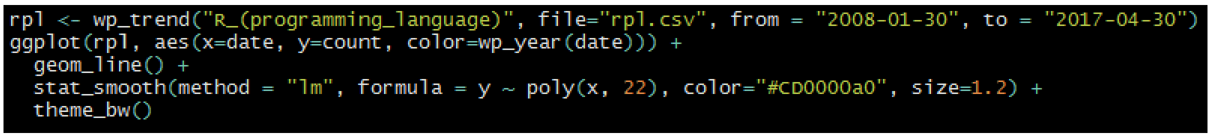
Time-series analysis in R – prophet package

https://research.fb.com/prophet-forecasting-at-scale/

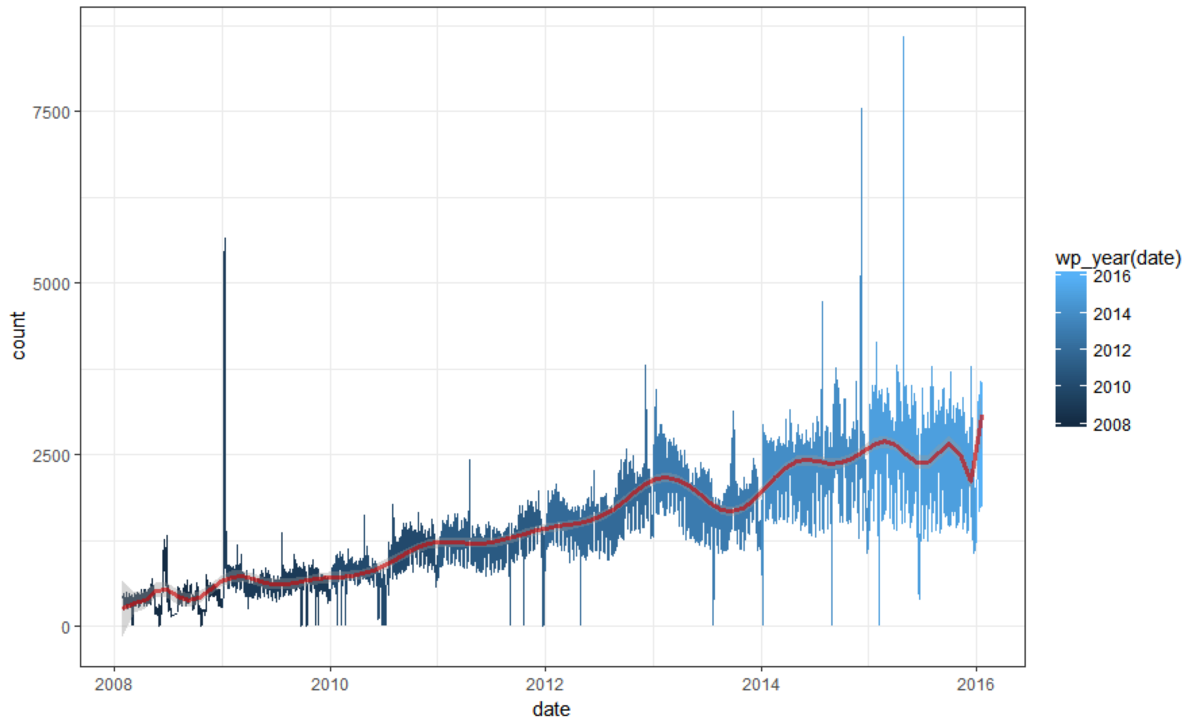
Prophet is an open source forecasting tool available in Python and R, which was release by Facebook in February 2017. The idea here is to use the 'wikipediatrend' package in R to scrap information about the number of visits a page has recorded over a period of time on its Wikipedia page, and do forecasting according to the trend generated.

The first thing here is to extract the information of a Wikipedia page. Here the page taken for analysis is the R (Programming Language) https://en.wikipedia.org/wiki/R_(programming_language)

Using the wp_trend() method of wikipediatrend package in R, data is collected from 2008-01-30 to 2017-04-30 and a trend graph is plotted using the data.

