Time series analysis in R

Predicting the future with Facebook Prophet and Anomaly Detection with Twitter AnomalyDetection

Hicham Zmarrou, PhD

2018-11-29

Aims of this lesson

Introduce you to a number of modern time series analysis techniques. In this lesson, I will talk about

- a new time series forcasting technique that was developed by Facebook named `Prophet`.

Aims of this lesson

Introduce you to a number of modern time series analysis techniques. In this lesson, I will talk about

- a new time series forcasting technique that was developed by Facebook named `Prophet`.
- a new time series anomaly detection technique that was developed by twitter named `anomalyDetection`

Aims of this lesson

Introduce you to a number of modern time series analysis techniques. In this lesson, I will talk about

- a new time series forcasting technique that was developed by Facebook named `Prophet`.
- a new time series anomaly detection technique that was developed by twitter named `anomalyDetection`
- a new time series technique for intervention analysis that was developed by Google named `CausalImpact`

Introduction

Predicting the future with Facebook Prophet

Introduction

According to the article on Facebook Research, Prophet was initially developed for the purpose of creating high quality business forecasts. This library tries to address the following difficulties common to many business time series:

- Seasonal effects caused by human behavior: weekly, monthly and yearly cycles, dips and peaks on public holidays.

The authors claim that, even with the default settings, in many cases, their library produces forecasts as accurate as those delivered by experienced analysts.

Introduction

According to the article on Facebook Research, Prophet was initially developed for the purpose of creating high quality business forecasts. This library tries to address the following difficulties common to many business time series:

- Seasonal effects caused by human behavior: weekly, monthly and yearly cycles, dips and peaks on public holidays.
- Changes in trend due to new products and market events.

The authors claim that, even with the default settings, in many cases, their library produces forecasts as accurate as those delivered by experienced analysts.

Introduction

According to the article on Facebook Research, Prophet was initially developed for the purpose of creating high quality business forecasts. This library tries to address the following difficulties common to many business time series:

- Seasonal effects caused by human behavior: weekly, monthly and yearly cycles, dips and peaks on public holidays.
- Changes in trend due to new products and market events.
- Outliers.

The authors claim that, even with the default settings, in many cases, their library produces forecasts as accurate as those delivered by experienced analysts.

The Prophet Forecasting Model

In its essence, `Prophet` utilizes an additive regression model $y(t)$ comprising the following components:

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t$$

where:

- Trend $g(t)$ models non-periodic changes.

The Prophet Forecasting Model

In its essence, `Prophet` utilizes an additive regression model $y(t)$ comprising the following components:

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t$$

where:

- Trend $g(t)$ models non-periodic changes.
- Seasonality $s(t)$ represents periodic changes.

The Prophet Forecasting Model

In its essence, `Prophet` utilizes an additive regression model $y(t)$ comprising the following components:

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t$$

where:

- Trend $g(t)$ models non-periodic changes.
- Seasonality $s(t)$ represents periodic changes.
- Holidays component $h(t)$ contributes information about holidays and events.

The Prophet Forecasting Model: Trend

The Prophet library implements two possible trend models for $g(t)$.

The first one is called nonlinear, saturating growth. It is represented in the form of the logistic growth model:

$$g(t) = \frac{C}{1 + e^{(-k(t-m))}}$$

where:

- C is the carrying capacity (that is the curve's maximum value).

The second trend model is a simple Piecewise Linear Model with a constant rate of growth.

The Prophet Forecasting Model: Trend

The Prophet library implements two possible trend models for $g(t)$.

The first one is called nonlinear, saturating growth. It is represented in the form of the logistic growth model:

$$g(t) = \frac{C}{1 + e^{(-k(t-m))}}$$

where:

- C is the carrying capacity (that is the curve's maximum value).
- k is the growth rate (which represents "the steepness" of the curve and m is an offset parameter.

The second trend model is a simple Piecewise Linear Model with a constant rate of growth.

The Prophet Forecasting Model: Seasonality

The seasonal component $s(t)$ provides a flexible model of periodic changes due to weekly and yearly seasonality.

Weekly seasonal data is modeled with dummy variables. Six new variables are added: `monday`, `tuesday`, etc. which take values 0 or 1 depending on the day of the week.

Yearly seasonality model in `Prophet` relies on Fourier series.

The Prophet Forecasting Model: Holidays and Events and Error

The component $h(t)$ represents predictable abnormal days of the year including those on irregular schedules, e.g., Black Fridays.

To utilize this feature, the analyst needs to provide a custom list of events.

The error term ϵ_t represents information that was not reflected in the model. Usually it is modeled as normally distributed noise.

Prophet implementation in R

The prophet package provides a prophet function that performs fitting and returns a model object. You can then call predict and plot on this model object.

```
library(prophet)
```

First we read in the data and create the outcome variable. This is a dataframe with columns ds and y, containing the date and numeric value respectively. The ds column should be YYYY-MM-DD for a date, or YYYY-MM-DD HH:MM:SS for a timestamp. We use here the log number of views to Peyton Manning's Wikipedia page, available here.

Prophet implementation in R

```
df <- read.csv("example_wp_log_peyton_manning.csv")
```

We call the prophet function to fit the model. The first argument is the historical dataframe. Additional arguments control how Prophet fits the data and are described in later pages of this documentation.

```
m <- prophet(df)
```

Prophet implementation in R

Predictions are made on a dataframe with a column ds containing the dates for which predictions are to be made. The make_future_dataframe function takes the model object and a number of periods to forecast and produces a suitable dataframe. By default it will also include the historical dates so we can evaluate in-sample fit.

```
future <- make_future_dataframe(m, periods = 365)
tail(future,3)
```

```
             ds
3268 2017-01-17
3269 2017-01-18
3270 2017-01-19
```

