Web Traffic Time Series Forecasting

Xin Kang

2019-02-26

Introduction

This project focuses on solving the problem of predicting the future web traffic for approximately 145,000 Wikipedia articles. Detailed data description is covered in the following section. Making future prediction on sequential or temporal observations has emerged in many key real-world problems. By forecasting the future values of multiple web traffic time series, we can answer some questions like how many severs you need in reality and what your total cost for next month is when you need to use external severs. If the performance is satisfactory, similar methods can be applied to other websites to predict their web traffic, and it can help people make smart advertisement decisions and make profit.

Data Description

Available training dataset consists of approximately 145k time series. Each of these time series represents a number of daily views of a different Wikipedia article, starting from July, 1st, 2015 up until June 30th, 2017. And the test dataset consists of times series ranging from July 1st, 2017 to September 10th, 2017. There are different types of traffic. For each time series, we are provided the name of the article as well as the type of traffic that this time series represent (all, mobile, desktop, spider). Unfortunately, the data source for this dataset does not distinguish between traffic values of zero and missing values. A missing value may mean the traffic was zero or that the data is not available for that day.

data.csv - contains traffic data. This is a csv file where each row corresponds to a particular article and each column correspond to a particular date. Some entries are missing data. The page names contain the Wikipedia project (e.g. en.wikipedia.org), type of access (e.g. desktop) and type of agent (e.g. spider). In other words, each article name has the following format: 'name_project_access_agent' (e.g. 'AKB48_zh.wikipedia.org_all-access_spider'). This data file contains times serises starting from July 1st, 2015 to September 10th, 2017, we need to divide it into training set and test set as indicated above.

key.csv - gives the mapping between the page names and the shortened Id column used for prediction.

Exploratory Data Analysis

Load Library and Data

```r
library(readr)
library(plyr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(tidyr)
library(data.table)
```

```
##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```r
library(tibble)
library(stringr)
library(ggplot2)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:data.table':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday,
##     week, yday, year

## The following object is masked from 'package:plyr':
##
##     here

## The following object is masked from 'package:base':
##
##     date
```

```r
library(reshape2)
```

```
##
## Attaching package: 'reshape2'

## The following objects are masked from 'package:data.table':
##
##     dcast, melt

## The following object is masked from 'package:tidyr':
##
##     smiths
```

```r
key = read_csv("web-traffic-time-series-forecasting/key.csv", n_max = 100)
```

```
## Parsed with column specification:
## cols(
##   Page = col_character(),
##   Id = col_character()
## )
```

```
data = read_csv("web-traffic-time-series-forecasting/data.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_integer(),
##   Page = col_character()
## )

## See spec(...) for full column specifications.
```

```
glimpse(key)
```

```
## Observations: 100
## Variables: 2
## $ Page <chr> "007_\u30b9\u30da\u30af\u30bf\u30fc_ja.wikipedia.org_all-...
## $ Id   <chr> "0b293039387a", "7114389dd824", "057b02ff1f09", "bd2aca21...
```

```
head(key)
```

```
## # A tibble: 6 x 2
##   Page                                                         Id
##   <chr>                                                        <chr>
## 1 "007_\u30b9\u30da\u30af\u30bf\u30fc_ja.wikipedia.org_all-acce~ 0b2930393~
## 2 "007_\u30b9\u30da\u30af\u30bf\u30fc_ja.wikipedia.org_all-acce~ 7114389dd~
## 3 "007_\u30b9\u30da\u30af\u30bf\u30fc_ja.wikipedia.org_all-acce~ 057b02ff1~
## 4 "007_\u30b9\u30da\u30af\u30bf\u30fc_ja.wikipedia.org_all-acce~ bd2aca21c~
## 5 "007_\u30b9\u30da\u30af\u30bf\u30fc_ja.wikipedia.org_all-acce~ c0effb42c~
## 6 "007_\u30b9\u30da\u30af\u30bf\u30fc_ja.wikipedia.org_all-acce~ 4ccd369ad~
```

```
dim(data)
```

```
## [1] 145063    804
```

```
select(head(data,10), 1:5, 800:804)
```

```
## # A tibble: 10 x 10
##    Page  `2015-07-01` `2015-07-02` `2015-07-03` `2015-07-04` `2017-09-06`
##    <chr>        <int>        <int>        <int>        <int>        <int>
##  1 2NE1~           18           11            5           13           27
##  2 2PM_~           11           14           15           18           25
##  3 3C_z~            1            0            1            1            7
##  4 4min~           35           13           10           94           16
##  5 52_H~           NA           NA           NA           NA           23
##  6 5566~           12            7            4            5           20
##  7 91Da~           NA           NA           NA           NA           10
##  8 A'N'~          118           26           30           24           44
##  9 AKB4~            5           23           14           12           44
## 10 ASCI~            6            3            5           12           32
## # ... with 4 more variables: `2017-09-07` <int>, `2017-09-08` <int>,
## #   `2017-09-09` <int>, `2017-09-10` <int>
```

```
select(tail(data,10), 1:5, 800:804)
```

```
## # A tibble: 10 x 10
##    Page  `2015-07-01` `2015-07-02` `2015-07-03` `2015-07-04` `2017-09-06`
##    <chr>        <int>        <int>        <int>        <int>        <int>
##  1 "Dra~           NA           NA           NA           NA            2
##  2 "Ska~           NA           NA           NA           NA            4
##  3 "Leg~           NA           NA           NA           NA            5
##  4 "Dob~           NA           NA           NA           NA           19
##  5 "Mi_~           NA           NA           NA           NA            8
##  6 "Und~           NA           NA           NA           NA            2
##  7 "Res~           NA           NA           NA           NA            5
##  8 "Ena~           NA           NA           NA           NA           13
##  9 "Has~           NA           NA           NA           NA            8
## 10 "Fra~           NA           NA           NA           NA            2
## # ... with 4 more variables: `2017-09-07` <int>, `2017-09-08` <int>,
## #   `2017-09-09` <int>, `2017-09-10` <int>
```

```
sum(is.na(data)) / (nrow(data) * ncol(data))
```

```
## [1] 0.06025302
```

```
head(data$Page, 10)
```

```
##  [1] "2NE1_zh.wikipedia.org_all-access_spider"
##  [2] "2PM_zh.wikipedia.org_all-access_spider"
##  [3] "3C_zh.wikipedia.org_all-access_spider"
##  [4] "4minute_zh.wikipedia.org_all-access_spider"
##  [5] "52_Hz_I_Love_You_zh.wikipedia.org_all-access_spider"
##  [6] "5566_zh.wikipedia.org_all-access_spider"
##  [7] "91Days_zh.wikipedia.org_all-access_spider"
##  [8] "A'N'D_zh.wikipedia.org_all-access_spider"
##  [9] "AKB48_zh.wikipedia.org_all-access_spider"
## [10] "ASCII_zh.wikipedia.org_all-access_spider"
```

Since key.csv is about 770 MB, I only load the first 100 rows to see its structure. The dimension of training set is 145063 * 804, which means it contains 145063 articles and 804 days. Let's show the first ten rows, the first five columns, and the last five columns. We can see there are many missing values in early dates, and the total is 6% missing values in the data, so we need to deal with these missing values before fitting models. #Data Transformation Since the page names contain the Wikipedia project (e.g. en.wikipedia.org), type of access (e.g. desktop) and type of agent (e.g. spider), which may influence the web traffic of articles, it is better to divide names into four separate parts.

During this process, I discover there are three types of project including wikipedia, wikimedia and mediawiki, so I need to deal with these three types separately.

```
data = rownames_to_column(data)
wikipedia = filter(data, str_detect(data$Page, "wikipedia.org")) %>% select(rowname, Page)
nrow(wikipedia)
```

```
## [1] 127208
```

```
wikipedia = wikipedia %>% separate(Page, into = c("first","second"), sep=".wikipedia.org_") %>%
  separate(first, c("name", "project"), sep=-3) %>% separate(second, c("access", "agent"), sep = "_") %
  mutate(project = str_sub(project, 2, 3))
wikipedia[1,]
```

```
## # A tibble: 1 x 5
##   rowname name  project access     agent
##   <chr>   <chr> <chr>   <chr>      <chr>
## 1 1       2NE1  zh      all-access spider
```

```
wikimedia = filter(data, str_detect(data$Page, "wikimedia.org")) %>% select(rowname, Page)
nrow(wikimedia)
```

```
## [1] 10555
```

```r
wikimedia = wikimedia %>% separate(Page, into = c("name", "second"), sep=".commons.wikimedia.org_") %>%
  separate(second, c("access", "agent"), sep = "_") %>% mutate(project = "wikimedia")
wikimedia[1,]
```

```
## # A tibble: 1 x 5
##   rowname name     access     agent  project
##   <chr>   <chr>    <chr>      <chr>  <chr>
## 1 13333   Accueil all-access spider wikimedia
```

```r
mediawiki = filter(data, str_detect(data$Page, "mediawiki.org")) %>% select(rowname, Page)
nrow(mediawiki)
```

```
## [1] 7300
```

```r
mediawiki = mediawiki %>% separate(Page, into = c("name", "second"), sep=".www.mediawiki.org_") %>%
  separate(second, c("access", "agent"), sep = "_") %>% mutate(project = "mediawiki")
mediawiki[1,]
```

```
## # A tibble: 1 x 5
##   rowname name                                  access     agent     project
##   <chr>   <chr>                                 <chr>      <chr>     <chr>
## 1 19612   "\"Keep_me_logged_in\"_extended_to_~ all-access all-age~ mediawi~
```

```r
nrow(mediawiki) + nrow(wikipedia) + nrow(wikimedia) == nrow(data)
```

```
## [1] TRUE
```

```r
Pages = full_join(wikipedia, wikimedia,  by = c("rowname", "name", "project", "access", "agent")) %>%
  full_join(mediawiki,  by = c("rowname", "name", "project", "access", "agent"))
head(Pages)
```

```
## # A tibble: 6 x 5
##   rowname name             project access     agent
##   <chr>   <chr>            <chr>   <chr>      <chr>
## 1 1       2NE1             zh      all-access spider
## 2 2       2PM              zh      all-access spider
## 3 3       3C               zh      all-access spider
## 4 4       4minute          zh      all-access spider
## 5 5       52_Hz_I_Love_You zh      all-access spider
## 6 6       5566             zh      all-access spider
```

