Prophet package

Lucas

30 novembre 2020

Overview and definition

What is Prophet ?

Prophet is an open source package developed by Facebook Data Science Team in order to resolve business problems related to time series forecasting. Prophet is a procedure to forecast time series based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. It works best with time series that have strong seasonal effects and several seasons of historical data. Prophet is robust to missing data and shifts in the trend, and typically handles outliers well.

Prophet is optimized for the business forecast tasks which typically have any of the following characteristics:

- hourly, daily, or weekly observations with at least a few months (preferably a year) of history strong multiple "human-scale" seasonalities: day of week and time of year
- important holidays that occur at irregular intervals that are known in advance (e.g. the Super Bowl)
- a reasonable number of missing observations or large outliers
- historical trend changes, for instance due to product launches or logging changes
- trends that are non-linear growth curves, where a trend hits a natural limit or saturates

Prophet's default settings can produce forecasts that are often accurate as those produced by skilled forecasters, with much less effort. With Prophet, you are not stuck with the results of a completely automatic procedure if the forecast is not satisfactory — an analyst with no training in time series methods can improve or tweak forecasts using a variety of easily-interpretable parameters. By combining automatic forecasting with analyst-in-the-loop forecasts for special cases, it is possible to cover a wide variety of business use-cases.

Therefore, the goal behind Prophet is to "make it easier for experts and non-experts to make high-quality forecasts that keep up with demand."

Real world Use caseof Prophet (some example)

- Prophet is used in many applications across Facebook for producing reliable forecasts for planning and goal setting
- forecast how many cars must be produce monthly to respond to the global demand
- forecast how many PS5 will be sold over years to prevent over producing

How to install Prophet

(follow the step on the CRAN) : https://cran.r-project.org/bin/windows/Rtools/

A simple model with Prophet

Before using Prophet you need to install it. It is a CRAN Package, so you can use install.packages()

```
#install.packages('prophet')
```

Now that the package is installed, let's have a quick overview of how to use prophet in R. We will use a csv file available here : https://github.com/facebook/prophet/blob/master/examples/example_wp_log_peyton_manning.csv

This csv contains how many persons saw Peyton Manning wikipedia page express in log.

First step is to import the package and the csv file.

The csv file must have specific columns names. This format must be respected otherwise prophet will not work.

- ds reflects the date (YYYY-MM-DD, or YYYY-MM-DD HH:MM:SS)
- y represents the output in this date (numerical values)

```
library(prophet)
```

```
## Loading required package: Rcpp
```

```
## Loading required package: rlang
```

```
df <- read.csv("peyton_wikipedia_page.csv")
head(df)
```

```
##           ds        y
## 1 2007-12-10 9.590761
## 2 2007-12-11 8.519590
## 3 2007-12-12 8.183677
## 4 2007-12-13 8.072467
## 5 2007-12-14 7.893572
## 6 2007-12-15 7.783641
```

We call the Prophet function and we pass our dataframe as the first argument. There are additional arguments that enable Prophet to fit the data. We will see some of them in the next part.

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

```
## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.
```

The make_future_dataframe function takes the model object and a number of periods to forecast. 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)
```