Prophet implementation in R

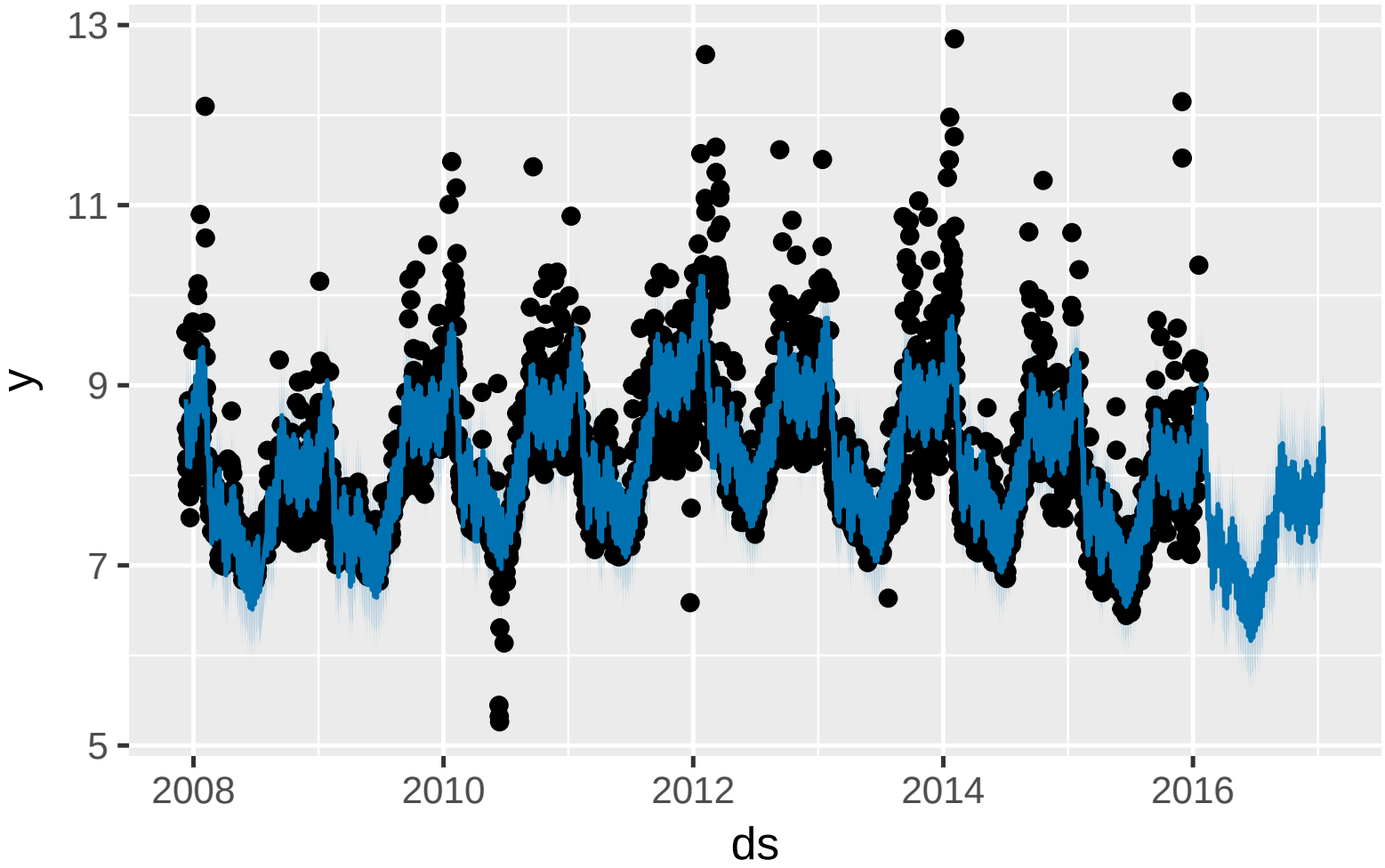
As with most modeling procedures in R, we use the generic `predict` function to get our forecast. The forecast object is a dataframe with a column `yhat` containing the `forecast`. It has additional columns for uncertainty intervals and seasonal components.

```
forecast <- predict(m, future)
tail(forecast[c('ds', 'yhat', 'yhat_lower', 'yhat_upper')])
```

```
             ds     yhat yhat_lower yhat_upper
3268 2017-01-17 8.313510   7.602899   8.997292
3269 2017-01-18 8.146114   7.417901   8.839822
3270 2017-01-19 8.158009   7.443551   8.875517
```

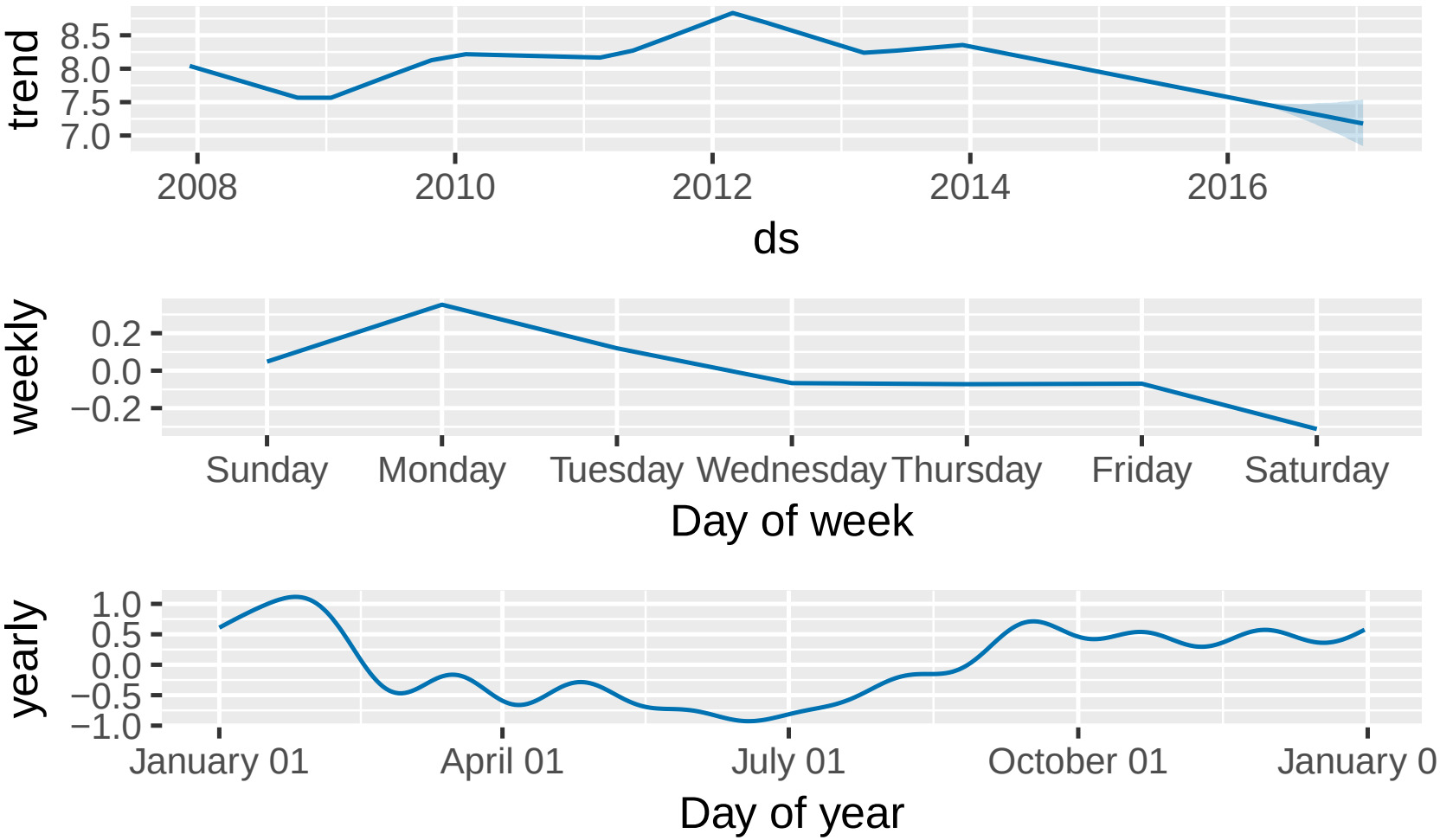
Prophet implementation in R

```
plot(m, forecast)
```



Prophet implementation in R

```
prophet_plot_components(m, forecast)
```



Prophet implementation in R

An interactive plot of the forecast using Dygraphs can be made with the command `dyplot.prophet(m, forecast)`

More details about the options available for each method are available in the docstrings, for example, via`?prophet` or `?fit.prophet`. This documentation is also available in the reference manual on CRAN.

Please complete
predicting_bike_sharing_demand_with_prophet_exercise

Anomaly Detection with Twitter AnomalyDetection

STL (Seasonal and trend decomposition using Loess)

The core idea of STL is that a time series can be decomposed into three components: seasonal, trend and remainder

$$y(t) = g(t) + s(t) + \epsilon_t$$

where:

- Trend $g(t)$ models non-periodic changes.