Data Exploration and Visualization

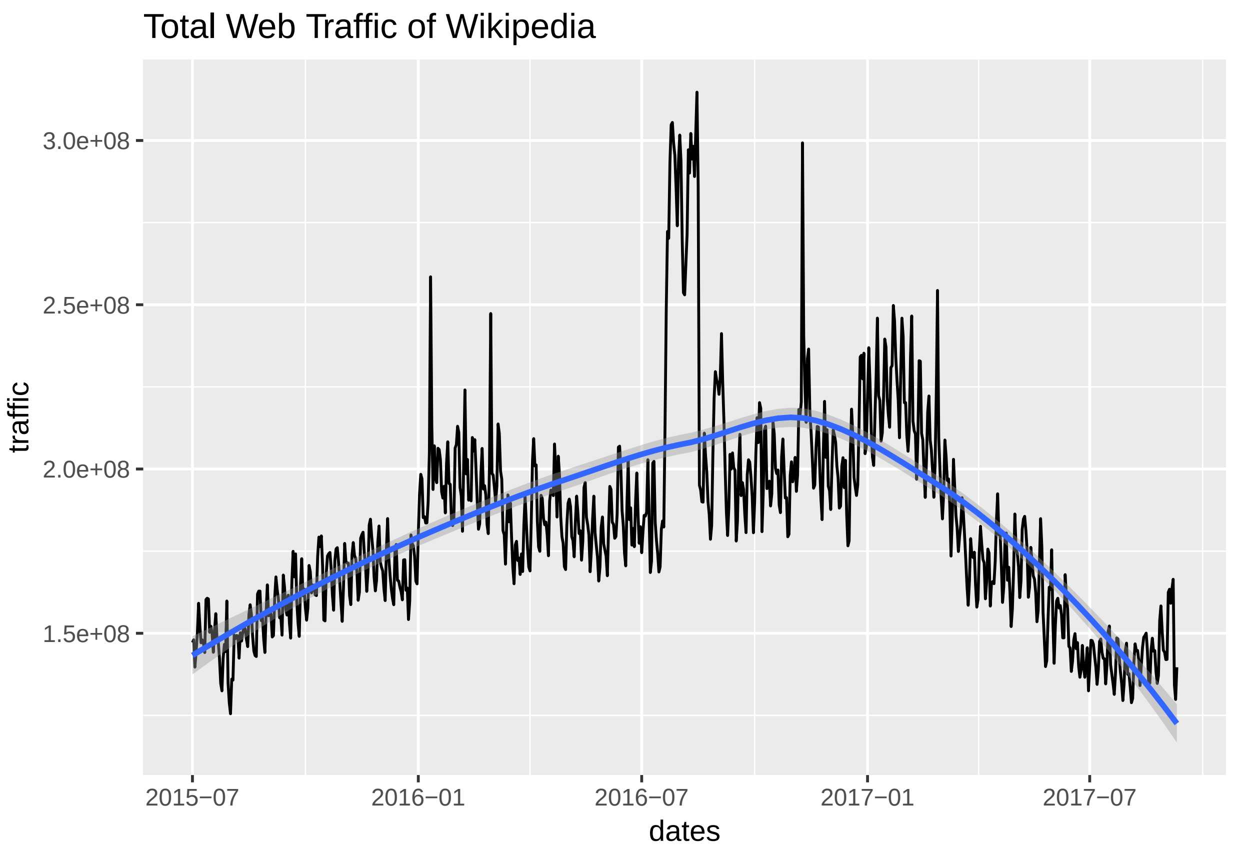
```r
temp = data %>% filter(str_detect(Page, "wikipedia")) %>% select(-c(rowname, Page))
wikipediaTotal = as.data.frame(t(sapply(temp, margin = 2, sum, na.rm = TRUE)))
temp = data %>% filter(str_detect(Page, "wikimedia")) %>% select(-c(rowname, Page))
wikimediaTotal = as.data.frame(t(sapply(temp, margin = 2, sum, na.rm = TRUE)))
temp = data %>% filter(str_detect(Page, "mediawiki")) %>% select(-c(rowname, Page))
mediawikiTotal = as.data.frame(t(sapply(temp, margin = 2, sum, na.rm = TRUE)))
```

Now we have respective total web traffic of wikipedia, wikimedia and mediawiki on every day, and want to detect their trends and compare them.

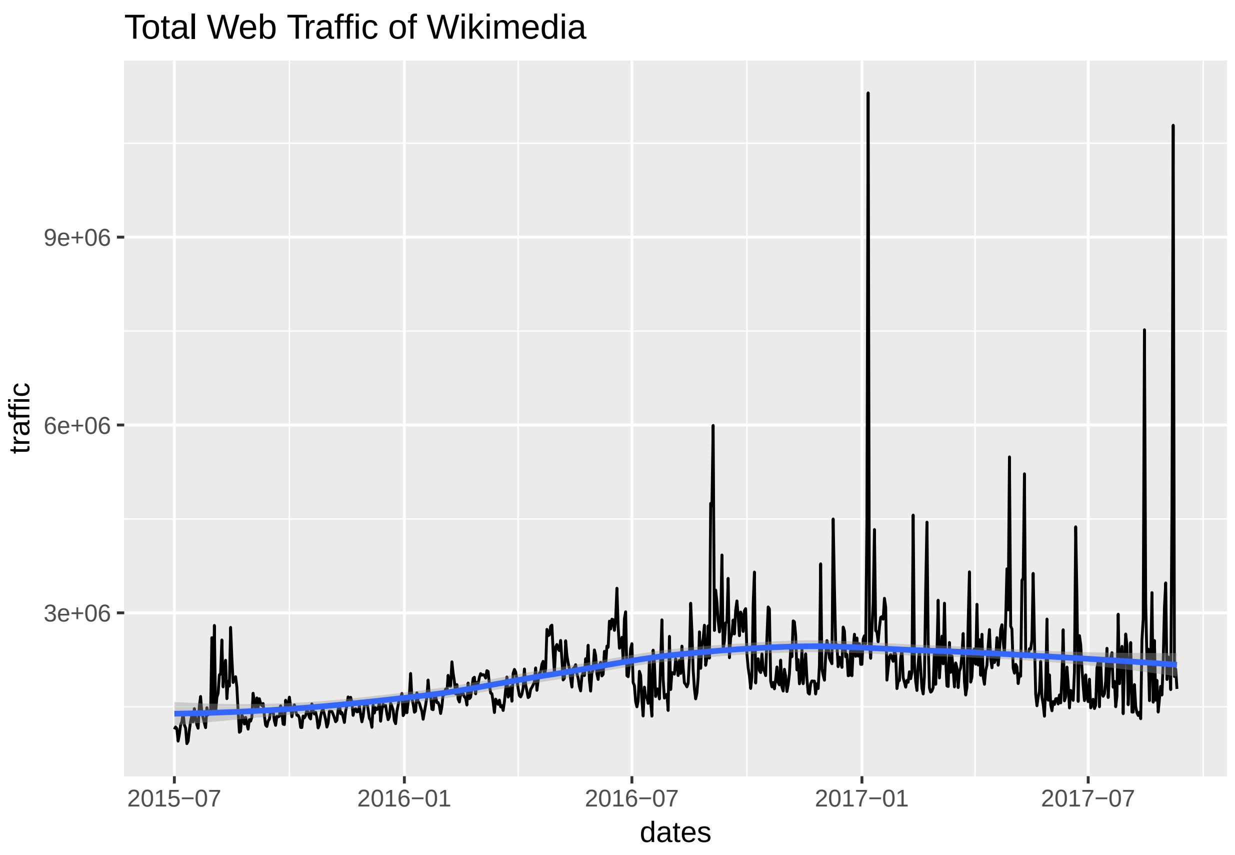
```r
wikipediaTotal = wikipediaTotal %>% t() %>% as.data.frame %>% rownames_to_column %>%
  rename(dates = rowname, traffic = V1) %>% mutate(dates = as.Date(dates))
```

```
## Warning in strptime(xx, f <- "%Y-%m-%d", tz = "GMT"): unknown timezone
## 'zone/tz/2018i.1.0/zoneinfo/Asia/Shanghai'
```