```
##              ds
## 3265 2017-01-14
## 3266 2017-01-15
## 3267 2017-01-16
## 3268 2017-01-17
## 3269 2017-01-18
## 3270 2017-01-19
```

We use the predict() function to forecast on the date that we built using make_future_dataframe. It takes 2 arguments the "m" our model and "future" created from the previous function. predict() returns a dataframe with different information. Here are some of them :

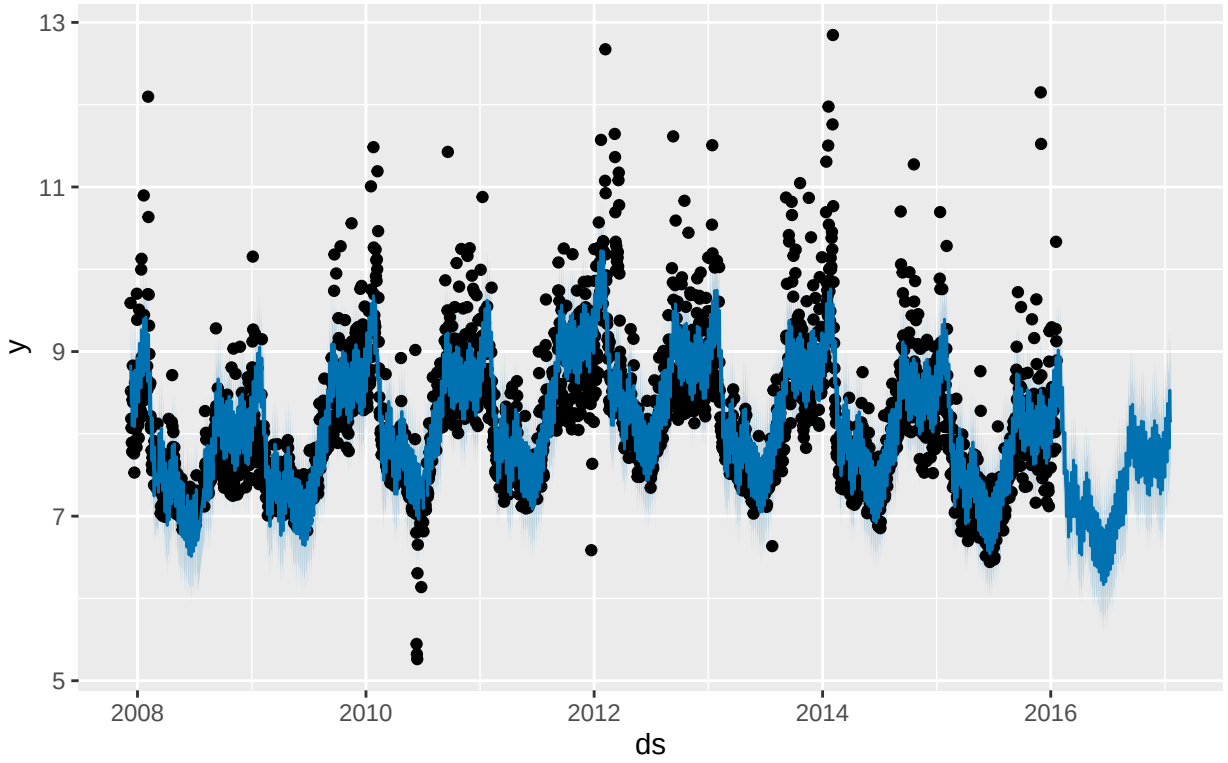
- yhat which is our forecast
- yhat_lower and yhat_upper which represent our uncertainty interval

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

