentation (http://google.github.io/CausalImpact/) to understand the underlying assumptions. In particular, the model assumes that the time series of the treated unit can be explained in terms of a set of covariates which were themselves not affected by the intervention whose causal effect we are interested in.

The easiest way of running a causal analysis is to call CausalImpact() with data, pre.period, post.period, model.args (optional), and alpha (optional). In this case, a time-series model is automatically constructed and estimated. The argument model.args offers some control over the model. See Example 1 below.

An alternative is to supply a custom model. In this case, the function is called with bsts.model, post.period.response, and alpha (optional). See Example 3 below.

Usage

```
CausalImpact(data = NULL, pre.period = NULL,
  post.period = NULL, model.args = NULL,
  bsts.model = NULL, post.period.response = NULL,
  alpha = 0.05)
```

Arguments

data	Time series of response variable and any covariates. This can be a zoo object, a vector, a matrix, or a data.frame. In any of these cases, the response variable must be in the first column, and any covariates in subsequent columns. A zoo object is recommended, as its time indices will be used to format the x-axis in plot().
pre.period	A vector specifying the first and the last time point of the pre-intervention period in the response vector y. This period can be thought of as a training period, used to determine the relationship between the response variable and the covariates. If data is a zoo object with a time attribute, pre.period must be indicated using the same time scale (i.e. using the same class as time(data), see Example 2 below). If data doesn't have a time attribute, post.period is indicated with indices.
post.period	A vector specifying the first and the last day of the post-intervention period we wish to study. This is the period after the intervention has begun whose effect we are interested in. The relationship between response variable and covariates, as determined during the pre-period, will be used to predict how the response variable should have evolved during the post-period had no intervention taken place. If data is a zoo object with a time attribute, post.period must be indicated using the same time scale. If data doesn't have a time attribute, post.period is indicated with indices.
model.args	Further arguments to adjust the default construction of the state-space model used for inference. One particularly important parameter is prior.level.sd, which specifies our a priori knowledge about the volatility of the data. For even more control over the model, you can construct your own model using the bsts package and feed the fitted model into CausalImpact(), as shown in Example 3.
bsts.model	Instead of passing in data and having CausalImpact() construct a model, it is possible to create a custom model using the bsts package. In this case, omit data, pre.period, and post.period. Instead only pass in bsts.model, post.period.response, and alpha (optional). The model must have been fitted on data where the response variable was set to NA during the post-treatment period. The actual observed data during this period must then be passed to the function in post.period.response.
post.period.response	Actual observed data during the post-intervention period. This is required if and only if a fitted bsts.model is provided instead of data.
alpha	Desired tail-area probability for posterior intervals. Defaults to 0.05, which will produce central 95% intervals.

Value

CausalImpact() returns a CausalImpact object containing the original observed response, its counterfactual predictions, as well as pointwise and cumulative impact estimates along with posterior credible intervals. Results can summarised using summary() and visualized using plot(). The object is a list with the following fields:

- `series`. Time-series object (`zoo`) containing the original response `response` as well as the computed inferences. The added columns are listed in the table below.
- `summary`. Summary statistics for the post-intervention period. This includes the posterior expectation of the overall effect, the corresponding posterior credible interval, and the posterior probability that the intervention had any effect, expressed in terms of a one-sided p-value. Note that checking whether the posterior interval includes zero corresponds to a two-sided hypothesis test. In contrast, checking whether the p-value is below `alpha` corresponds to a one-sided hypothesis test.
- `report`. A suggested verbal interpretation of the results.
- `model`. A list with four elements `pre.period`, `post.period`, `bsts.model` and `alpha`. `pre.period` and `post.period` indicate the first and last time point of the time series in the respective period, `bsts.model` is the fitted model returned by `bsts()`, and `alpha` is the user-specified tail-area probability.

The field `series` is a `zoo` time-series object with the following columns:

`response`	Observed response as supplied to `CausalImpact()`.
`cum.response`	Cumulative response during the modeling period.
`point.pred`	Posterior mean of counterfactual predictions.
`point.pred.lower`	Lower limit of a (`1 - alpha`) posterior interval.
`point.pred.upper`	Upper limit of a (`1 - alpha`) posterior interval.
`cum.pred`	Posterior cumulative counterfactual predictions.
`cum.pred.lower`	Lower limit of a (`1 - alpha`) posterior interval.
`cum.pred.upper`	Upper limit of a (`1 - alpha`) posterior interval.
`point.effect`	Point-wise posterior causal effect.
`point.effect.lower`	Lower limit of the posterior interval (as above).
`point.effect.lower`	Upper limit of the posterior interval (as above).
`cum.effect`	Posterior cumulative effect.
`cum.effect.lower`	Lower limit of the posterior interval (as above).
`cum.effect.lower`	Upper limit of the posterior interval (as above).