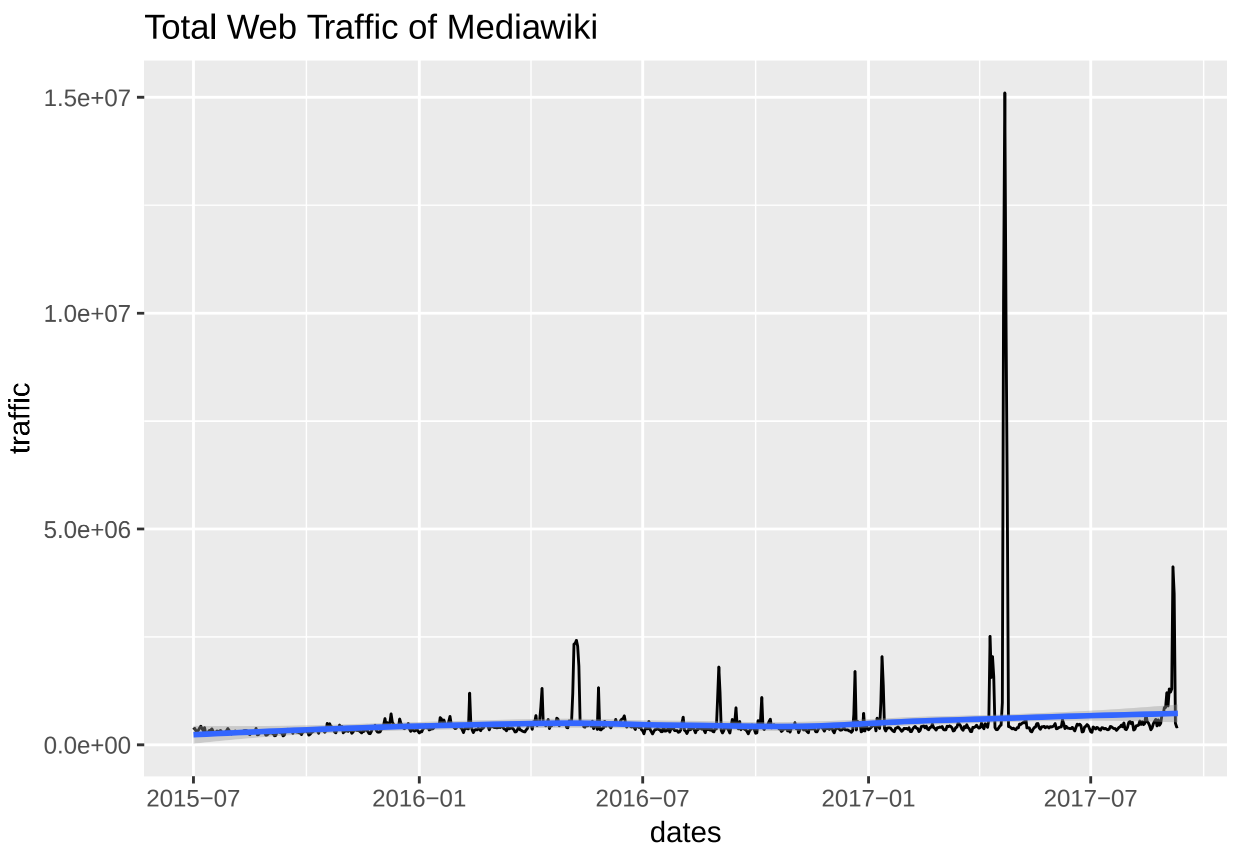
```
wikimediaTotal = wikimediaTotal %>% t() %>% as.data.frame %>% rownames_to_column %>%
  rename(dates = rowname, traffic = V1) %>% mutate(dates = as.Date(dates))
mediawikiTotal = mediawikiTotal %>% t() %>% as.data.frame %>% rownames_to_column %>%
  rename(dates = rowname, traffic = V1) %>% mutate(dates = as.Date(dates))
ggplot(wikipediaTotal, aes(dates, traffic)) + geom_line() + geom_smooth(method = 'loess') +
  labs(title = "Total Web Traffic of Wikipedia")
```



```
ggplot(wikimediaTotal, aes(dates, traffic)) + geom_line() + geom_smooth(method = 'loess') +
  labs(title = "Total Web Traffic of Wikimedia")
```



```r
ggplot(mediawikiTotal, aes(dates, traffic)) + geom_line() + geom_smooth(method = 'loess') +
  labs(title = "Total Web Traffic of Mediawiki")
```



Wikipedia has a much higher number of views than wikimedia and mediawiki. The web traffic of wikipedia increases a lot from 2015-07 to the end of 2016, and then decreases. Wikimedia shows a smoothly increasing trend, and the trend of mediawiki is a flat curve. There are different types of wikipedia project, one thing that might be interesting is that how these different project might affect web traffic.

```
table(Pages$project)
```

```
##
##        de        en        es        fr        ja mediawiki        ru
##     18547     24108     14069     17802     20431      7300     15022
## wikimedia        zh
##     10555     17229
```

We can see that there are seven languages plus wikimedia and mediawiki. The languages used here are: English, Japanese, German, French, Chinese, Russian, and Spanish.

```
rowsnum = Pages %>% filter(project == "en") %>% select(rowname)
english = data %>% filter(rowname %in% as.vector(t(rowsnum))) %>% select(-c(rowname, Page))
german = data %>% filter(rowname %in% Pages$rowname[which(Pages$project == "de")])  %>%
  select(-c(rowname, Page))
spanish = data %>% filter(rowname %in% Pages$rowname[which(Pages$project == "es")])  %>%
  select(-c(rowname, Page))
french = data %>% filter(rowname %in% Pages$rowname[which(Pages$project == "fr")])  %>%
  select(-c(rowname, Page))
japanese = data %>% filter(rowname %in% Pages$rowname[which(Pages$project == "ja")])  %>%
  select(-c(rowname, Page))
russian = data %>% filter(rowname %in% Pages$rowname[which(Pages$project == "ru")])  %>%
  select(-c(rowname, Page))
chinese = data %>% filter(rowname %in% Pages$rowname[which(Pages$project == "zh")])  %>%
```

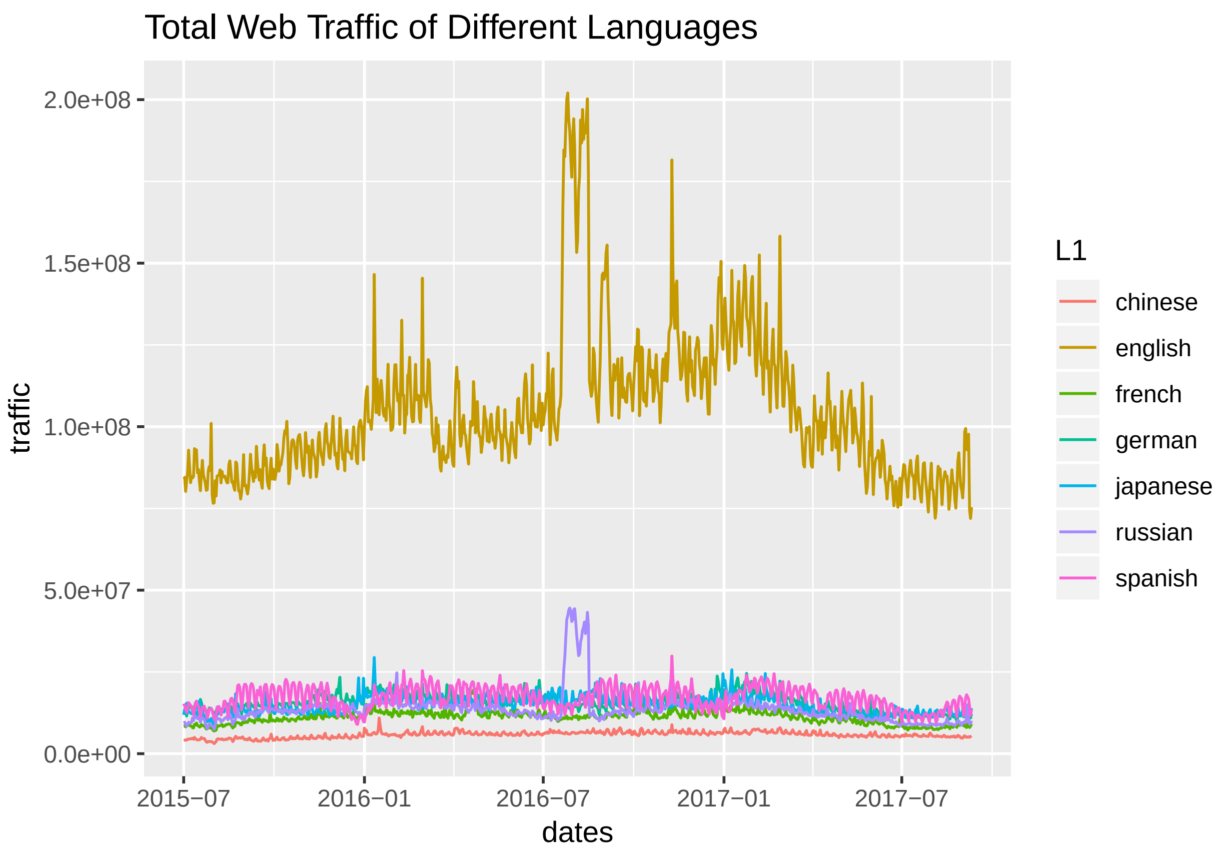
```r
  select(-c(rowname, Page))
languages = list(english=english, german=german, spanish=spanish, french=french,
  japanese=japanese, russian=russian, chinese=chinese)
langs = names(languages)
languagesSum = list()
for (l in langs){
    languagesSum[[l]] = as.data.frame(t(sapply(languages[[l]], margin = 2, sum, na.rm = TRUE)))
}
plotLang = melt(languagesSum)
```

```
## No id variables; using all as measure variables
## No id variables; using all as measure variables
## No id variables; using all as measure variables
## No id variables; using all as measure variables
## No id variables; using all as measure variables
## No id variables; using all as measure variables
## No id variables; using all as measure variables
```

```r
sample_n(plotLang, 6)
```

```
##     variable    value       L1
## 1 2015-07-13 92391927  english
## 2 2015-11-09 10987618   french
## 3 2015-10-14 12005638 japanese
## 4 2016-11-18  6188885  chinese
## 5 2016-05-19  5485875  chinese
## 6 2017-05-18 17869433  spanish
```

```r
ggplot(plotLang, aes(as.Date(variable), value)) + geom_line(aes(color=L1)) +
  labs(x="dates", y="traffic", title = "Total Web Traffic of Different Languages")
```



English shows a much higher number of views. The English and Russian plots show very large spikes around 2016-08, with several more spikes in the English data later in 2016 and earlier in 2017. There are also several spikes in the Engilish data earlier in 2016. There is a clear periodic structure in the Spanish data.