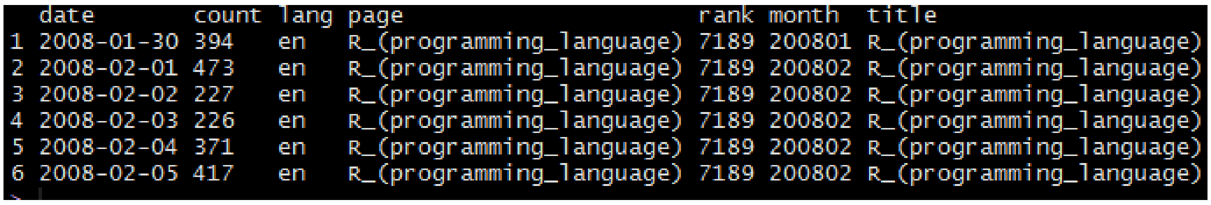


```
rpl <- wp_trend("R_(programming_language)", file="rpl.csv", from = "2008-01-30", to = "2017-04-30")
ggplot(rpl, aes(x=date, y=count, color=wp_year(date))) +
  geom_line() +
  stat_smooth(method = "lm", formula = y ~ poly(x, 22), color="#CD0000a0", size=1.2) +
  theme_bw()
```



As we can see from the plot generated, there was a spike in 2009 about search for R programming language on Wikipedia. After 2012, even though there have been some spikes in the number of counts, the smoothening curve shows that the search for R programming language has increased over time. Also, instead of smoothening the curve by using linear regression, polynomial regression is applied as it can be used to fir models of order n > 1to the data and modelling of non linear relationships can also be tried.

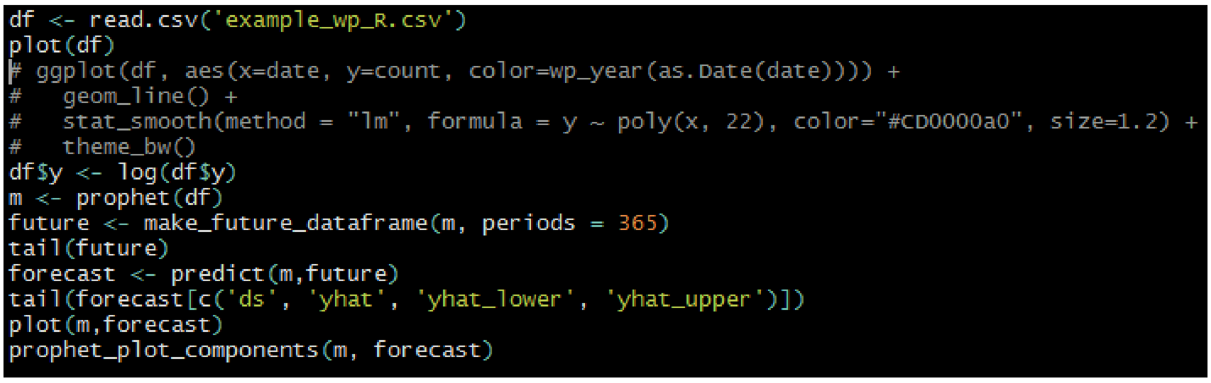
The incoming data extracted in json format from stats.grok.se is saved in a csv file. The cleaning of this csv file is done in excel as to extract the date attribute and the count attribute. The first five rows of the gathered csv files looks something like this:



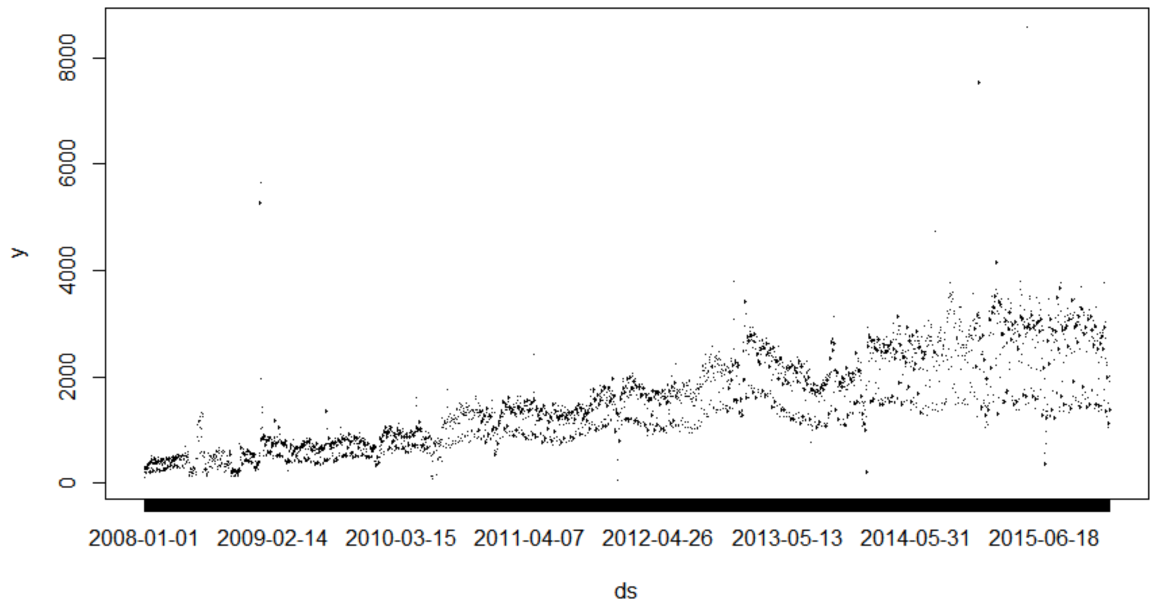
```
  date       count lang page                       rank month  title
1 2008-01-30 394   en   R_(programming_language) 7189 200801 R_(programming_language)
2 2008-02-01 473   en   R_(programming_language) 7189 200802 R_(programming_language)
3 2008-02-02 227   en   R_(programming_language) 7189 200802 R_(programming_language)
4 2008-02-03 226   en   R_(programming_language) 7189 200802 R_(programming_language)
5 2008-02-04 371   en   R_(programming_language) 7189 200802 R_(programming_language)
6 2008-02-05 417   en   R_(programming_language) 7189 200802 R_(programming_language)
```

The first two columns are extracted and saved as "example_wp_R.csv". Also, to check the efficiency of the forecast done by prophet, some latest records were removed to see how well the forecast fit with the actual trend.

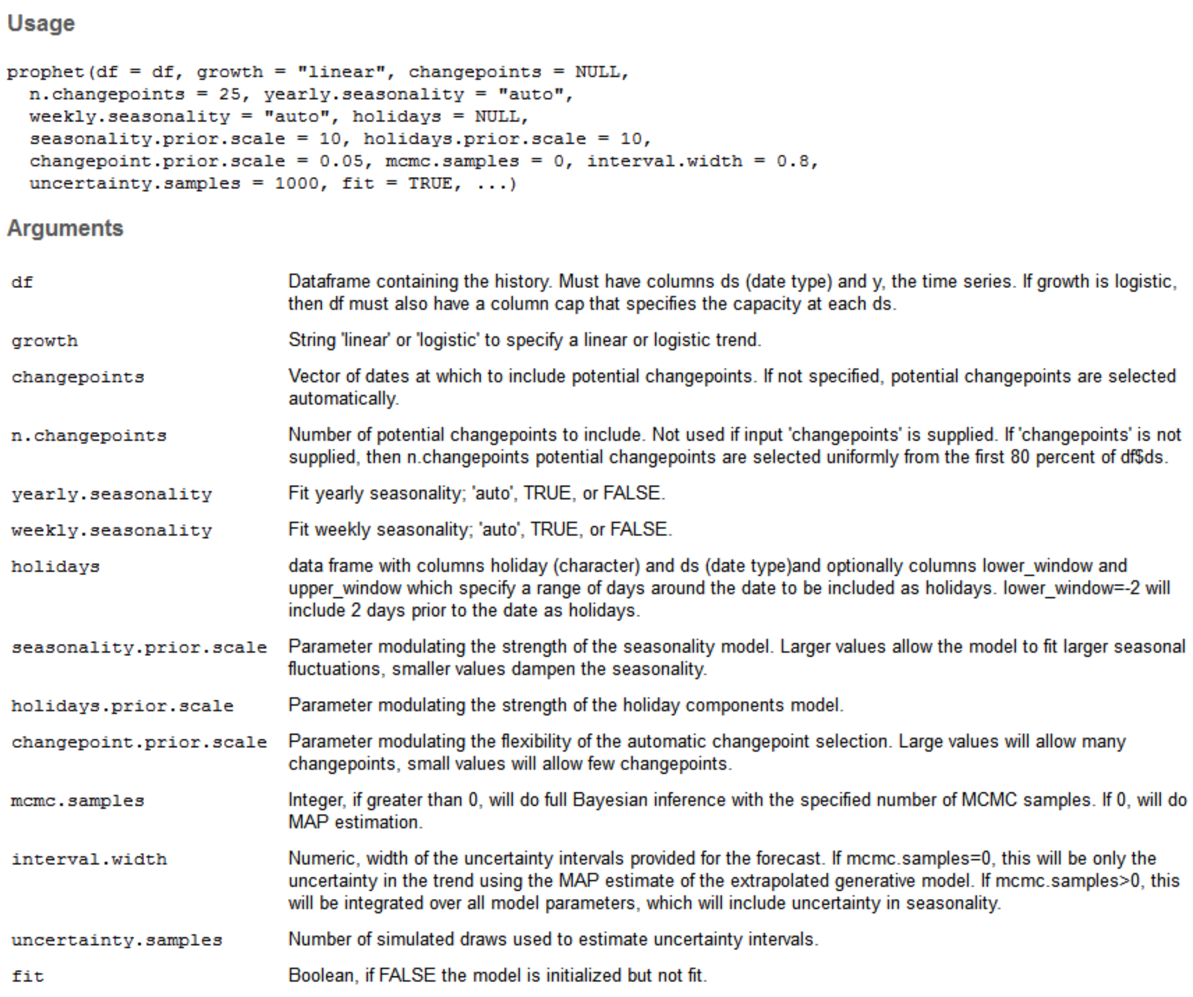
The file was loaded in R and a raw plot was created just to see, whether some processing is required in the dataset or not.



```
df <- read.csv('example_wp_R.csv')
plot(df)
# ggplot(df, aes(x=date, y=count, color=wp_year(as.Date(date)))) +
#   geom_line() +
#   stat_smooth(method = "lm", formula = y ~ poly(x, 22), color="#CD0000a0", size=1.2) +
#   theme_bw()
df$y <- log(df$y)
m <- prophet(df)
future <- make_future_dataframe(m, periods = 365)
tail(future)
forecast <- predict(m,future)
tail(forecast[c('ds', 'yhat', 'yhat_lower', 'yhat_upper')])
plot(m,forecast)
prophet_plot_components(m, forecast)
```