Note

Optional arguments can be passed as a list in `model.args`, providing additional control over model construction:

- `niter`. Number of MCMC samples to draw. Higher numbers yield more accurate inferences. Defaults to 1000.
- `standardize.data`. Whether to standardize all columns of the data using moments estimated from the pre-intervention period before fitting the model. This is equivalent to an empirical Bayes approach to setting the priors. It ensures that results are invariant to linear transformations of the data. Defaults to `TRUE`.
- `prior.level.sd`. Prior standard deviation of the Gaussian random walk of the local level, expressed in terms of data standard deviations. Defaults to 0.01, a typical choice for well-behaved and stable datasets with low residual volatility after regressing out known predictors (e.g., web searches or sales in high quantities). When in doubt, a safer option is to use 0.1,

as validated on synthetic data, although this may sometimes give rise to unrealistically wide prediction intervals.

- `nseasons`. Period of the seasonal components. In order to include a seasonal component, set this to a whole number greater than 1. For example, if the data represent daily observations, use 7 for a day-of-week component. This interface currently only supports up to one seasonal component. To specify multiple seasonal components, use `bsts` to specify the model directly, then pass the fitted model in as `bsts.model`. Defaults to 1, which means no seasonal component is used.
- `season.duration`. Duration of each season, i.e., number of data points each season spans. For example, to add a day-of-week component to data with daily granularity, supply the arguments `model.args = list(nseasons = 7, season.duration = 1)`. Alternatively, use `model.args = list(nseasons = 7, season.duration = 24)` to add a day-of-week component to data with hourly granularity. Defaults to 1.
- `dynamic.regression`. Whether to include time-varying regression coefficients. In combination with a time-varying local trend or even a time-varying local level, this often leads to overspecification, in which case a static regression is safer. Defaults to `FALSE`.

Author(s)

Kay H. Brodersen <kbrodersen@google.com>

Examples

```
# Example 1
#
# Example analysis on a simple artificial dataset
# consisting of a response variable y and a
# single covariate x1.
set.seed(1)
x1 <- 100 + arima.sim(model = list(ar = 0.999), n = 52)
y <- 1.2 * x1 + rnorm(52)
y[41:52] <- y[41:52] + 10
data <- cbind(y, x1)
pre.period <- c(1, 40)
post.period <- c(41, 52)
impact <- CausalImpact(data, pre.period, post.period)

# Print and plot results
summary(impact)
summary(impact, "report")
plot(impact)
plot(impact, "original")
plot(impact$model$bsts.model, "coefficients")

# For further output, type:
names(impact)

## Not run:
# Example 2
#
```

```
# Weekly time series: same data as in example 1, annotated
# with dates.
times <- seq.Date(as.Date("2016-01-03"), by = 7, length.out = 52)
data <- zoo(cbind(y, x1), times)

impact <- CausalImpact(data, times[pre.period], times[post.period])

summary(impact)  # Same as in example 1.
plot(impact)  # The plot now shows dates on the x-axis.

# Example 3
#
# For full flexibility, specify a custom model and pass the
# fitted model to CausalImpact(). To run this example, run
# the code for Example 1 first.
post.period.response <- y[post.period[1] : post.period[2]]
y[post.period[1] : post.period[2]] <- NA
ss <- AddLocalLevel(list(), y)
bsts.model <- bsts(y ~ x1, ss, niter = 1000)
impact <- CausalImpact(bsts.model = bsts.model,
                       post.period.response = post.period.response)
plot(impact)

## End(Not run)
```

`CausalImpactMethods` *Printing and plotting a* `CausalImpact` *object*

Description

Methods for printing and plotting the results of an analysis results object returned by `CausalImpact()`.

Usage

```
  ## S3 method for class 'CausalImpact'
summary(object, ...)
  ## S3 method for class 'CausalImpact'
print(x, ...)
  ## S3 method for class 'CausalImpact'
plot(x, ...)
```

Arguments

`object`	A `CausalImpact` results object, as returned by `CausalImpact()`.
`x`	A `CausalImpact` results object, as returned by `CausalImpact()`.
`...`	Optional additional arguments. For `summary()` and `print()`, the first optional argument is `output`, which can be `"summary"` (default) or `"report"`. Partial matches are allowed. Furthermore, `digits` can be used to customize the precision of the output, e.g.: `summary(..., "summary", digits = 3)`

For `plot()`, the additional argument `metrics` can be used to specify which panels to include in the plot. The argument can be any combination of `"original"`, `"pointwise"`, `"cumulative"`. Partial matches are allowed.

Author(s)

Kay H. Brodersen <kbrodersen@google.com>

Examples

```
## Not run:
impact <- CausalImpact(...)

# Print a summary table
print(impact)
summary(impact)

# Print a verbal analysis description
print(impact, "report")
summary(impact, "report")

# Create a plot
plot(impact)
plot(impact, "original")
plot(impact, "pointwise")
plot(impact, "cumulative")
plot(impact, c("original", "pointwise"))

# Customize a plot
impact.plot <- plot(impact)
impact.plot <- impact.plot + theme_bw(base_size = 20)

## End(Not run)
```

Index