Next analyzing different types of access and different types of agent.

```
table(Pages$project)
```

```
##
##        de        en        es        fr        ja mediawiki        ru
##     18547     24108     14069     17802     20431      7300     15022
## wikimedia        zh
##     10555     17229
```

```
table(Pages$access)
```

```
##
## all-access    desktop mobile-web
##      74315      34809      35939
```

```
table(Pages$agent)
```

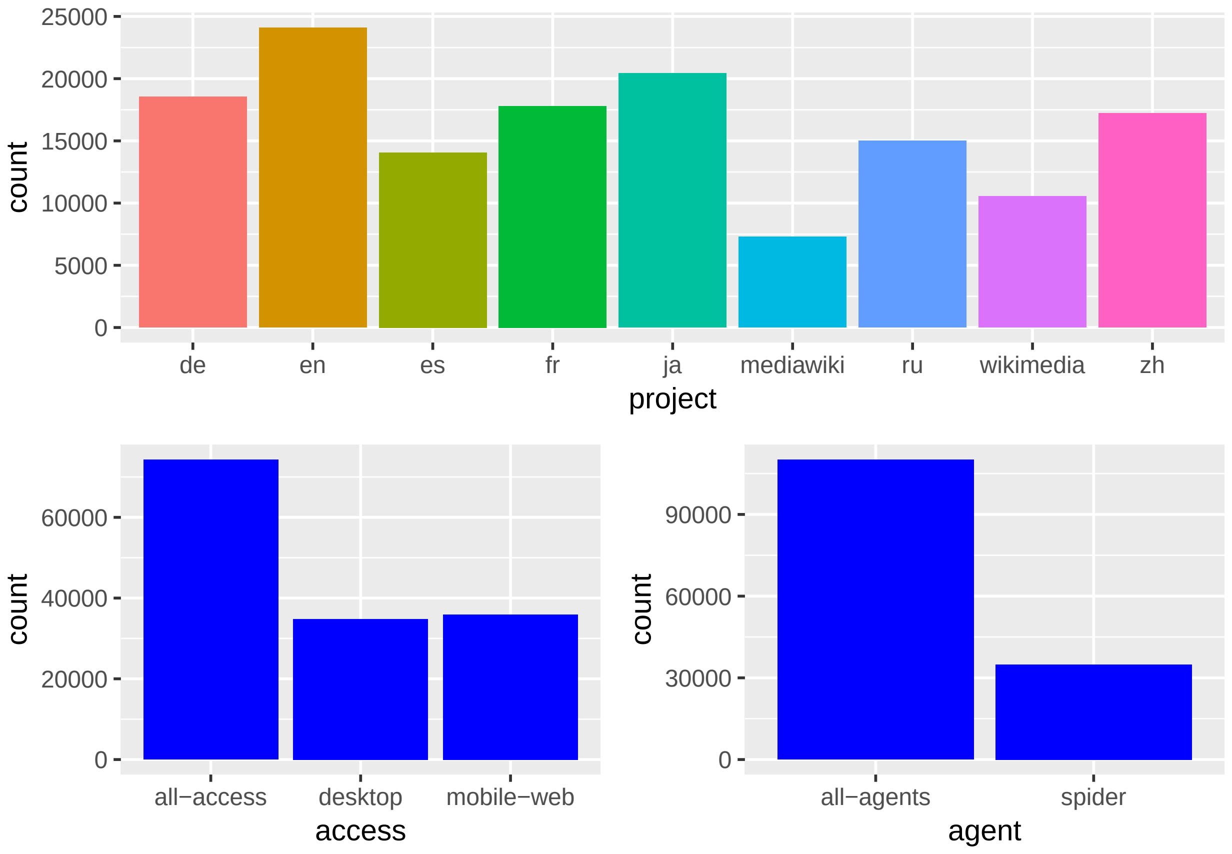
```
##
## all-agents     spider
##     110150      34913
```

In addition to seven languages, there are three types of access including all-access, desktop and mobile-web, and two types of agent including all-agents and spider.

```
library(grid)
#library(Rmisc)
```

```r
p1 = ggplot(Pages, aes(project, fill=project)) + geom_bar(show.legend = FALSE)
p2 = ggplot(Pages, aes(access)) + geom_bar(fill = 'blue')
p3 = ggplot(Pages, aes(agent)) + geom_bar(fill = 'blue')

grid.newpage()
pushViewport(viewport(layout = grid.layout(2,2)))
vplayout = function(x,y)viewport(layout.pos.row = x,layout.pos.col = y)
print(p1,vp = vplayout(1,1:2))
print(p2,vp = vplayout(2,1))
print(p3,vp = vplayout(2,2))
```



```r
dev.off()
```

```
## null device
##           1
```

```r
# multiplot(plotlist = list(p1,p2,p3), layout = matrix(c(1,1,2,3), nrow = 2, byrow = TRUE))
```

```r
nrow(data)
```

```
## [1] 145063
```

```r
Data = merge(data, Pages, by = 'rowname', sort = FALSE)
nrow(Data)
```

```
## [1] 145063
```

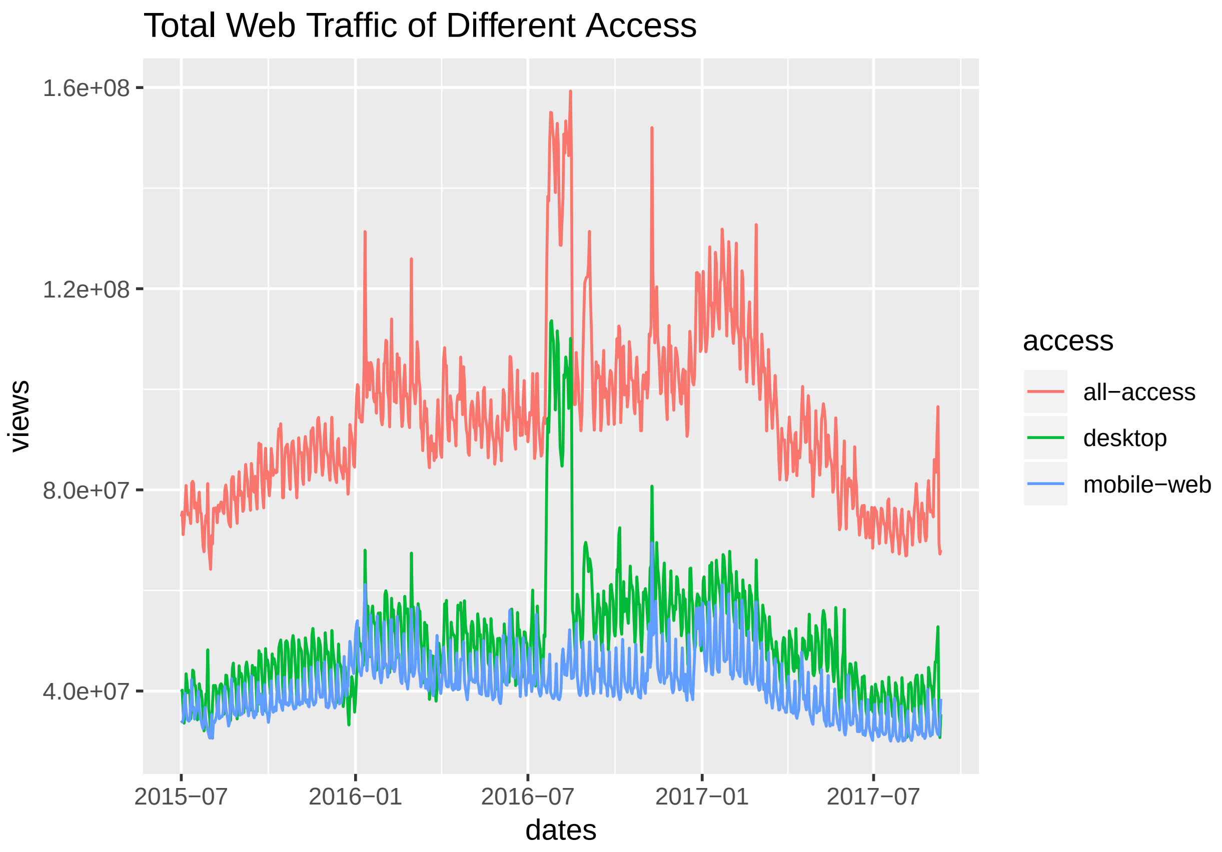
```r
Data = Data %>% select(-c(rowname, Page)) %>% gather(dates, traffic, -c(name, project, access, agent))
head(Data)
```

```
##                name project     access  agent      dates traffic
## 1              2NE1      zh all-access spider 2015-07-01      18
## 2               2PM      zh all-access spider 2015-07-01      11
## 3                3C      zh all-access spider 2015-07-01       1
## 4           4minute      zh all-access spider 2015-07-01      35
## 5 52_Hz_I_Love_You      zh all-access spider 2015-07-01      NA
## 6              5566      zh all-access spider 2015-07-01      12
```