STL (Seasonal and trend decomposition using Loess)

The core idea of STL is that a time series can be decomposed into three components: seasonal, trend and remainder

$$y(t) = g(t) + s(t) + \epsilon_t$$

where:

- Trend $g(t)$ models non-periodic changes.
- Seasonality $s(t)$ represents periodic changes.

STL (Seasonal and trend decomposition using Loess)

The core idea of STL is that a time series can be decomposed into three components: seasonal, trend and remainder

$$y(t) = g(t) + s(t) + \epsilon_t$$

where:

- Trend $g(t)$ models non-periodic changes.
- Seasonality $s(t)$ represents periodic changes.
- Remainder (or irregular or error)component at period t

STL (Seasonal and trend decomposition using Loess)

STL
decomposition

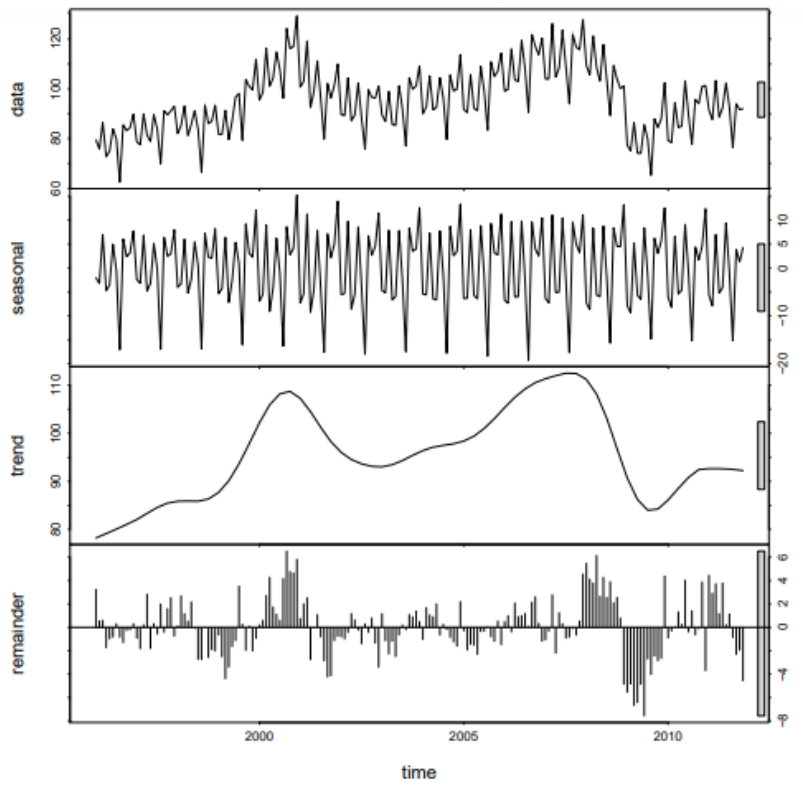


Figure 1: STL decomposition

STL (Seasonal and trend decomposition using Loess)

The use of the stl function can be demonstrated using one of the data sets available within the base R installation. The well used nottem data set (Average Monthly Temperatures at Nottingham, 1920-1939) is a good starting point.

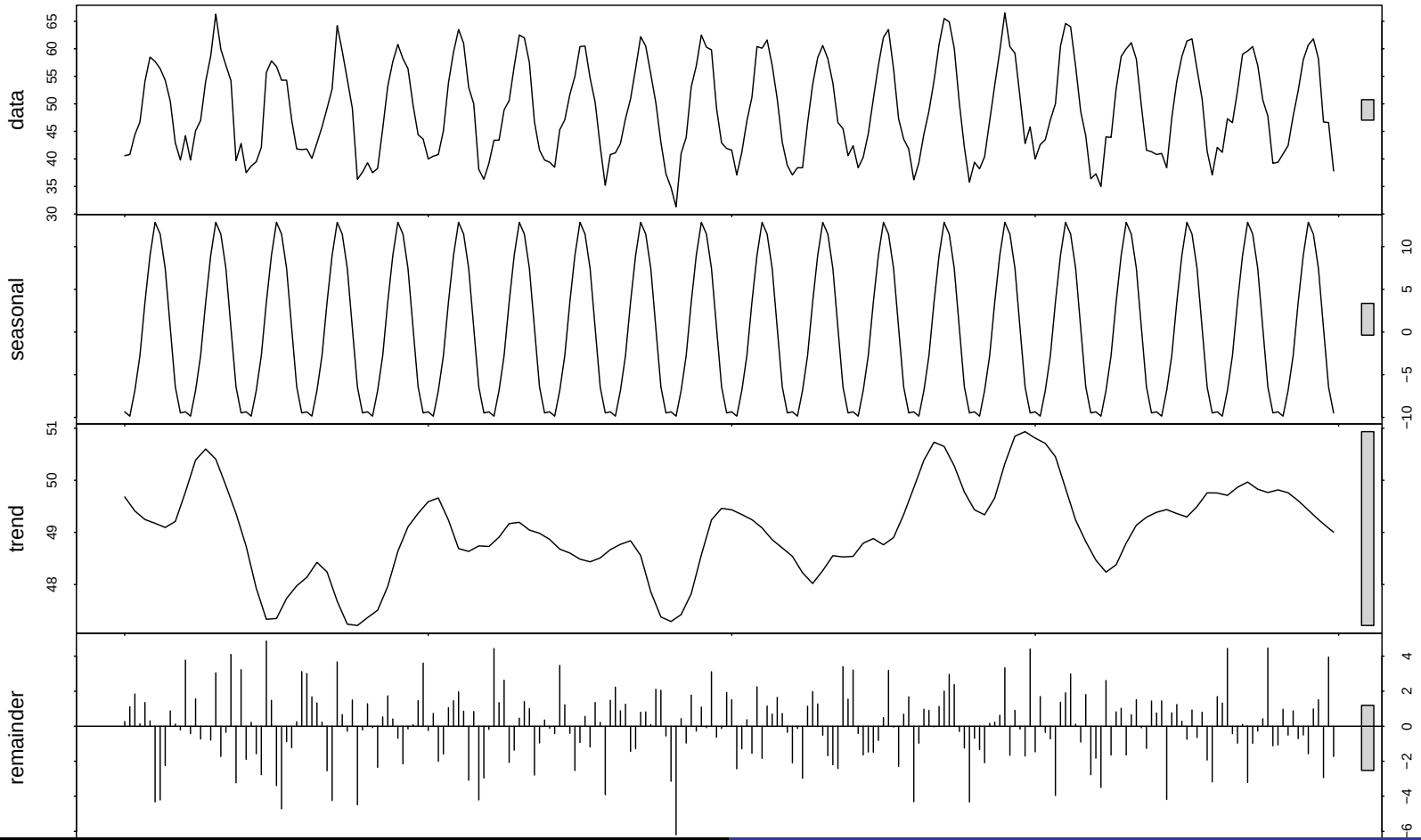
```
head(nottem)
```

```
[1] 40.6 40.8 44.4 46.7 54.1 58.5
```

```
## [1] 40.6 40.8 44.4 46.7 54.1 58.5
modelStl <- stl(nottem, s.window = "periodic")
```

STL (Seasonal and trend decomposition using Loess)

```
plot(modelStl)
```



Grubbs Test and ESD

These techniques use time series decomposition STL to determine the seasonal component of a given time series. S-ESD then applies Extreme Studentized Deviate (ESD) on the resulting time series to detect the anomalies.

