Prophet_Exploration

paul

March 6, 2017

Contents

References:

- github Source Code
- Quick Start In R
- Wikipediatrend Vignette
  - used to scrape data from wikipedia pages.
- Prophet Function Documentation

Notes about this Package:

- This package is used to forcast data and is used by facebook to account for seasonal and holiday affects on the data.
- Always take a data frame with 2 columns:
  - *ds:* a date/datetime column indicating time.
  - *y:* a column for the data to be forcasted. **Must be Numeric Values**
    - At least a year worth of data.
- Returns a model object that can be processed using *predict* and *plot*.
- The *predict* function defaults to predicting a linear trend. When trying to log transform the y column, special attention needs to be taken to modify values of *0*.

General Steps:

1: Use *prophet* function to fit the historic data. 2: Use *make_future_dataframe* function to make dataframe with future dates for forecasting. 3: Forcast using *predict* function with both the historic and future data as parameters. 4: Call *plot* to plot the forcast (the historic data and the forecast)

Working Through Tutorial Online:

- ***Goal: Forecasting the time series of daily page views for the Wikipedia page for Peyton Manning.***_

```
## Due to limited understanding of statics, all zeros are removed from the data set.

df = read.csv("./PeytonManningData.csv")%>% select(date:count)%>%
filter(count>0)%>%mutate(count=log(count))

##Beware, The names of the data frame column needs to be ds and y.
colnames(df)= c("ds","y")
```

```
m=prophet(df)
```

```
## STAN OPTIMIZATION COMMAND (LBFGS)
## init = user
## save_iterations = 1
## init_alpha = 0.001
## tol_obj = 1e-012
## tol_grad = 1e-008
## tol_param = 1e-008
## tol_rel_obj = 10000
## tol_rel_grad = 1e+007
## history_size = 5
## seed = 511924679
## initial log joint probability = -8.52316
## Optimization terminated normally:
##   Convergence detected: relative gradient magnitude is below tolerance
```

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

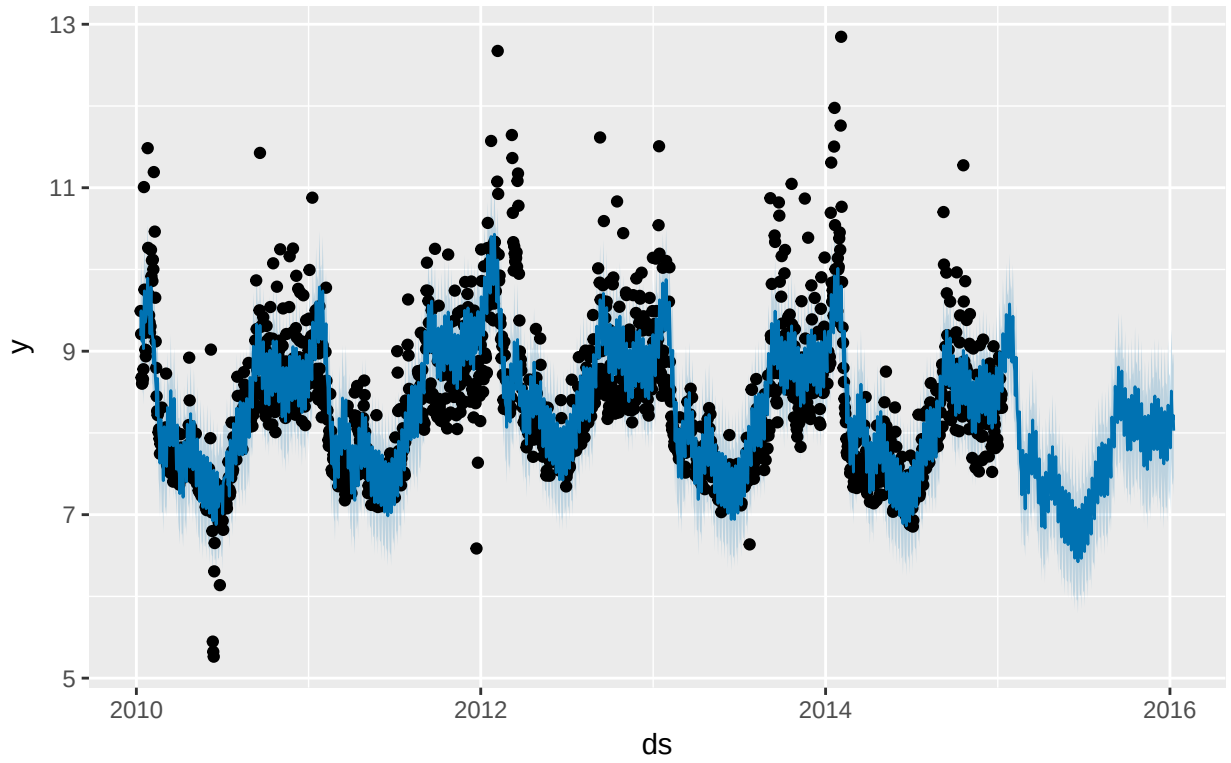
```
##             ds
## 2167 2016-01-04
## 2168 2016-01-05
## 2169 2016-01-06
## 2170 2016-01-07
## 2171 2016-01-08
## 2172 2016-01-09
```