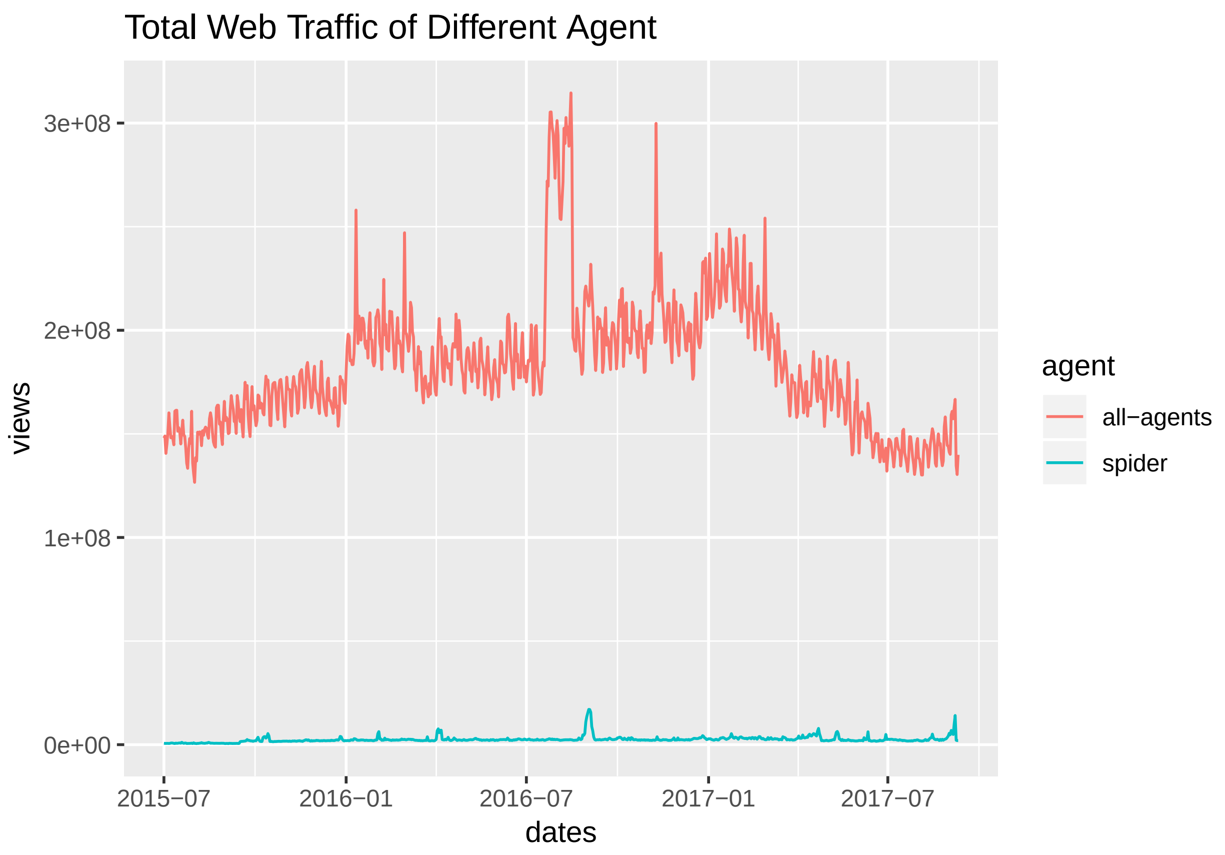
```
nrow(Data)
```

```
## [1] 116485589
```

```
temp = Data %>% select(dates, access, traffic) %>% group_by(dates, access) %>%
  summarize(views = sum(traffic, na.rm = TRUE))
temp %>% ggplot(aes(ymd(dates), views, color = access)) + geom_line() +
  labs(x = "dates", y = "views", title = "Total Web Traffic of Different Access")
```



```
temp = Data %>% select(dates, agent, traffic) %>% group_by(dates, agent) %>%
  summarize(views = sum(traffic, na.rm = TRUE))
temp %>% ggplot(aes(ymd(dates), views, color = agent)) + geom_line() +
  labs(x = "dates", y = "views", title = "Total Web Traffic of Different Agent")
```



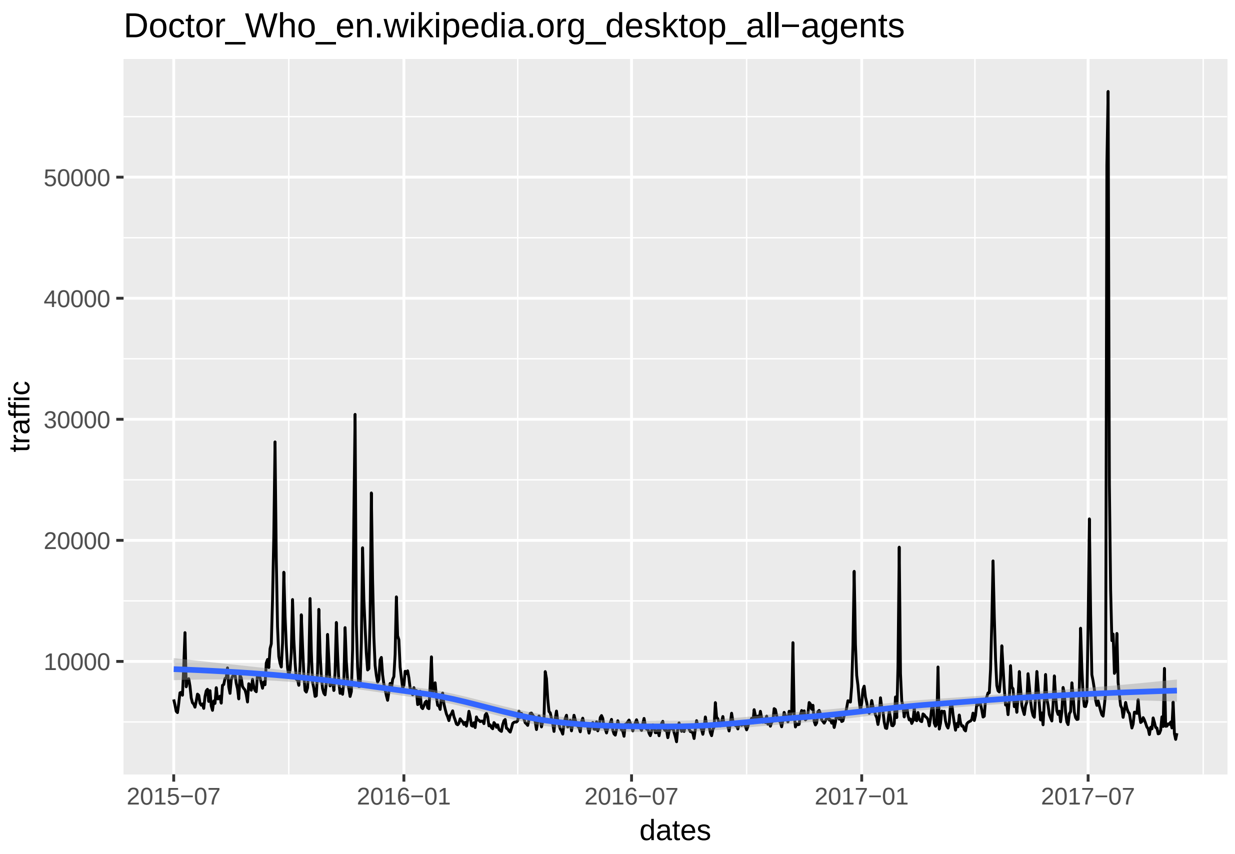
Picking top 3 articles for different types of project.

```
Data %>% group_by(project, name) %>% summarize(views = sum(as.numeric(traffic),
         na.rm = TRUE)) %>% top_n(3, views)
```

```
## # A tibble: 27 x 3
## # Groups:   project [9]
##    project name                          views
##    <chr>   <chr>                         <dbl>
##  1 de      Hauptseite                122958675
##  2 de      Spezial:Suche             634959450
##  3 de      Wikipedia:Hauptseite     4351586319
##  4 en      Main_Page               34143306286
##  5 en      Special:Book              460161692
##  6 en      Special:Search           3983242945
##  7 es      Especial:Buscar           632021971
##  8 es      Especial:Entrar           137325510
##  9 es      Wikipedia:Portada        1920205952
## 10 fr      "Sp\u00e9cial:Recherche"   287116031
## # ... with 17 more rows
```

In order to conveniently extract one particular time series, we need a helper function that allows us to plot one time series systematically based on a row number.

```
extractTimeSeries <- function(rowNumber){
    curPageName = data[rowNumber,]['Page']
    data[rowNumber,] %>% select(-c(rowname, Page)) %>% t %>% as.data.frame %>%
                    rownames_to_column %>% dplyr::rename(dates = rowname, traffic = V1) %>%
                    mutate(dates = as.Date(dates)) %>% ggplot(aes(dates, traffic)) +
                      geom_line() + geom_smooth(method = "loess") + labs(title = str_c(curPageName))
}
extractTimeSeries(11214)
```



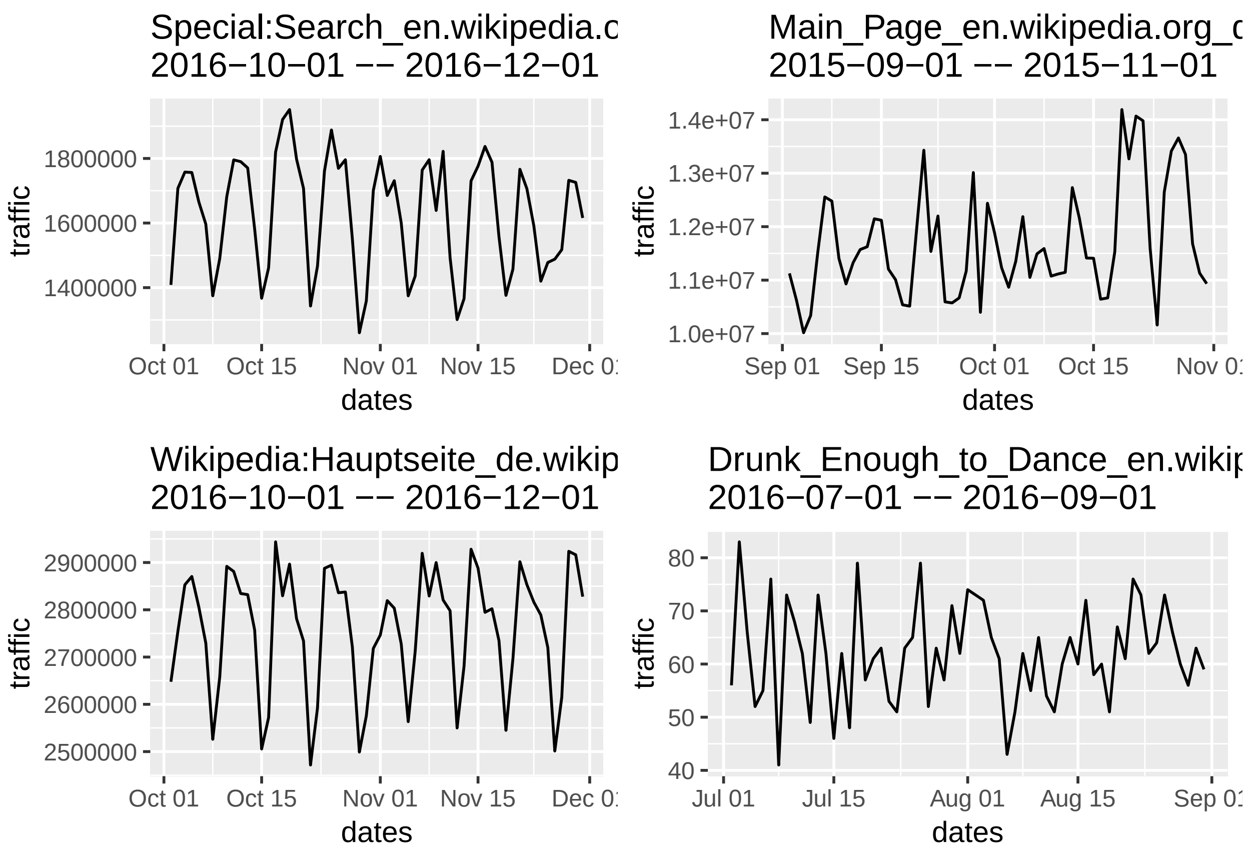
Taking a closer look to the short-term variability.

```
plotZoomIn <- function(rowNumber, beginDate, endDate){
    curPageName = data[rowNumber,]['Page']
    data[rowNumber,] %>% select(-c(rowname, Page)) %>% t %>% as.data.frame %>%
                     rownames_to_column %>% dplyr::rename(dates = rowname, traffic = V1) %>%
                     mutate(dates = as.Date(dates)) %>% filter(dates > beginDate & dates < endDate) %>%
                     ggplot(aes(dates, traffic)) + geom_line() +
                     labs(title = str_c(curPageName, "\n", beginDate, " -- ", endDate))
}

p1 <- plotZoomIn(10404, "2016-10-01", "2016-12-01")
p2 <- plotZoomIn(9775, "2015-09-01", "2015-11-01")
p3 <- plotZoomIn(139120, "2016-10-01", "2016-12-01")
p4 <- plotZoomIn(110658, "2016-07-01", "2016-09-01")
library(Rmisc)
```

```
## Loading required package: lattice
```

```
layout <- matrix(c(1,2,3,4),2,2,byrow=TRUE)
multiplot(p1, p2, p3, p4, layout=layout)
```