Grubbs Test

H0 :There are no outliers in the data set
H1 :There is at least one outlier in the data set
The Grubbs' test statistic is defined as follows:

$$G = \frac{\max |x_t - \bar{x}|}{s}$$

where, $\bar{x}$ and s denote the mean and variance of the time series X.

Grubbs Test

For the two-sided test, the hypothesis of no outliers is rejected at significance level α if

$$G_{crit} > \frac{N-1}{\sqrt{N}} \sqrt{\frac{t^2_{\alpha/(2N),N-2}}{N-2+t^2_{\alpha/(2N),N-2}}}$$

The largest data point in the time series that is greater than the test statistic is labeled as an anomaly.

Extreme Studentized Deviate (ESD)

ESD test is a generalization of Grubbs' Test and handles more than one outlier. All you need to do is provide an upper bound on the number of potential outliers. `ESD` computes the following test statistic for the `k` most extreme values in the data set.

We test the null hypothesis that the data has no outliers vs. the alternative hypothesis that there are at most k outliers (for some user specified value of k).

H0 :There are no outliers in the data set

H1 :There are at most k outliers (for some user specified value of k)

Extreme Studentized Deviate (ESD)

To test the data set X with n elements we generate k test statistics $G_1, G_2, .., G_k$ where each G_j is a two-tailed Grubbs statistic, defined as follows:

$$S_1 = S$$

$\bar{x_j}$ is the mean of S_j and s_j is the standard deviation of S_j

$$G_j = \frac{\max\{|x_t - \bar{x_j}|, x_t \in S_j\}}{s_j}$$

$S_{j+1} = S_j \setminus x_j$ where $x_j =$ the element in S_j such that $|x_j - x|$ is maximized

ESD

Essentially you run k separate Grubbs tests, testing whether $G_j > G_{j_{crit}}$ where $G_{j_{crit}}$ is G_{crit} as described above, but adjusted for the correct value of the sample size; i.e. n is replaced by $n - j + 1$.

Now let r be the largest value of $j \leq k$ such that $G_j > G_{j_{crit}}$. Then we conclude there are r outliers, namely $x_1, \cdots x_r$. If $r = 0$ there are no outliers.

Note that if $G_j > G_{j_{crit}}$ and $h < j$, then both x_h and x_j are outliers even if $G_h \leq G_{h_{crit}}$.

Seasonal ESD (S-ESD)

- Many time series data exhibits heavy seasonality.

Seasonal ESD (S-ESD)

- Many time series data exhibits heavy seasonality.
- In most cases, the underlying distribution exhibits a multimodal distribution.

Seasonal ESD (S-ESD)

- Many time series data exhibits heavy seasonality.
- In most cases, the underlying distribution exhibits a multimodal distribution.
- This limits the applicability of existing anomaly detection techniques such as Grubbs and ESD as the existing techniques assume a normal data distribution.

Seasonal ESD (S-ESD)

- Many time series data exhibits heavy seasonality.
- In most cases, the underlying distribution exhibits a multimodal distribution.
- This limits the applicability of existing anomaly detection techniques such as Grubbs and ESD as the existing techniques assume a normal data distribution.
- We use time series decomposition wherein a given time series X is decomposed into three parts seasonal S_X, trend T_X, and residual R_X

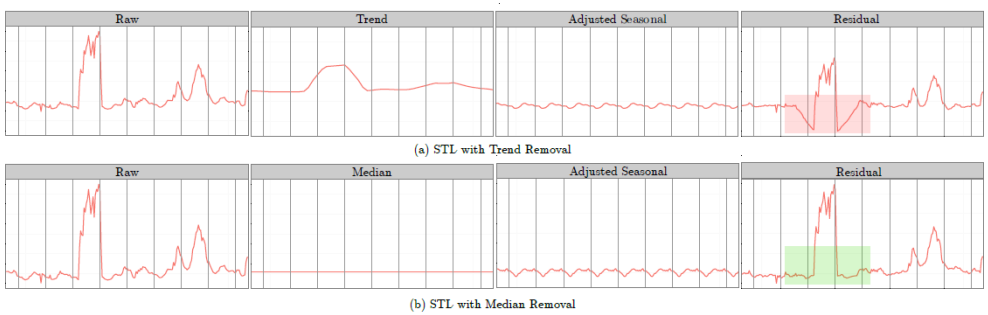
Seasonal ESD (S-ESD)

- Many time series data exhibits heavy seasonality.
- In most cases, the underlying distribution exhibits a multimodal distribution.
- This limits the applicability of existing anomaly detection techniques such as Grubbs and ESD as the existing techniques assume a normal data distribution.
- We use time series decomposition wherein a given time series X is decomposed into three parts seasonal S_X, trend T_X, and residual R_X
- Seasonal ESD algorithm uses a modified STL decomposition to extract the residual component of the input time series and then applies ESD to the residuals to detect anomalies.

Seasonal ESD (S-ESD)

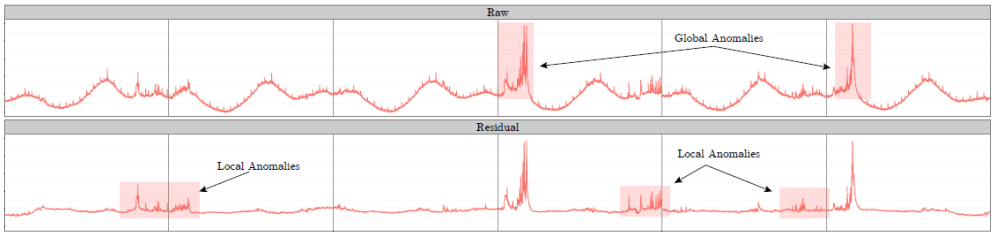
- Many time series data exhibits heavy seasonality.
- In most cases, the underlying distribution exhibits a multimodal distribution.
- This limits the applicability of existing anomaly detection techniques such as Grubbs and ESD as the existing techniques assume a normal data distribution.
- We use time series decomposition wherein a given time series X is decomposed into three parts seasonal S_X, trend T_X, and residual R_X
- Seasonal ESD algorithm uses a modified STL decomposition to extract the residual component of the input time series and then applies ESD to the residuals to detect anomalies.
- This two step process allows S-ESD to detect **both** global and anomalies that extend beyond the expected seasonal minimum and maximum and local anomalies that would otherwise be masked by the seasonality.

Seasonal ESD (S-ESD)



STL (a) vs. STL Variant (b) Decomposition

Figure 2: STL vs STL variant



Global and Local Anomalies Exposed using S-ESD

Figure 3: Global and local anomalies using S-EDS

Seasonal ESD (S-ESD)

Algorithm 1 S-ESD Algorithm

Input:

X = A time series

n = number of observations in X

k = max anomalies (iterations in ESD)

Output:

X_A = An anomaly vector wherein each element is a tuple $(timestamp, observed\ value)$

Require:

$k \leq (n \times .49)$

1. Extract seasonal component S_X using STL Variant
2. Compute median $\tilde{X}$

/* Compute residual */

3. $R_X = X - S_X - \tilde{X}$

/* Detect anomalies vector X_A using ESD */

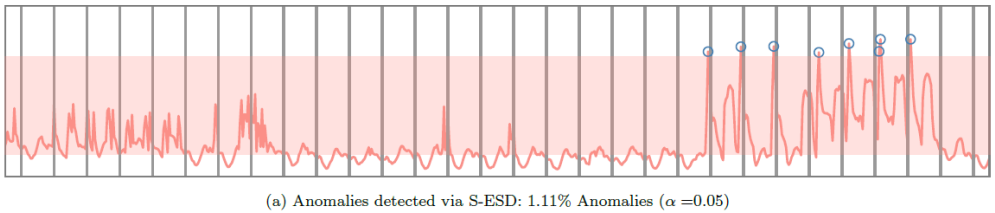
4. $X_A = \text{ESD}(R, k)$

return X_A

Figure 4: S-ESD Algorithm

S-ESD Limitations

S-ESD does not perform well when applied to data sets that have a high percentage of anomalies. This is exemplied by Figure 8a wherein S-ESD does not capture the anomalies corresponding to the region highlighted by the red rectangle.



(a) Anomalies detected via S-ESD: 1.11% Anomalies (α =0.05)

(b) Anomalies detected via S-H-ESD: 29.68% Anomalies (α =0.05)

S-ESD (a) vs. S-H-ESD performance with highly anomalous data

Figure 5: S-ESD vs S-H-EDS Algorithm

Seasonal Hybrid ESD (SHESD)

- Use more robust statistical techniques and metrics during the calculation of the test statistic

Mean	Median	Std. Dev.	MAD
6.59	5.45	3.08	1.52

Figure 6: S-ESD vs S-H-EDS Algorithm

Seasonal Hybrid ESD (SHESD)

- Use more robust statistical techniques and metrics during the calculation of the test statistic
- Use se the median and MAD, as these metrics exhibit a higher breakdown point

Mean	Median	Std. Dev.	MAD
6.59	5.45	3.08	1.52

Figure 6: S-ESD vs S-H-EDS Algorithm

Anomaly Detection with Twitter pakkage in R

```
#install.packages("devtools")
#devtools::install_github("twitter/AnomalyDetection")
library(AnomalyDetection)
```

The functionAnomalyDetectionTs is called to detect one or more statistically significant anomalies in the input time series. The documentation of the function AnomalyDetectionTs, which can be seen by using the following command, details the input arguments and the output of the function AnomalyDetectionTs.

```
# help(AnomalyDetectionTs)
```

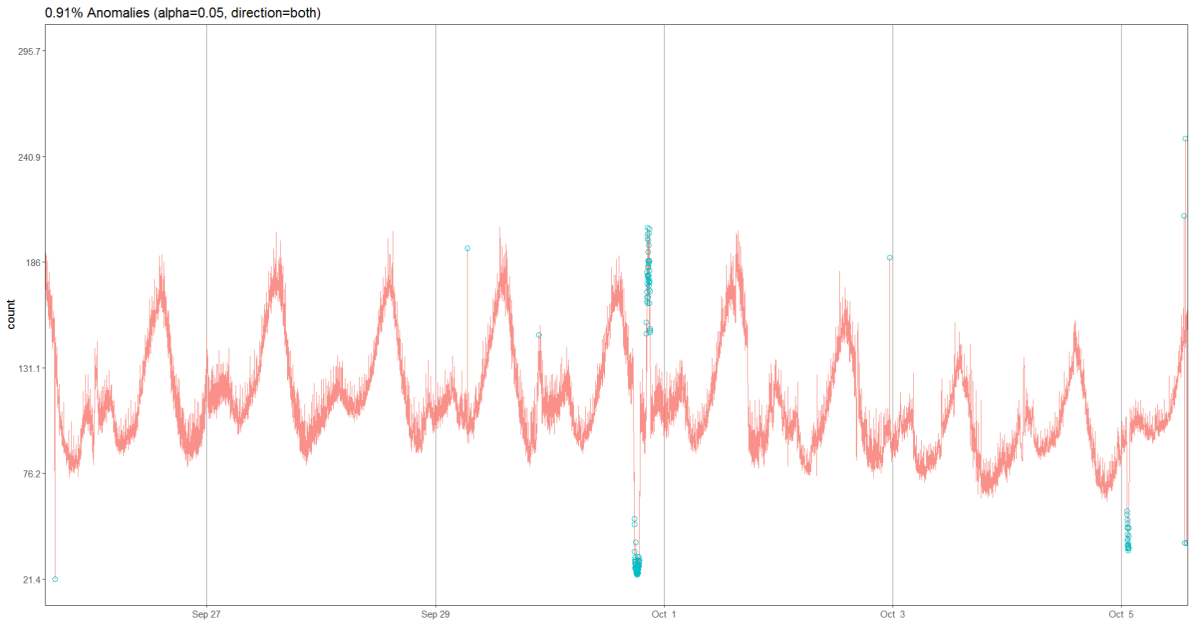
Anomaly Detection with Twitter pakkage in R

The function `AnomalyDetectionVec` is called to detect one or more statistically significant anomalies in a vector of observations. The documentation of the function `AnomalyDetectionVec`, which can be seen by using the following command, details the input arguments and the output of the function AnomalyDetectionVec.

```
# help(AnomalyDetectionVec)
```

Anomaly Detection with Twitter pakkage in R

```
data(raw_data)
res = AnomalyDetectionTs(raw_data,
                         max_anoms=0.02,
                         direction='both', plot=TRUE)
res$plot
```



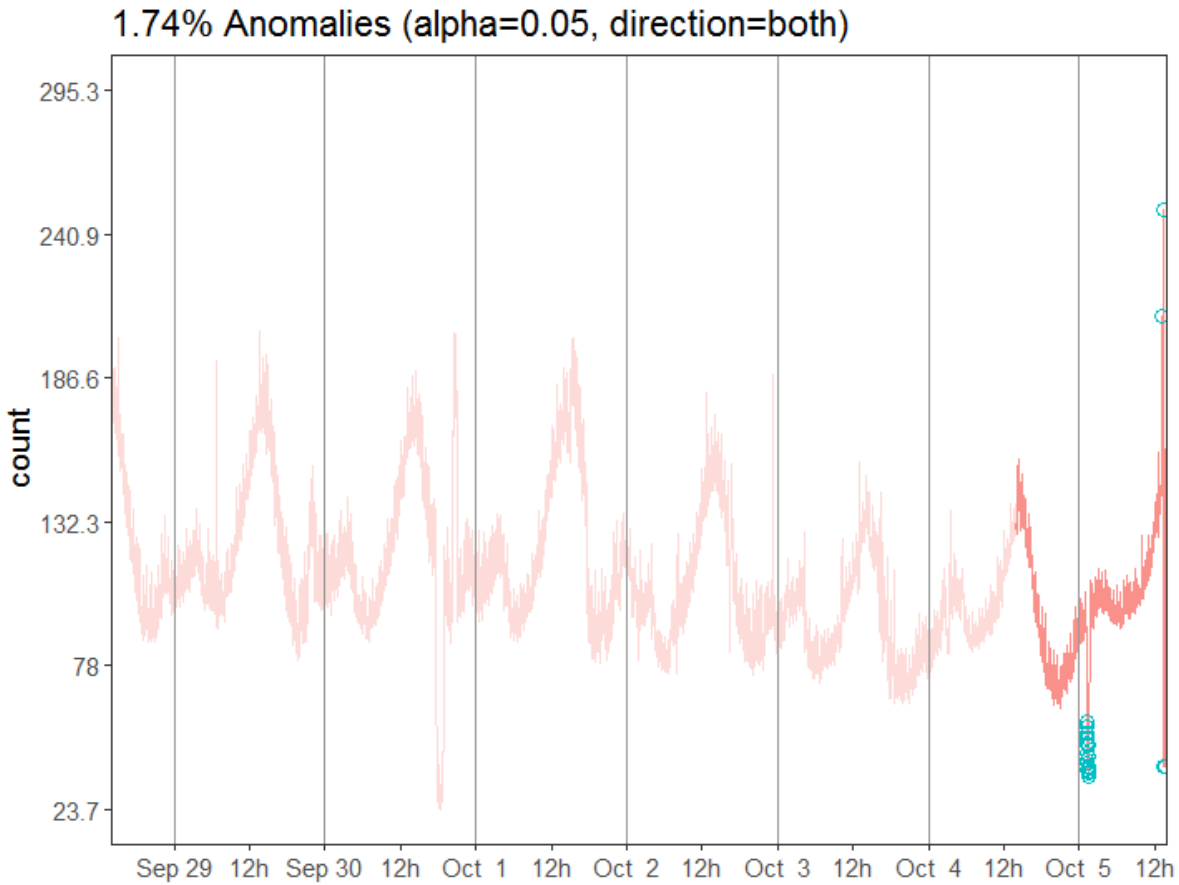
Anomaly Detection with Twitter pakkage in R