```
##           ds     yhat yhat_lower yhat_upper
## 1 2007-12-10 8.841369   8.238833   9.443766
## 2 2007-12-11 8.589910   7.957658   9.157891
## 3 2007-12-12 8.385769   7.770256   8.997400
## 4 2007-12-13 8.363668   7.781949   8.977238
## 5 2007-12-14 8.351665   7.735816   9.001348
## 6 2007-12-15 8.096989   7.514233   8.713121
```

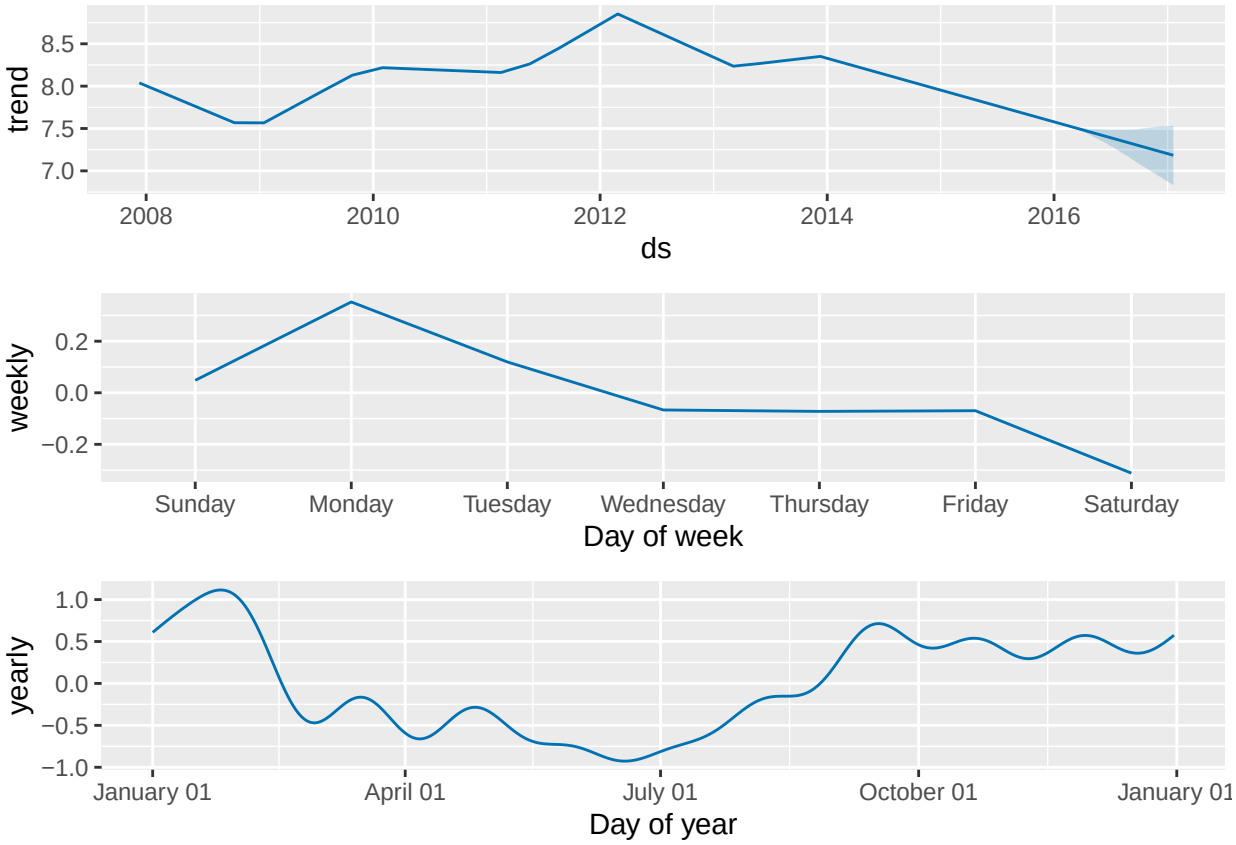
You can plot the forecast, by passing in the model and the forecast dataframe. You will have the 1st part of the plot that represents past data and the 2nd part which is the forecast with the uncertainty interval.

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



Finally,the Prophet_plot_components() break down the forecast into 3 components: the trend, the week seasonality, and the year seasonality. .

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



Intuition Behind Prophet

As said earlier, Prophet is a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. But how does Prophet work ? The Prophet equation relies the deconstruction of a time series into several components, each representing one of the underlying categories of patterns. Prophet is based on 3 mains components : trend, seasonality and holidays. In its simplest forms we can write the Prophet equation as : y(t) = g(t) + s(t) + h(t) + e(t) where :

- g(t) trend (i.e. growth over time)
- s(t) seasonality (i.e. weekly, monthly, yearly)
- h(t)ties in effects of holidays (on potentially irregular schedules >= 1 day(s))
- e(t)represents the errors

What Prophet does is "framing the forecasting problem as a curve-fitting exercise" rather than looking explicitly at the time based dependence of each observation.

Some Maths for a better understanding

Let's see in more details the Prophet equation. y(t) = g(t) + s(t) + h(t) + e(t)

We are going to see how the different parameters are modeling within Prophet.

We will use the example_wp_log_R.csv dataset to illustrate each function and parameter of Prophet.

example_wp_log_R.csv contains the informations of the number of people who have visited the wikipedia R page a specific day.

The trend g(t)

Prophet allows 2 types of modeling for the trend :

- piecewise linear model
- saturating growth model

Piecewise linear model

Linear trend model is a simple Piecewise Linear Model with a constant rate of growth. It is best suited for problems without a market cap or other max in sight. Modeling the linear trend is easily realized with Prophet.

g(t) = (k+a(t)T)t+(m+5T°()°

where:

- k is the growth rate
- has the rate adjustments
- m is the offset parameter

and, to make the function continuous, is set to:

By default, Prophet is set to do its forecast using a linear model.

We are first going to build the model using a linear model trend.