A log transformation is applied on the count attribute, which is renamed as y in this csv file. prophet() method is used on the dataset. The explanation of this method is given below:



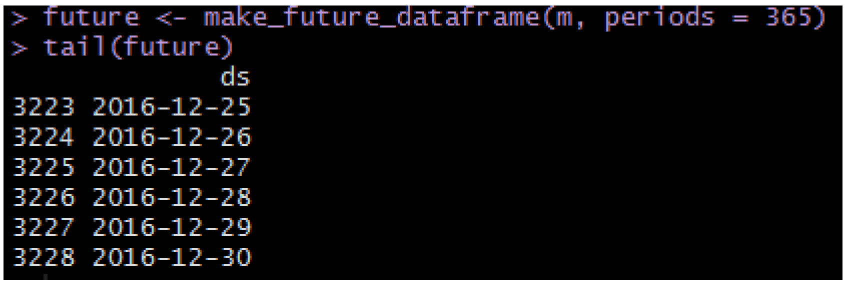
Usage

```
prophet(df = df, growth = "linear", changepoints = NULL,
  n.changepoints = 25, yearly.seasonality = "auto",
  weekly.seasonality = "auto", holidays = NULL,
  seasonality.prior.scale = 10, holidays.prior.scale = 10,
  changepoint.prior.scale = 0.05, mcmc.samples = 0, interval.width = 0.8,
  uncertainty.samples = 1000, fit = TRUE, ...)
```