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

#yhat contains the predicated information.
tail(forecast[c('ds', 'yhat', 'yhat_lower', 'yhat_upper')])
```

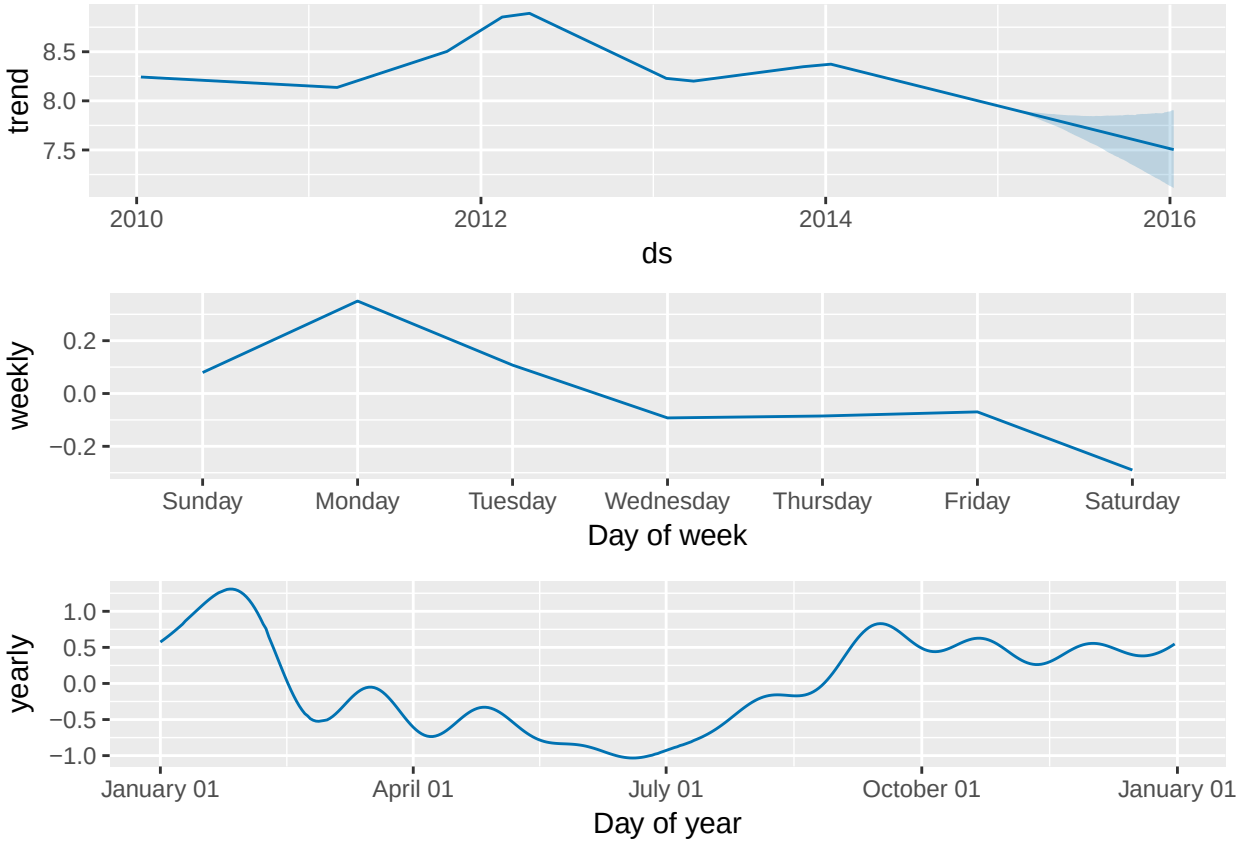
```
##             ds     yhat yhat_lower yhat_upper
## 2167 2016-01-04 8.512442   7.690695   9.266893
## 2168 2016-01-05 8.299222   7.506934   9.085707
## 2169 2016-01-06 8.129643   7.379897   8.818478
## 2170 2016-01-07 8.168003   7.363703   8.960132
## 2171 2016-01-08 8.215535   7.452870   8.959295
## 2172 2016-01-09 8.027930   7.221805   8.833867
```

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




```
## more detailed plot outlining where the data is broken down
## into: trend, weekly seasonality, and yearly seasonality

prophet_plot_components(m, forecast)
```



Comparing To Actual Data:

```
## Last 365 data point of forecast. The date for these data is needed
## to request the corresponding information from wikipedia
predicted_Data = tail(forecast[c("ds","yhat","yhat_lower","yhat_upper")],n=365)
start_Date = tail(forecast[["ds"]],n=365)[[1]]
end_Date = tail(forecast[["ds"]],n=365)[[365]]

# data = wikipediatrend:: wp_trend(page="Peyton_Manning",
# from = start_Date, to=end_Date)
# ##writing data:
#  if(!file.exists("actualData.csv"))
#  {
#      file.create("actualData.csv")
#  }
# write.table(data, "./actualData.csv",sep=",")

actual_Data = read.csv("./actualData.csv") %>% select(date:count)%>%
    filter(count>0)%>%mutate(count=log(count))
```

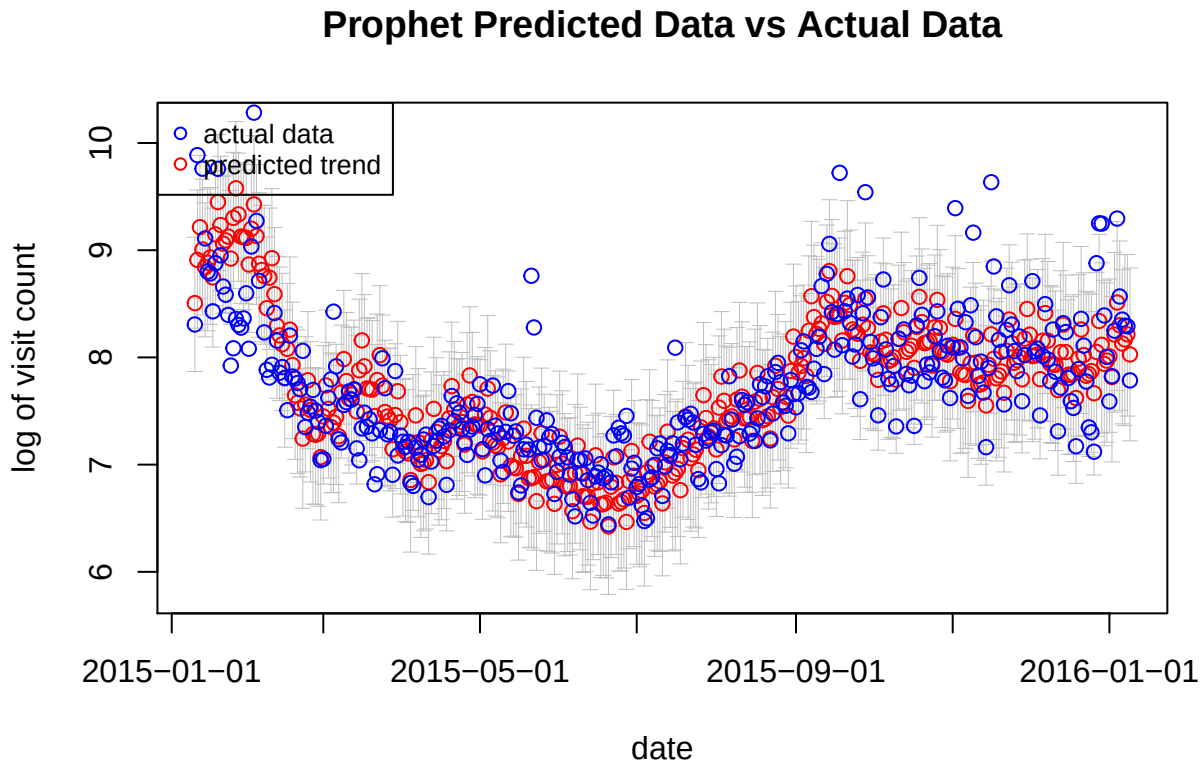
```
# plot(x=actual_Data[["date"]], y=actual_Data[["count"]],xlab="date",
# ylab="log of visit count",col="blue",main = "Prophet Trained Forcast
# vs Actual Data",type="l")
Hmisc::errbar(x=predicted_Data[["ds"]],y=predicted_Data[["yhat"]],
```

```
                    yplus=predicted_Data[["yhat_upper"]],yminus = predicted_Data[["yhat_lower"]],
                    lwd=0.1,pch=NA,errbar.col="gray",xlab="date",ylab="log of visit count",xaxt="n")

axis.Date(1, x=predicted_Data[["ds"]],format="%Y-%m-%d")
    points(x=predicted_Data[["ds"]],y=predicted_Data[["yhat"]], col="red")
    points(x=as.Date(actual_Data[["date"]],format="%Y-%m-%d"),
           y=actual_Data[["count"]],col="blue")

    title(main="Prophet Predicted Data vs Actual Data")

    legend("topleft",legend=c("actual data","predicted trend"),
           col=c("blue","red"),pch=1,cex=0.8)
```



Facebook Prophet Mathmatics:

Facebook Forcasting At Scale