```
library(prophet)

df <- read.csv("log_wikipedia_R.csv")

#we specify growth = linear but you can also not specify (default setting)
m <- prophet(df, growth = "linear")
```

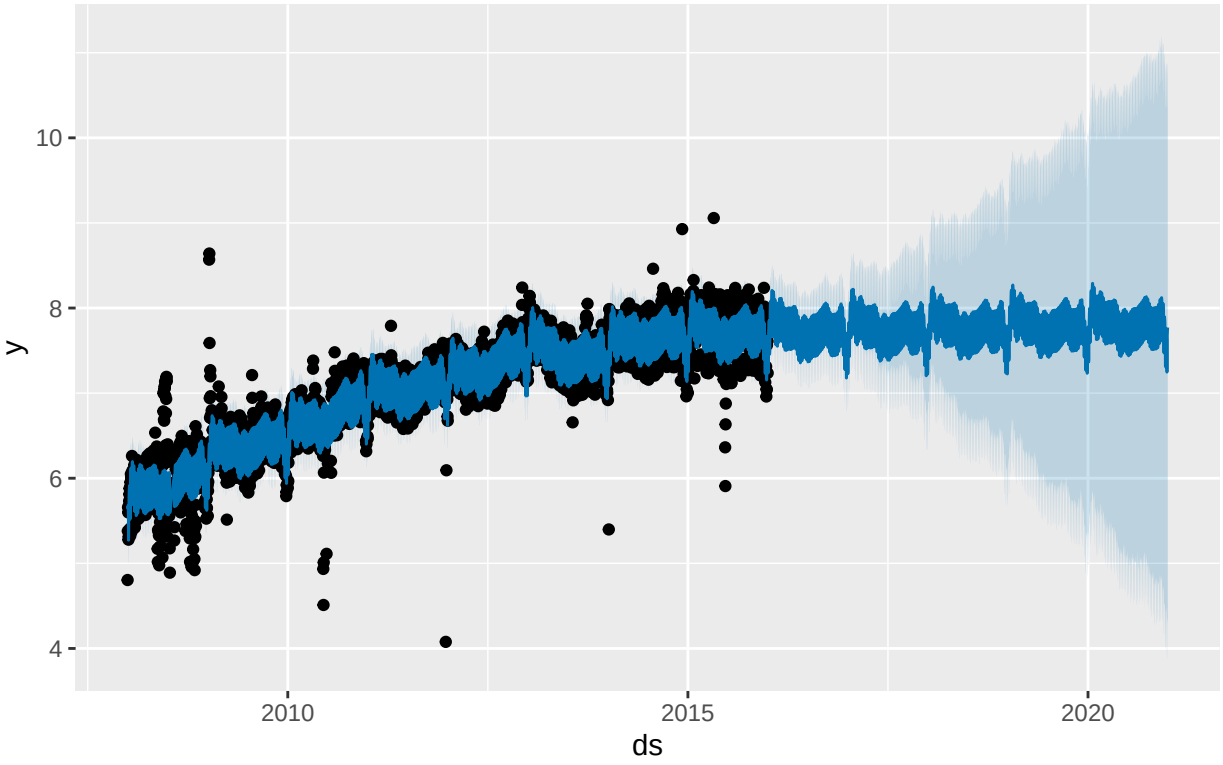
```
## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.
```

```
#we use the same process for the forecasting part

future <- make_future_dataframe(m, periods = 1826)

forecast <- predict(m,future)

plot(m,forecast)
```




```
#we will store our plot
linear <- plot(m,forecast)
```

Saturating growth model

If there is a saturation problem for example number of people with access to internet.

g(t) equation is : g(t)=C/(1+exp(-k(t-m)))

Where :

- C = caring capacity
- k = growth rate
- m = offset parameter

We can see from the previous plot that the forecast in terms of uncertainty is not so good.

When forecasting growth, there are usually some maximum achievable points: total market size, total population size, etc. This is called the carrying capacity, and the forecast should saturate at this point.

Let's modify our 1st model and precise the maximum capacity. The point at which the model should saturate.

The carrying capacity does not have to be constant. If the size of the market is increasing, the capacity can be increasing too.