Arguments

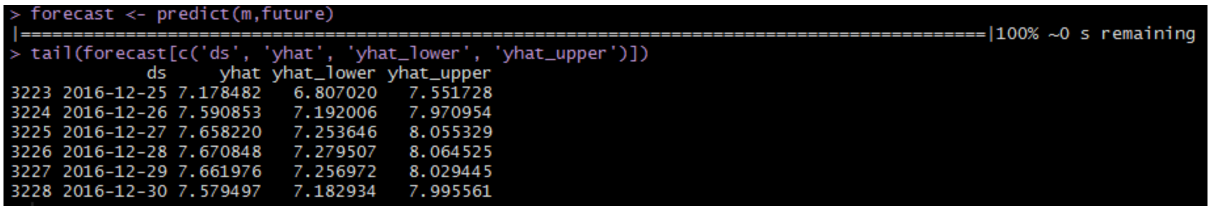
`df`	Dataframe containing the history. Must have columns ds (date type) and y, the time series. If growth is logistic, then df must also have a column cap that specifies the capacity at each ds.
`growth`	String 'linear' or 'logistic' to specify a linear or logistic trend.
`changepoints`	Vector of dates at which to include potential changepoints. If not specified, potential changepoints are selected automatically.
`n.changepoints`	Number of potential changepoints to include. Not used if input 'changepoints' is supplied. If 'changepoints' is not supplied, then n.changepoints potential changepoints are selected uniformly from the first 80 percent of df$ds.
`yearly.seasonality`	Fit yearly seasonality; 'auto', TRUE, or FALSE.
`weekly.seasonality`	Fit weekly seasonality; 'auto', TRUE, or FALSE.
`holidays`	data frame with columns holiday (character) and ds (date type)and optionally columns lower_window and upper_window which specify a range of days around the date to be included as holidays. lower_window=-2 will include 2 days prior to the date as holidays.
`seasonality.prior.scale`	Parameter modulating the strength of the seasonality model. Larger values allow the model to fit larger seasonal fluctuations, smaller values dampen the seasonality.
`holidays.prior.scale`	Parameter modulating the strength of the holiday components model.
`changepoint.prior.scale`	Parameter modulating the flexibility of the automatic changepoint selection. Large values will allow many changepoints, small values will allow few changepoints.
`mcmc.samples`	Integer, if greater than 0, will do full Bayesian inference with the specified number of MCMC samples. If 0, will do MAP estimation.
`interval.width`	Numeric, width of the uncertainty intervals provided for the forecast. If mcmc.samples=0, this will be only the uncertainty in the trend using the MAP estimate of the extrapolated generative model. If mcmc.samples>0, this will be integrated over all model parameters, which will include uncertainty in seasonality.
`uncertainty.samples`	Number of simulated draws used to estimate uncertainty intervals.
`fit`	Boolean, if FALSE the model is initialized but not fit.

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



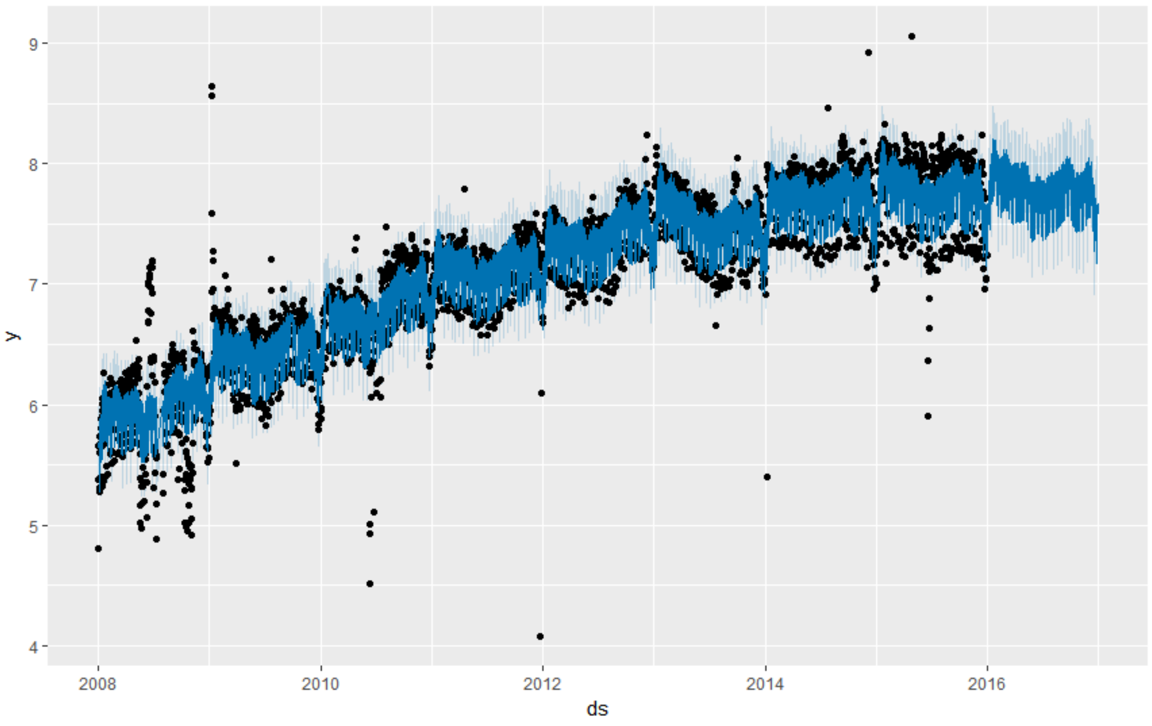
```
> future <- make_future_dataframe(m, periods = 365)
> tail(future)
             ds
3223 2016-12-25
3224 2016-12-26
3225 2016-12-27
3226 2016-12-28
3227 2016-12-29
3228 2016-12-30
```

As with most modelling procedures in R, the generic predict() function is used 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.