From above figure, we can see that the two plots on the left hand side show a similar periodicity. The two plots on the right hand side show a similar structure, but the upper right plot is influenced by a upward trend, and the lower right plot is influenced by a decreasing view counts. These plots provide evidence that there is variability on a weekly scale.

Data Preprocessing

Missing values

Here we convert NA to zero.

```
data[is.na(data)] = 0
```

Dealing with data

```
Max = apply(data[,-c(1,2)], 1, max)
Min = apply(data[,-c(1,2)], 1, min)
Avg = apply(data[,-c(1,2)], 1, mean)
Std = apply(data[,-c(1,2)], 1, sd)

# dataSet = data[, -c(1,2)] %>% as.matrix() %>% t() %>% scale() %>% t() %>% as.data.frame() %>% rowname
# head(dataSet)
```

Use log1p to transform data, and to improve model performance, use medians as one of the features.

```
dataSet = log1p(data[, -c(1,2)]) %>% rownames_to_column()
# head(dataSet)
```

Splitting dataset into training set and testing set

Now dividing original dataset into traning set and test set. The time series in training set starts from July, 1st, 2015 up until June 30th, 2017. And the test dataset consists of times series ranging from July 1st, 2017 to September 10th, 2017.

```
i = match("2017-06-30", names(dataSet))
trainSet = select(dataSet, 1:i)
testSet = select(dataSet, c(1,(i+1):804))
trainSet %>% names %>% head(10)
```

```
##  [1] "rowname"    "2015-07-01" "2015-07-02" "2015-07-03" "2015-07-04"
##  [6] "2015-07-05" "2015-07-06" "2015-07-07" "2015-07-08" "2015-07-09"
```

```
testSet %>% names %>% head(10)
```

```
##  [1] "rowname"    "2017-07-01" "2017-07-02" "2017-07-03" "2017-07-04"
##  [6] "2017-07-05" "2017-07-06" "2017-07-07" "2017-07-08" "2017-07-09"
```