```r
library(prophet)

df <- read.csv("log_wikipedia_R.csv")

#We must specify the carrying capacity in every row in the dataframe
#the carrying capacity is usually set using data expertise about the market size
df$cap <- 8.5

#now we specify growth to logistic
m <- prophet(df, growth = "logistic")
```

```
## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.
```

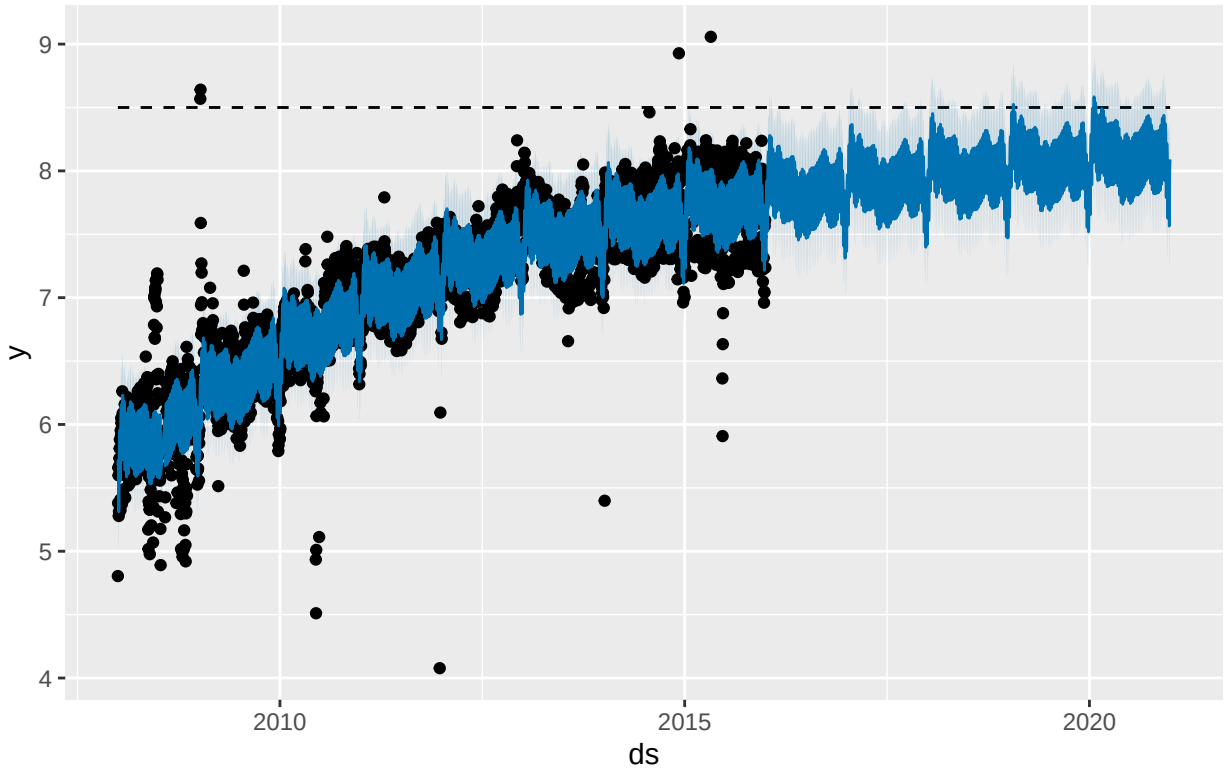
```r
#we use the same process for the forecasting part

future <- make_future_dataframe(m, periods = 1826)

#the capacity here is set to 8.5
future$cap <- 8.5

forecast <- predict(m,future)

plot(m,forecast)
```



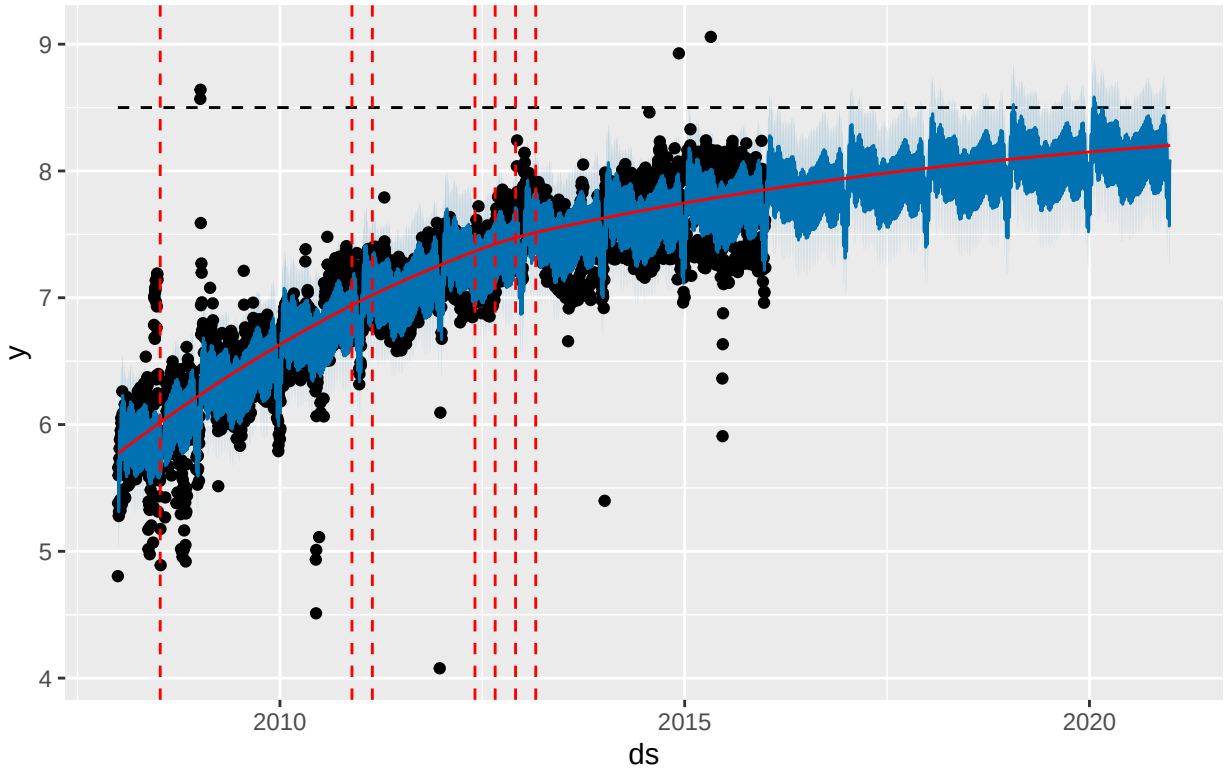

```r
#we will store our plot
growth <- plot(m,forecast)
```

Seasonality s(t)

The seasonal component s(t) provides an adaptability to the model by allowing periodic changes based on sub-daily, daily, weekly and yearly seasonality.

Prophet detects changepoints and allow the trend to adapt appropriately. The locations of the significant changepoints can be visualized with :

```
plot(m,forecast) + add_changepoints_to_plot(m)
```

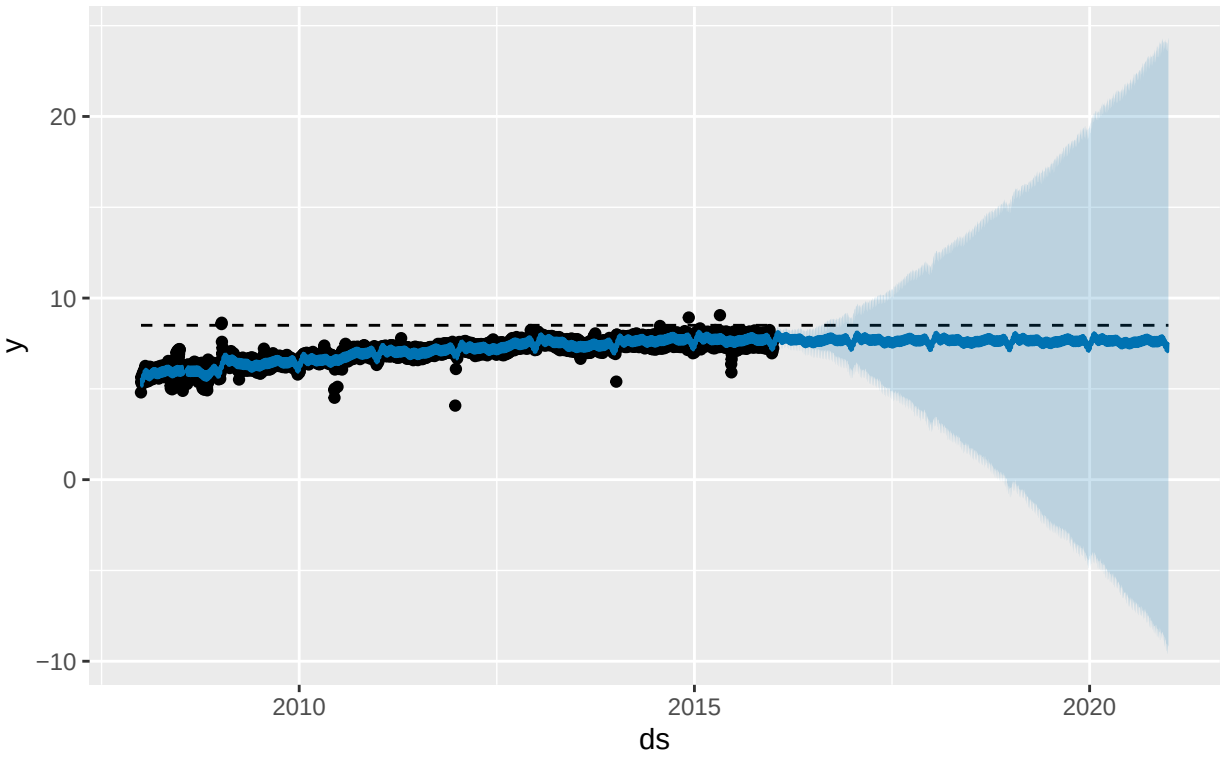


You can also adjust the strength of the sparse prior using the input argument changepoint_prior_scale. By default, this parameter is set to 0.05. Increasing it will make the trend more flexible: (Decreasing it will make the trend less flexible:)

```
m <- prophet(df, changepoint.prior.scale = 0.5)
```

```
## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.
```

```
forecast <- predict(m, future)
plot(m, forecast)
```



holydays and events h(t)

If there are specific events you want to model, there are multiple choices.

We will see 2 of them :

- create a dataframe that includes the date you want to include.
- Use the built-in holidays country-specific collection

```
#library(dplyr)

#playoffs <- data_frame(holiday = 'playoff',ds = as.Date(c('2008-01-13', '2009-01-03', '2010-01-16','20
#                  '2014-01-12', '2014-01-19',
#                  '2014-02-02', '2015-01-11', '2016-01-17',
#                  '2016-01-24', '2016-02-07')),
#  lower_window = 0,
#  upper_window = 1
#)

#holidays <- bind_rows(playoffs)

#library(prophet)
#df <- read.csv("peyton_wikipedia_page.csv")



#including the holidays we create previously
#m <- prophet(df,holidays =  holidays)
```

```
#m <- add_country_holidays(m, country_name =  'US')

#m <- fit.prophet(m, df)

#you can see the holiday included
#m$train.holiday.names



#we use the same process for the forecasting part

#forecast <- predict(m, future)
#prophet_plot_components(m, forecast)
```

Bibliography - usefull links - data source

- https://facebook.github.io/prophet/docs/quick_start.html#r-api
- https://facebook.github.io/prophet/
- https://medium.com/future-vision/intro-to-prophet-9d5b1cbd674e
- https://ledatascientist.com/facebook-prophet-la-prevision-a-grande-echelle/
- https://towardsdatascience.com/a-quick-start-of-time-series-forecasting-with-a-practical-example-using-fb-prophet-31c4447a2274
- https://mode.com/example-gallery/forecasting_prophet_r_cookbook/
- https://peerj.com/preprints/3190/
- https://research.fb.com/blog/2017/02/prophet-forecasting-at-scale/