Often, anomaly detection is carried out on a periodic basis. For instance, at times, one may be interested in determining whether there was any anomaly yesterday. To this end, the package supports a flag `only_last` whereby one can subset the anomalies that occurred during the last day or last hour. Execute the following command:

```
res = AnomalyDetectionTs(raw_data, max_anoms=0.02,
                                direction='both',
                                only_last='day', plot=TRUE)
# res$plot
```

Anomaly Detection with Twitter pakkage in R

```
res$plot
```



Intervention analysis with Google CausalImpact

Introduction

- *Marketing analytics*: analyze the impact of a causal impact from a new local TV campaign, a major PR event, or the emergence of a new local competitor.

From an analytical standpoint these types of events all have one thing in common: The impact cannot be tracked at the individual customer level and hence we have to analyze the impact from a bird's eye view using time series analysis at the ensemble level.

when analyzing interventions through time series analysis we typically go through two steps, each of which can involve multiple analytical decisions:

Introduction

- *Marketing analytics*: analyze the impact of a causal impact from a new local TV campaign, a major PR event, or the emergence of a new local competitor.
- *Predictive maintenance*: when a maintenance event happend, did the intervention produce the intended impacts?

From an analytical standpoint these types of events all have one thing in common: The impact cannot be tracked at the individual customer level and hence we have to analyze the impact from a bird's eye view using time series analysis at the ensemble level.

when analyzing interventions through time series analysis we typically go through two steps, each of which can involve multiple analytical decisions:

Introduction

- *Marketing analytics*: analyze the impact of a causal impact from a new local TV campaign, a major PR event, or the emergence of a new local competitor.
- *Predictive maintenance*: when a maintenance event happend, did the intervention produce the intended impacts?
- *Medecine*: introduction new drugs

From an analytical standpoint these types of events all have one thing in common: The impact cannot be tracked at the individual customer level and hence we have to analyze the impact from a bird's eye view using time series analysis at the ensemble level.

when analyzing interventions through time series analysis we typically go through two steps, each of which can involve multiple analytical decisions:

Introduction

- Finding matching control time series for the test time series where the event took place using time series matching based on historical data prior to the event (the `pre period`).

The purpose of this lesson is to describe a robust approach to intervention analysis based on two key `R` packages: the `CausalImpact` package written by Kay Brodersen at Google and the dtw package available in CRAN.

Introduction

- Finding matching control time series for the test time series where the event took place using time series matching based on historical data prior to the event (the `pre period`).
- Analyzing the causal impact of the event by comparing the observed data for the test and control time series following the event (the "post period"), while factoring in differences between the time series prior to the event.

The purpose of this lesson is to describe a robust approach to intervention analysis based on two key `R` packages: the `CausalImpact` package written by Kay Brodersen at Google and the dtw package available in CRAN.

A traditional approach

For the inference step, the traditional approach is a Difference-in-Difference estimation.

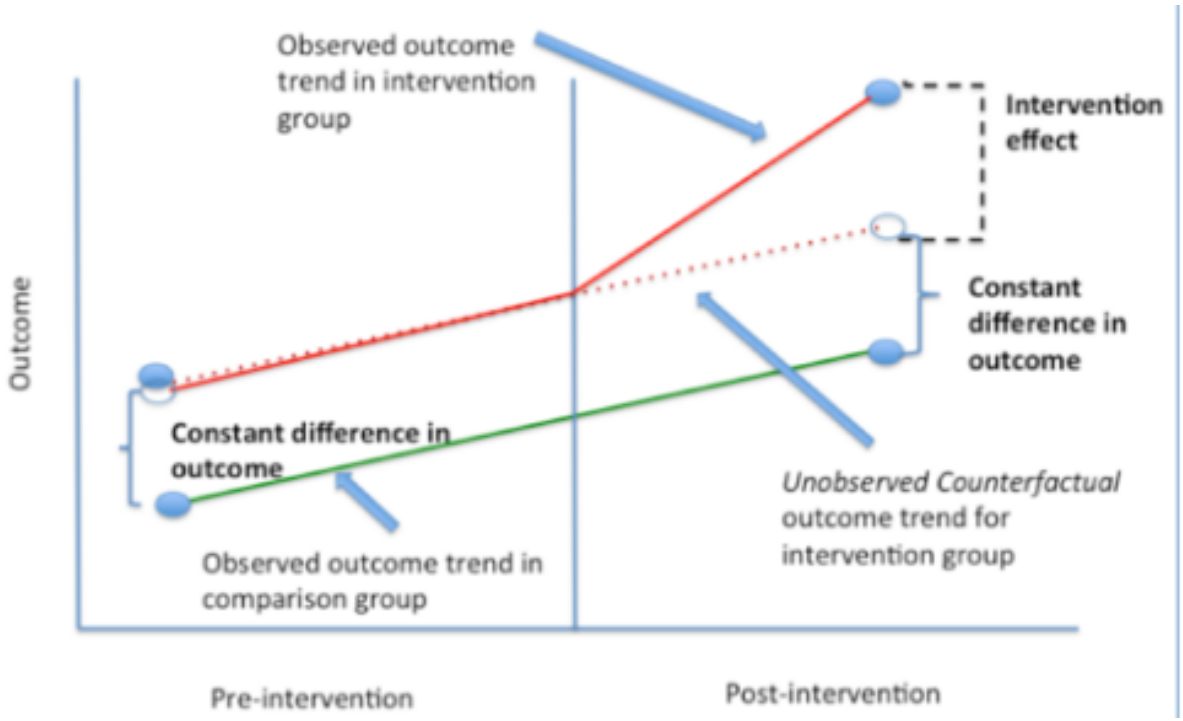


Figure 9: Difference-in-Difference, graphical illustration

A More Flexible and Robust Approach

A better approach is to use

- A technique named dynamic time warping to do the time series matching

A More Flexible and Robust Approach

A better approach is to use

- A technique named dynamic time warping to do the time series matching
- For the intervention analysis the `CausalImpact` package provides an approach that is more flexible and robust than the "diff in diff" model

A More Flexible and Robust Approach

- Pre-screening step: find the best control time series for each time series in the dataset using dynamic time warping.

A More Flexible and Robust Approach

- Pre-screening step: find the best control time series for each time series in the dataset using dynamic time warping.
- Inference step: fit a Bayesian structural time series model that utilizes the control time series identified in step 1 as predictors. Based on this model, create a synthetic control series by producing a counterfactual prediction for the post period assuming that the event did not take place. We can then calculate the difference between the synthetic control and the test market for the post-intervention period - which is the estimated impact of the event - and compare to the posterior interval to gauge uncertainty.

Some Technical Details

When we use `CausalImpact` the following structural time series model (state space model) is created for the pre-intervention period:

$$Y_t = \mu_t + X_t\beta + \epsilon_t, \epsilon_t \sim \mathcal{N}(0, \sigma_\epsilon^2),\ \mu_t = \mu_{t-1} + \nu_t,\ \nu_t \sim \mathcal{N}(0, \sigma_\nu^2)$$

Here x_t denotes the control time series and μ_t is the local level term. The local level term defines how the latent state evolves over time and is often referred to as the unobserved trend. The linear regression term, $x_t\beta$, averagesover the selected control time series.

Causal impact in R

```
#install.packages("CausalImpact")
library(CausalImpact)
# Creating an example dataset
set.seed(1)
x1 <- 100 + arima.sim(model = list(ar = 0.999), n = 100)
y <- 1.2 * x1 + rnorm(100)
y[71:100] <- y[71:100] + 10
data <- cbind(y, x1)
dim(data)
```

```
[1] 100   2
```

Causal impact in R

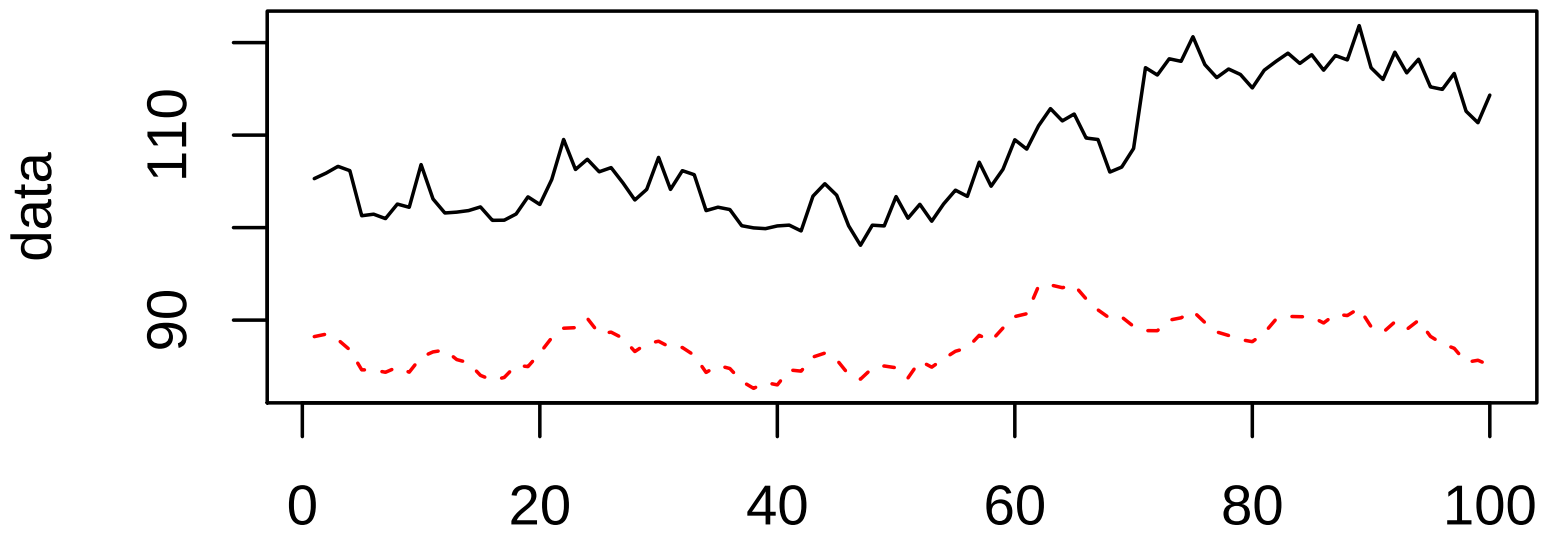
```
head(data)
```

```
             y       x1
[1,] 105.2950 88.21513
[2,] 105.8943 88.48415
[3,] 106.6209 87.87684
[4,] 106.1572 86.77954
[5,] 101.2812 84.62243
[6,] 101.4484 84.60650
```

Causal impact in R

We visualize the generated data using:

```
matplot(data, type = "l")
```



Running an analysis

To estimate a causal effect, we begin by specifying which period in the data should be used for training the model (pre-intervention period) and which period for computing a counterfactual prediction (post-intervention period).

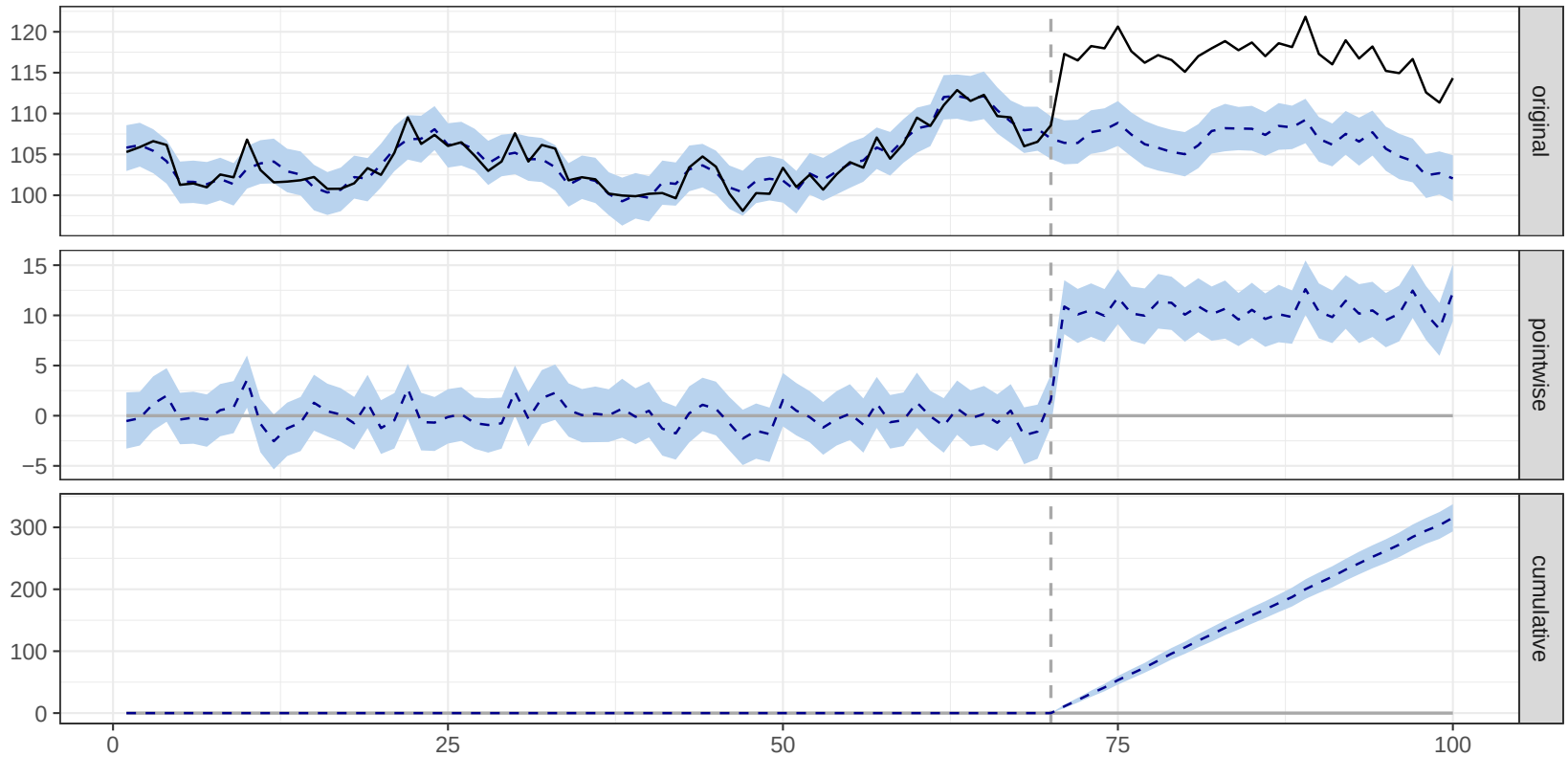
```
pre.period <- c(1, 70)
post.period <- c(71, 100)
```

To perform inference, we run the analysis using:

```
impact <- CausalImpact(data, pre.period, post.period)
```

Plotting the results

```
plot(impact)
```



Working with dates and times

```
time.points <- seq.Date(as.Date("2014-01-01"), by = 1, leng
data <- zoo(cbind(y, x1), time.points)
head(data)
```

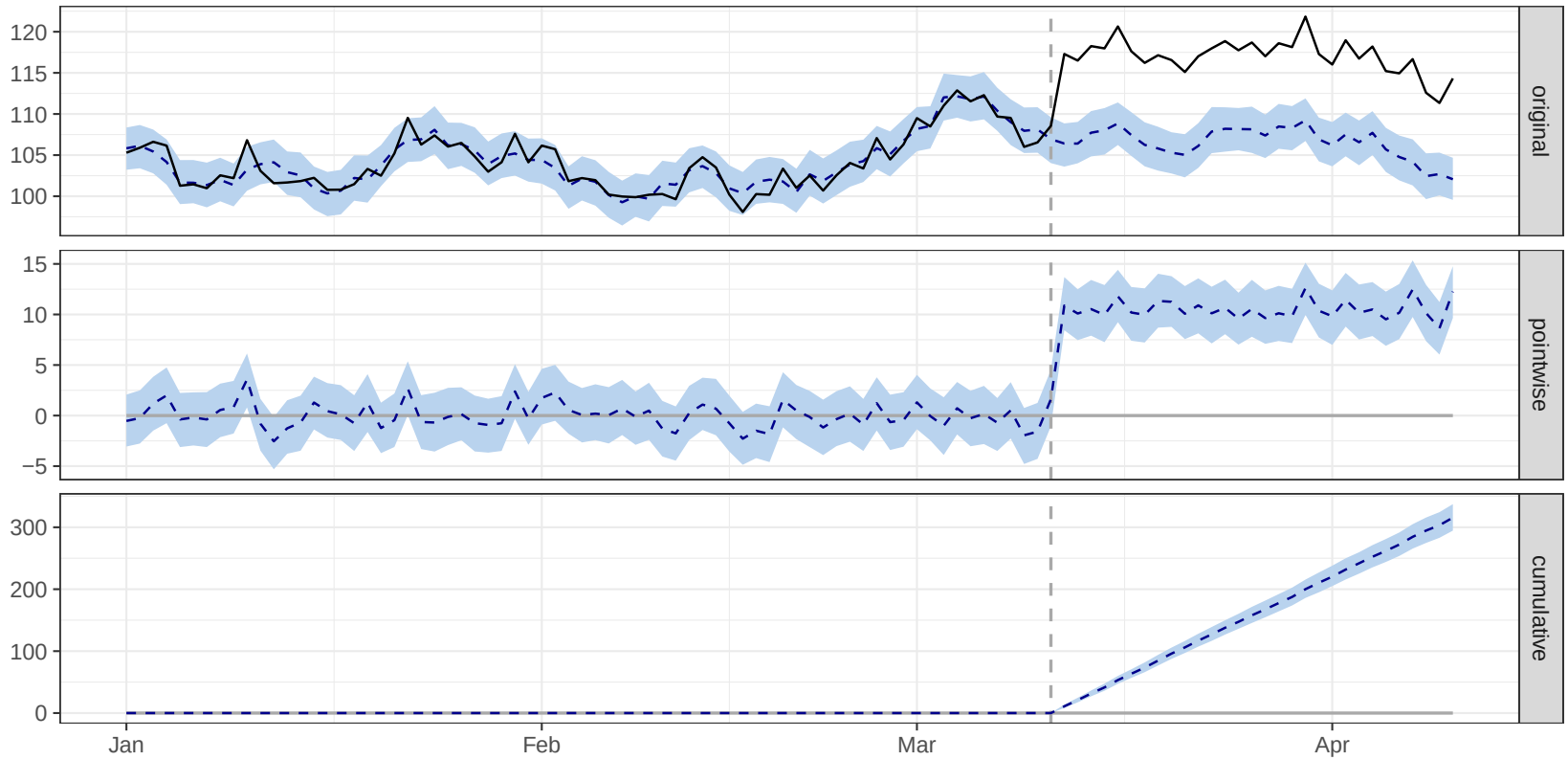
```
                   y       x1
2014-01-01 105.2950 88.21513
2014-01-02 105.8943 88.48415
2014-01-03 106.6209 87.87684
2014-01-04 106.1572 86.77954
2014-01-05 101.2812 84.62243
2014-01-06 101.4484 84.60650
```

Working with dates and times

```
## Period :
pre.period <- as.Date(c("2014-01-01", "2014-03-11"))
post.period <- as.Date(c("2014-03-12", "2014-04-10"))
```

Analysis and Plotting

```
impact <- CausalImpact(data, pre.period, post.period)
plot(impact)
```



Printing a summary table

```
summary(impact)
```

```
Posterior inference {CausalImpact}

                         Average        Cumulative
Actual                   117            3511
Prediction (s.d.)        107 (0.36)     3196 (10.93)
95% CI                   [106, 107]     [3174, 3217]

Absolute effect (s.d.)   11 (0.36)      316 (10.93)
95% CI                   [9.8, 11]      [294.6, 337]

Relative effect (s.d.)   9.9% (0.34%)   9.9% (0.34%)
95% CI                   [9.2%, 11%]    [9.2%, 11%]
```

Printing a summary report

```
summary(impact, "report")
```

```
Analysis report {CausalImpact}


During the post-intervention period, the response variable

Summing up the individual data points during the post-inter

The above results are given in terms of absolute numbers. I

This means that the positive effect observed during the int

The probability of obtaining this effect by chance is very

# Adjusting the model fit from a period including the
```