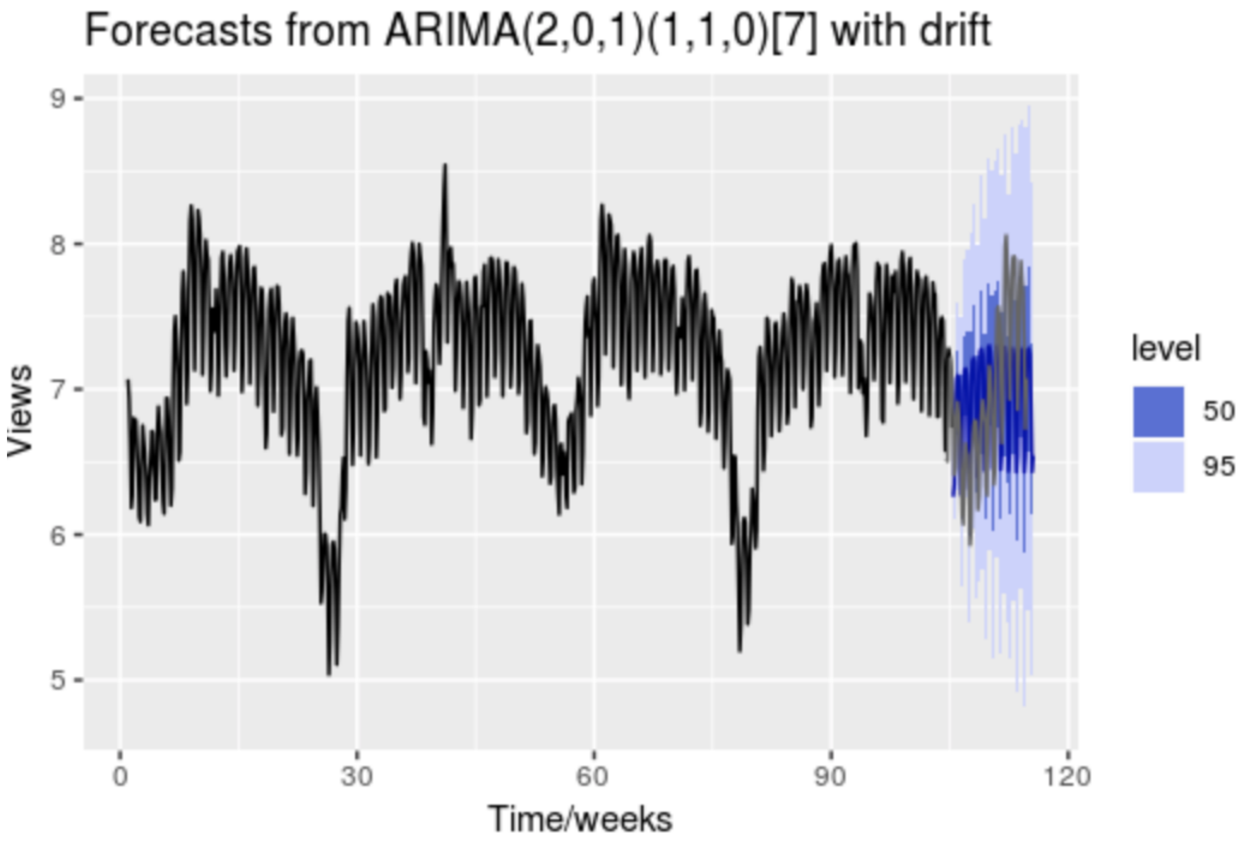
Model Fitting

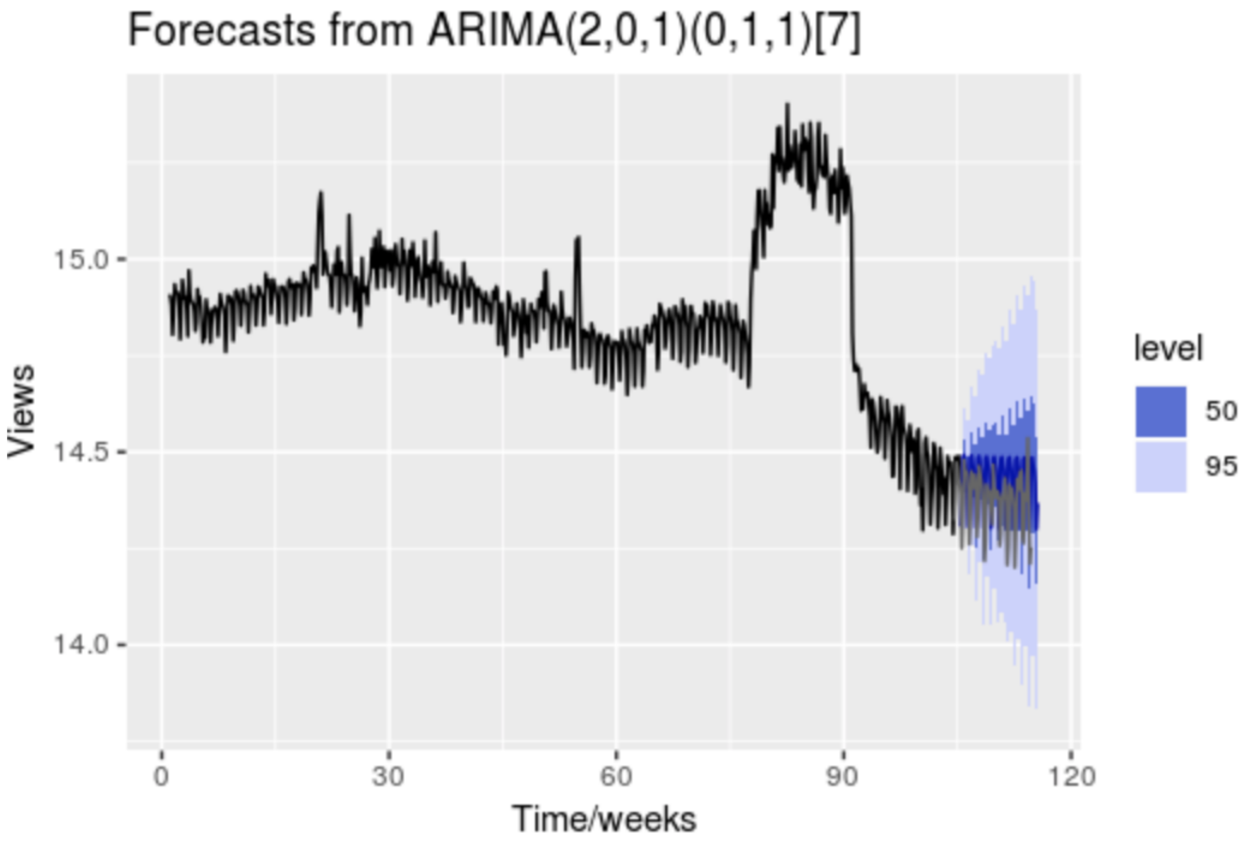
ARIMA

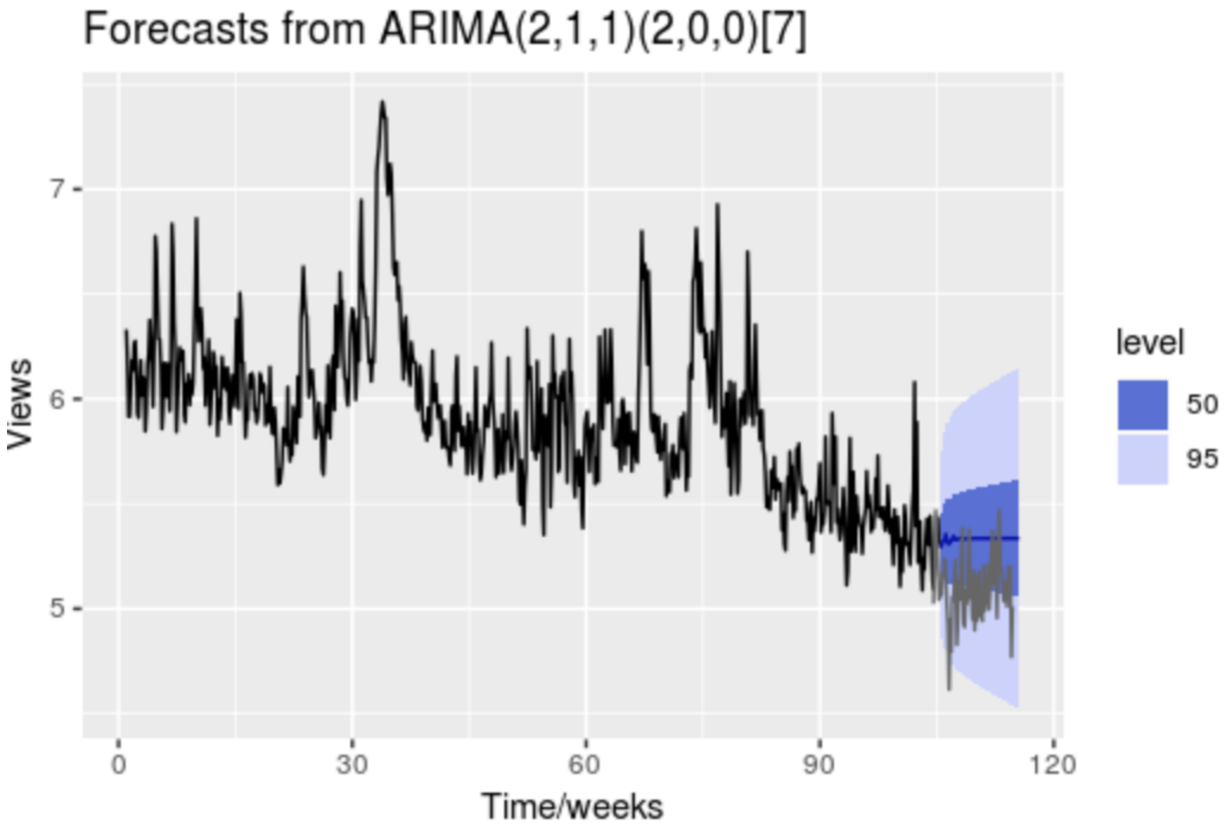
```
library(doSNOW)
library(doRNG)
library(foreach)
library(parallel)
library(tseries)
library(forecast)
cl = makeCluster(20)
registerDoSNOW(cl)
fc.wiki <- foreach(i=1:nrow(trainSet), .combine=rbind, .packages="forecast") %dopar% {
  y <- tsclean(as.ts(unlist(trainSet[i, -1]), frequency = 7))
  forecast(auto.arima(y, max.p=2, max.d=2, max.q=1), h=72)$mean
}
stopCluster(cl)
```

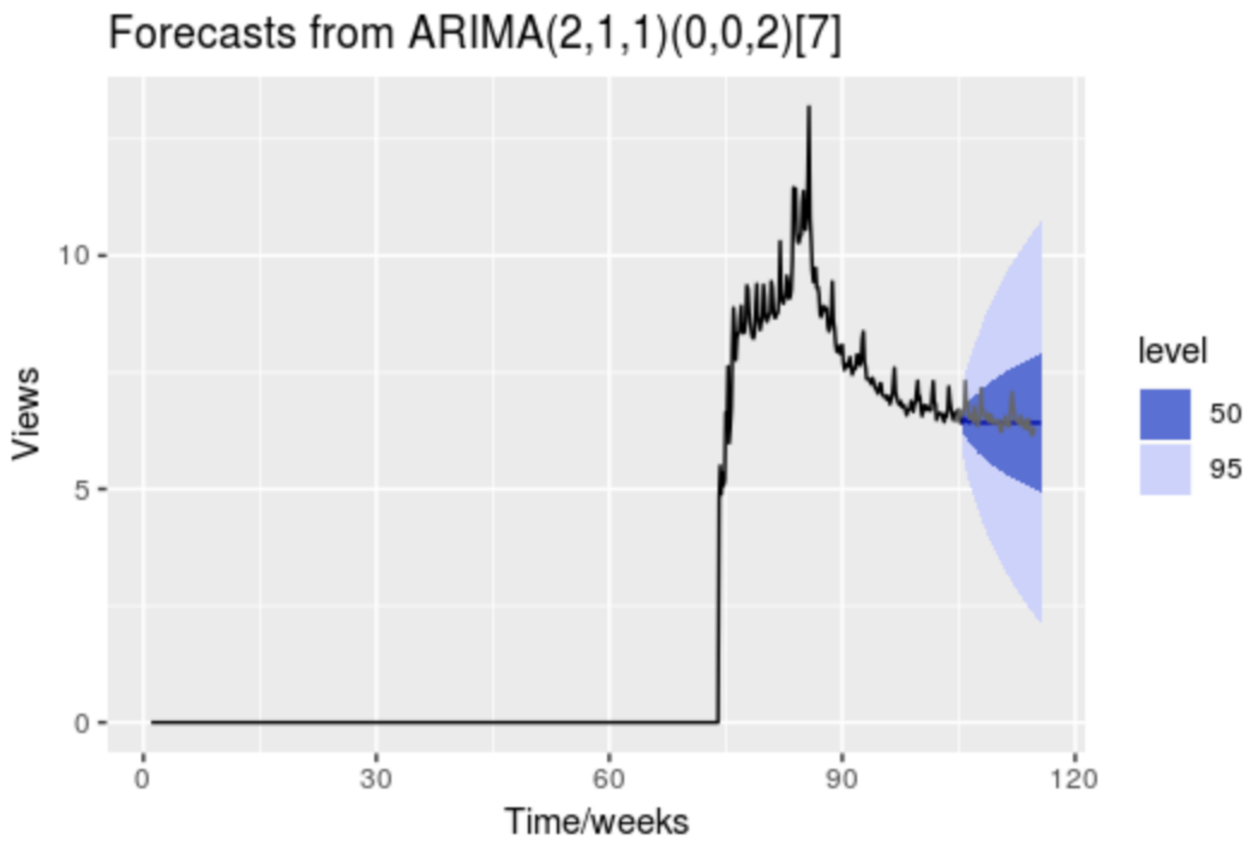
Let's take a look about how ARIMA performs.

```
plotARIMA <- function(index){
  y <- tsclean(ts(unlist(trainSet[index, -1]), frequency = 7))
  fc.index = forecast(auto.arima(y, max.p=2, max.d=2, max.q=1), h=72, level = c(50,95))
  true.index = testSet[index, -1] %>% t %>% as.data.frame() %>% rownames_to_column()
  colnames(true.index) = c("dates", "traffic")
  autoplot(fc.index) + geom_line(aes(c(732:803)/7, traffic), data = true.index, color = "grey40") + labs
}
plotARIMA(95786)
plotARIMA(139120)
plotARIMA(6597)
plotARIMA(37862)
```




Forecasts from ARIMA(2,0,1)(0,1,1)[7]
Views
15.0
14.5
14.0
0
30
60
90
120
Time/weeks
level
50
95


Forecasts from ARIMA(2,1,1)(2,0,0)[7]
Views
7
6
5
0
30
60
90
120
Time/weeks
level
50
95



LSTM

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

data = pd.read_csv('web-traffic-time-series-forecasting/data.csv').fillna(0)
data.head()
data = data.drop("Page",1)
names = data.columns.values
i = np.where(names == "2017-06-30")
trainSet = data.iloc[:,0:731]
trainSet.head()
testSet = data.iloc[:,731:]
testSet.head()
look_back = 7
train_index = trainSet.iloc[37862,:].values
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler(feature_range = (0, 1))
training_set_scaled = sc.fit_transform(train_index)
X_train = []
y_train = []
for i in range(look_back, len(train_index)):
    X_train.append(training_set_scaled[i-look_back:i])
    y_train.append(training_set_scaled[i])
```