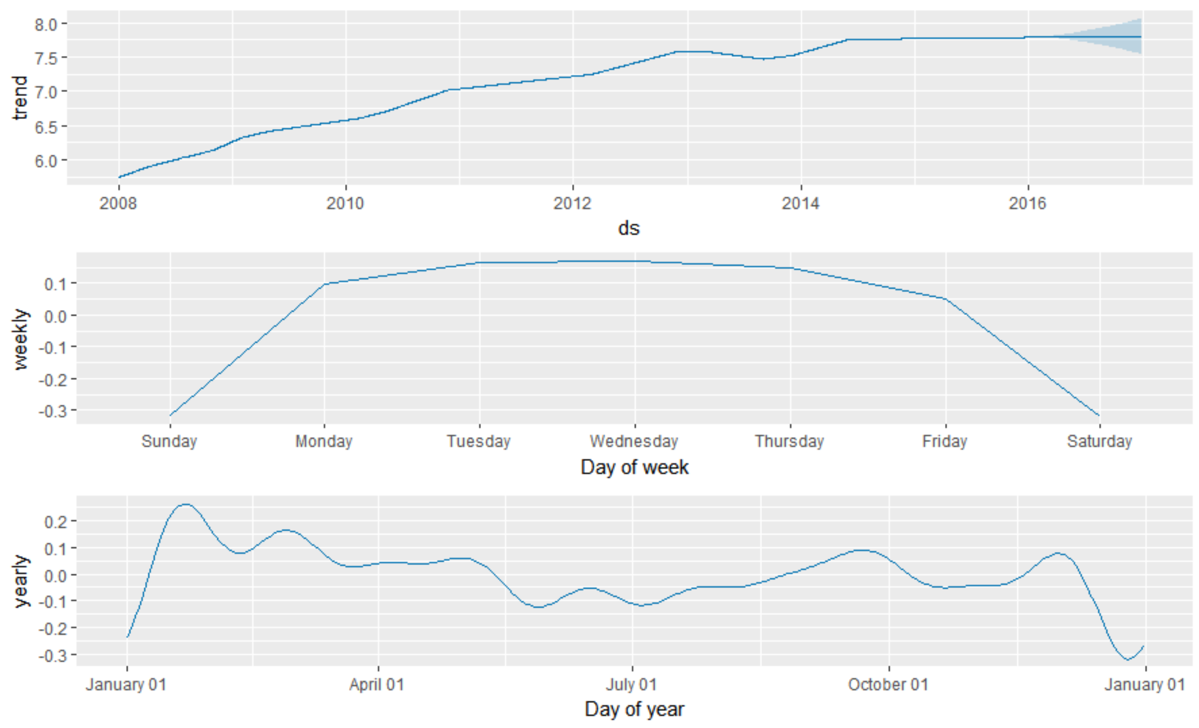


```
> forecast <- predict(m,future)
|==============================================================================================|100% ~0 s remaining
> tail(forecast[c('ds', 'yhat', 'yhat_lower', 'yhat_upper')])
             ds     yhat yhat_lower yhat_upper
3223 2016-12-25 7.178482   6.807020   7.551728
3224 2016-12-26 7.590853   7.192006   7.970954
3225 2016-12-27 7.658220   7.253646   8.055329
3226 2016-12-28 7.670848   7.279507   8.064525
3227 2016-12-29 7.661976   7.256972   8.029445
3228 2016-12-30 7.579497   7.182934   7.995561
```

The generic plot function is used to plot the forecast.

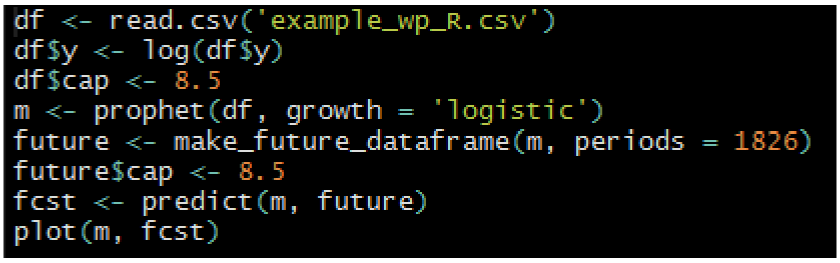


Also, the prophet_plot_component function can be used to see the forecast broken down into trend, weekly seasonality, and yearly seasonality.



By default, Prophet uses a linear model for its forecast, When forecasting growth, there is usually some maximum achievable point: total market size, total population size, etc. This is called the carrying capacity, and the forecast should saturate at this point.

Prophet can make forecast using logistic growth trend model, with a specified carrying capacity.

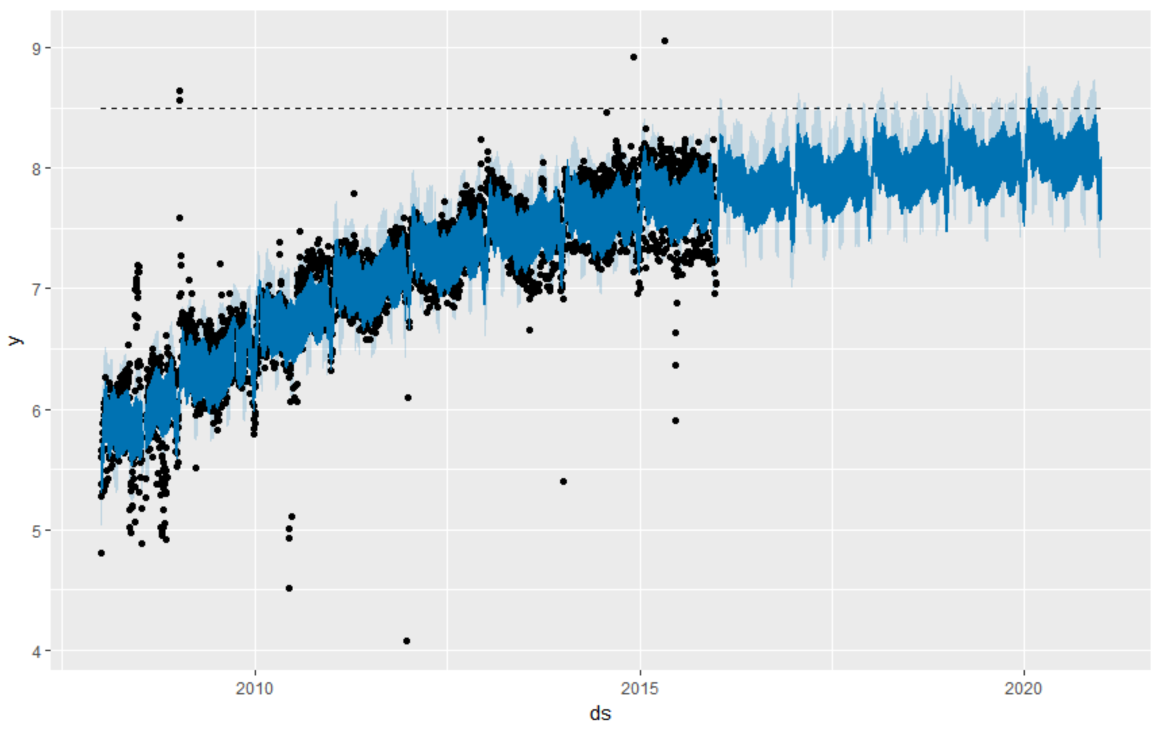


```
df <- read.csv('example_wp_R.csv')
df$y <- log(df$y)
df$cap <- 8.5
m <- prophet(df, growth = 'logistic')
future <- make_future_dataframe(m, periods = 1826)
future$cap <- 8.5
fcst <- predict(m, future)
plot(m, fcst)
```

The important thing to note here is that cap must be specified for every row in the dataframe, and that it does not have to be constant. If the market size is growing, then cap can be an increasing sequence.

The model is fitted as before, except passing in an additional argument to specify logistic growth.

A dataframe for future predictions is made as before, except the capacity in the future must also be specified. The capacity is kept constant at the same values in the history, and forecast 3 years into the future:



Note: extra analysis on how to handle outliers given in script