```
X_train, y_train = np.array(X_train), np.array(y_train)
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
X_train.shape
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
regressor = Sequential()
regressor.add(LSTM(units = 12, activation = 'relu', input_shape = (None, 1)))
regressor.add(Dense(units = 1))
regressor.compile(optimizer = 'adam', loss = 'mean_squared_error')
regressor.fit(X_train, y_train, batch_size = 10, epochs = 100, verbose = 0)
test_index = testSet.iloc[37862,:].values
test_set_scaled = sc.fit_transform(test_index)
X_test = []
y_test = []
for i in range(look_back, len(test_set_scaled)):
    X_test.append(test_set_scaled[i-look_back:i])
    y_test.append(test_set_scaled[i])
X_test, y_test = np.array(X_test), np.array(y_test)
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
y_pred = regressor.predict(X_test)

plt.figure
plt.plot(y_test, color = 'black', label = 'Real Web View')
plt.plot(y_pred, color = 'blue', label = 'Predicted Web View')
plt.title('Web View Forecasting')
plt.xlabel('Number of Days')
plt.ylabel('Web View')
plt.legend()
plt.show()
```

Conclusion

Based on the results we have and some indicators like MAE, LSTM is better than ARIMA in real-time forecasting. However, ARIMA performance can be used as a benchmark to compare other models' performances. To further improve forecasting performance, maybe it is better to first cluster these time-series, and then to train different models for each cluster.

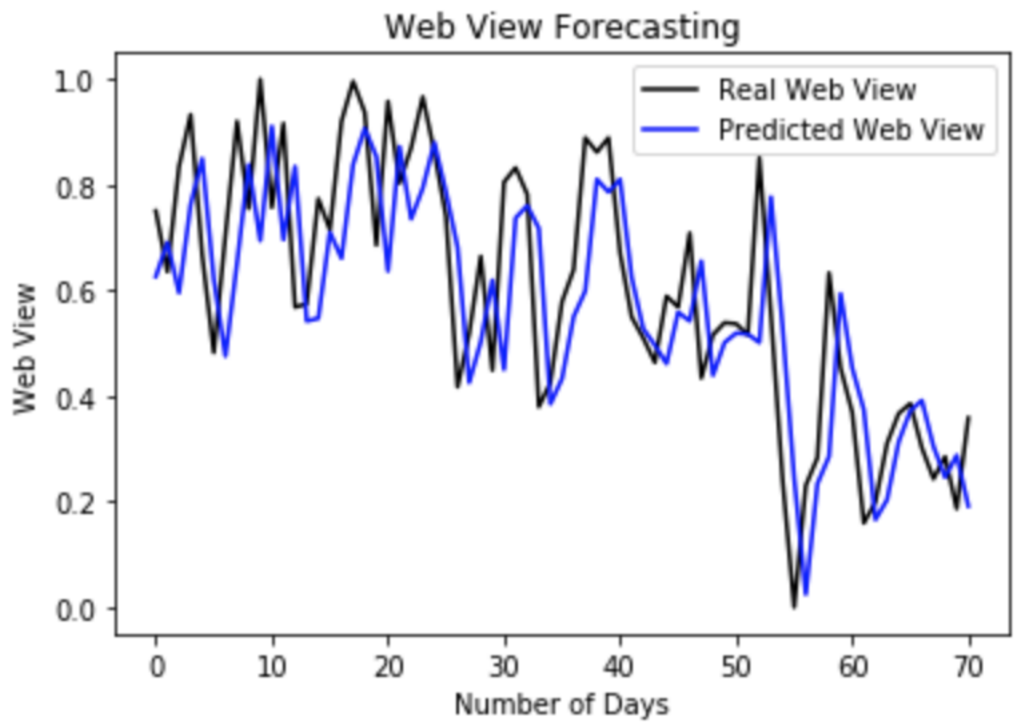


Figure 1:

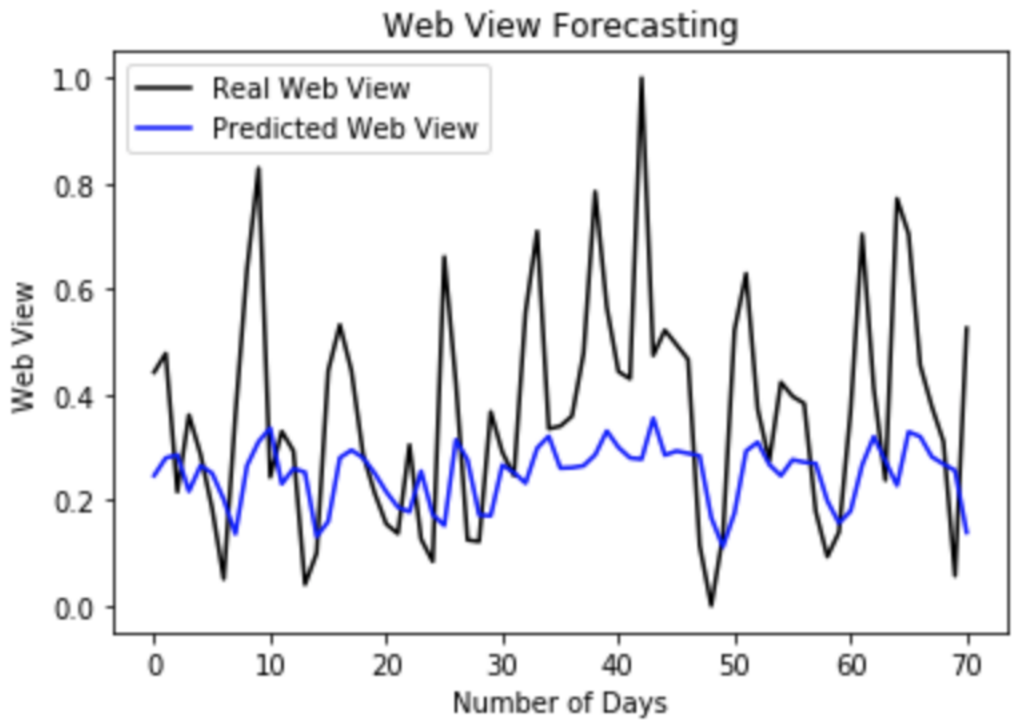


Figure 2:

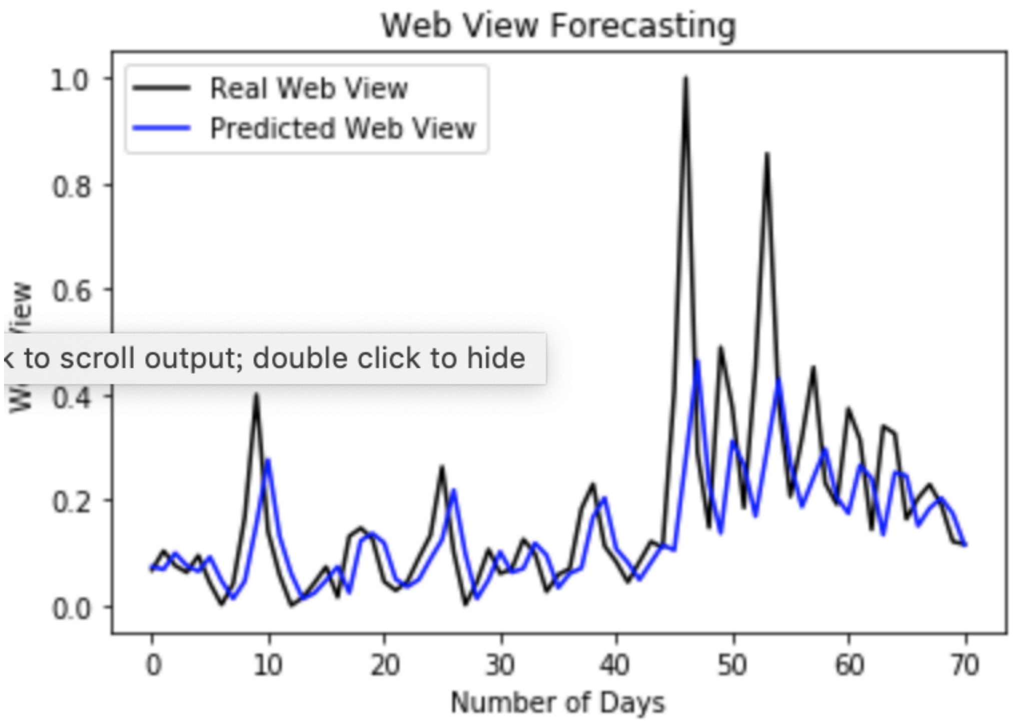


Figure 3:

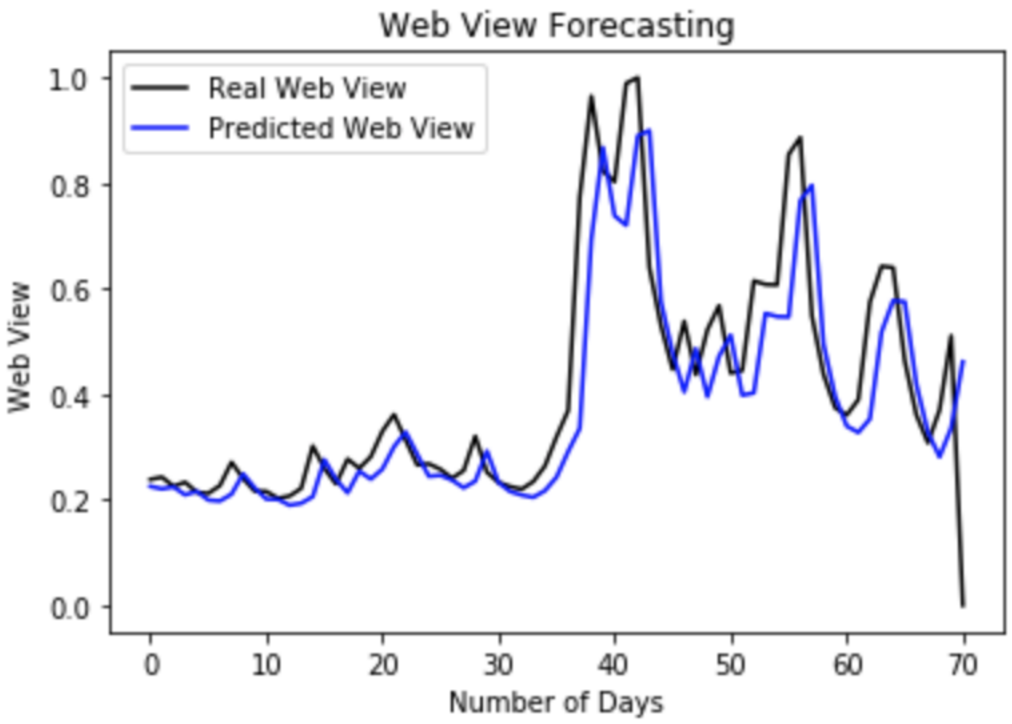


Figure 4: