Road Traffic Dataset

Total lines of code 102.

FINISHED

Data loading and running of some SQL queries took between 1 and 2 minutes as the data set consists of more than 20 million rows.

Took 3 sec. Last updated by anonymous at March 31 2017, 1:26:15 PM.

```
%pyspark
from pandas import Series, DataFrame
import pandas as pd
import numpy as np
import glob,os
path =  "/Users/Kapil/Downloads/traffic_feb_june"
roadtraffic = pd.concat(map(pd.read_csv, glob.glob(os.path.join(path, "*.csv"))))
```

FINISHED

Took 52 sec. Last updated by anonymous at April 16 2017, 9:44:29 PM.

```
%pyspark
inputPath =  "/Users/Kapil/Downloads/traffic_feb_june"
roadtraffic2 = sqlContext.read.format("com.databricks.spark.csv").option("header", "true"
```

FINISHED

Took 2 min 0 sec. Last updated by anonymous at April 16 2017, 9:46:43 PM.

```
%pyspark
roadtraffic2.registerTempTable("road_traffic")
```

FINISHED

Took 0 sec. Last updated by anonymous at April 16 2017, 10:29:53 PM.

```
%pyspark
print(roadtraffic.count())
```

FINISHED

```
status               20713165
avgMeasuredTime      20713165
avgSpeed             20713165
extID                20713165
medianMeasuredTime   20713165
TIMESTAMP            20713165
vehicleCount         20713165
_id                  20713165
REPORT_ID            20713165
dtype: int64
```

Took 11 sec. Last updated by anonymous at March 31 2017, 12:21:50 PM.

```
%pyspark
roadtraffic[-5:]
```

FINISHED

```
      status  avgMeasuredTime  avgSpeed  extID  medianMeasuredTime  \
16938     OK              112        36    623                 112
16939     OK              112        36    623                 112
16940     OK              112        36    623                 112
16941     OK              112        36    623                 112
16942     OK              112        36    623                 112
                 TIMESTAMP  vehicleCount       _id  REPORT_ID
16938  2014-09-30T23:35:00             0  28062086     210199
16939  2014-09-30T23:40:00             0  28062468     210199
16940  2014-09-30T23:45:00             0  28062917     210199
16941  2014-09-30T23:50:00             0  28063308     210199
16942  2014-09-30T23:55:00             0  28063757     210199
```

Took 0 sec. Last updated by anonymous at March 31 2017, 12:21:55 PM. (outdated)

FINISHED

```
%pyspark
import re
roadtraffic['hour'] = roadtraffic['TIMESTAMP'].str[11:13]
roadtraffic['minutes'] = roadtraffic['TIMESTAMP'].str[14:16]
roadtraffic['date'] = roadtraffic['TIMESTAMP'].str[0:10]
roadtraffic['months'] = roadtraffic['TIMESTAMP'].str[5:7]
```

Took 39 sec. Last updated by anonymous at March 31 2017, 12:36:24 PM.

FINISHED

```
%pyspark
roadtraffic[-5:]
```

```
      status  avgMeasuredTime  avgSpeed  extID  medianMeasuredTime  \
16938     OK              112        36    623                 112
16939     OK              112        36    623                 112
16940     OK              112        36    623                 112
16941     OK              112        36    623                 112
16942     OK              112        36    623                 112
                 TIMESTAMP  vehicleCount       _id  REPORT_ID hour minutes  \
16938  2014-09-30T23:35:00             0  28062086     210199   23      35
16939  2014-09-30T23:40:00             0  28062468     210199   23      40
16940  2014-09-30T23:45:00             0  28062917     210199   23      45
16941  2014-09-30T23:50:00             0  28063308     210199   23      50
16942  2014-09-30T23:55:00             0  28063757     210199   23      55
             date months
16938  2014-09-30     09
16939  2014-09-30     09
16940  2014-09-30     09
16941  2014-09-30     09
16942  2014-09-30     09
```

Took 0 sec. Last updated by anonymous at March 31 2017, 12:36:32 PM.

FINISHED

```
%pyspark
roadtraffic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 20713165 entries, 0 to 16942
Data columns (total 13 columns):
status                object
avgMeasuredTime       int64
avgSpeed              int64
```

```
extID                 int64
medianMeasuredTime    int64
TIMESTAMP             object
vehicleCount          int64
_id                   int64
REPORT_ID             int64
hour                  object
minutes               object
date                  object
months                object
dtypes: int64(7), object(6)
memory usage: 2.2+ GB
```

Took 8 sec. Last updated by anonymous at March 31 2017, 12:36:45 PM.

FINISHED

```
%pyspark
def get_stats(group): return {'min': group.min(), 'max': group.max(), 'count': group.cour
    group.mean()}
grouped_avgspeed_byhour =  roadtraffic['avgSpeed'].groupby(roadtraffic['hour'])
grouped_avgspeed_byhour.apply(get_stats).unstack()
```

```
         count    max       mean  min
hour
00    781546.0  149.0  48.058822  0.0
01    887090.0  149.0  48.348357  0.0
02    904458.0  149.0  48.544265  0.0
03    900229.0  150.0  48.353697  0.0
04    865603.0  150.0  46.639626  0.0
05    846795.0  150.0  42.866610  0.0
06    867806.0  149.0  40.894512  0.0
07    894057.0  150.0  42.059450  0.0
08    895654.0  150.0  42.058984  0.0
09    885253.0  150.0  41.679704  0.0
10    878261.0  149.0  41.382687  0.0
11    894081.0  149.0  41.330556  0.0
12    896679.0  149.0  40.876565  0.0
13    898122.0  149.0  39.753508  0.0
```

Took 6 sec. Last updated by anonymous at March 31 2017, 12:40:34 PM. (outdated)

FINISHED

```
%pyspark
grouped_vehicleCount_byhour = roadtraffic['vehicleCount'].groupby(roadtraffic['hour'])
grouped_vehicleCount_byhour.apply(get_stats).unstack()
```

```
06    867806.0  121.0  5.720163  0.0
07    894057.0   99.0  5.069171  0.0
08    895654.0   79.0  5.036747  0.0
09    885253.0   65.0  5.164977  0.0
10    878261.0   67.0  5.303841  0.0
11    894081.0   77.0  5.357505  0.0
12    896679.0   90.0  5.627419  0.0
13    898122.0   97.0  6.022087  0.0
14    895562.0   94.0  6.050365  0.0
15    910856.0   85.0  4.940116  0.0
16    908012.0   74.0  3.505207  0.0
17    912939.0   71.0  2.405558  0.0
18    920914.0   68.0  1.797495  0.0
19    914818.0   78.0  1.529196  0.0
20    759371.0   86.0  1.253169  0.0
```

```
20     739371.0   86.0  1.232169  0.0
21     702923.0   85.0  0.782027  0.0
22     765726.0   58.0  0.398256  0.0
23     726410.0   67.0  0.249556  0.0
```

Took 4 sec. Last updated by anonymous at March 31 2017, 12:40:30 PM.

```
%pyspark
grouped_avgMeasuredTime = roadtraffic['avgMeasuredTime'].groupby(roadtraffic['hour'])
grouped_avgMeasuredTime.apply(get_stats).unstack()
```

FINISHED

```
          count     max        mean  min
hour
00     781546.0  3595.0   95.816972  0.0
01     887090.0  3587.0   95.751300  0.0
02     904458.0  3587.0   95.795096  0.0
03     900229.0  3587.0   96.949256  0.0
04     865603.0  3587.0  102.146635  0.0
05     846795.0  3587.0  114.061216  0.0
```

Took 4 sec. Last updated by anonymous at March 31 2017, 12:39:16 PM. (outdated)

```
%pyspark
grouped_avgspeed_bymonth =  roadtraffic['avgSpeed'].groupby(roadtraffic['months'])
grouped_avgspeed_bymonth.apply(get_stats).unstack()
```

FINISHED

```
            count    max       mean  min
months
02      1910192.0  149.0  42.935319  0.0
03      3485620.0  150.0  43.568671  0.0
04      3705591.0  150.0  43.705196  0.0
05      3681921.0  150.0  44.117599  0.0
06       793808.0  149.0  43.076908  0.0
08      3608396.0  150.0  44.692306  0.0
09      3527637.0  150.0  44.316682  0.0
```

Took 3 sec. Last updated by anonymous at March 31 2017, 12:43:16 PM. (outdated)

```
%pyspark
grouped_vehicle_bymonth =  roadtraffic['vehicleCount'].groupby(roadtraffic['months'])
grouped_vehicle_bymonth.apply(get_stats).unstack()
```

FINISHED

```
            count    max      mean  min
months
02      1910192.0  111.0  3.380066  0.0
03      3485620.0   97.0  3.443799  0.0
04      3705591.0  111.0  2.854224  0.0
05      3681921.0  100.0  3.252956  0.0
06       793808.0  121.0  2.805493  0.0
08      3608396.0  107.0  3.151881  0.0
09      3527637.0  108.0  3.200058  0.0
```

Took 4 sec. Last updated by anonymous at March 31 2017, 12:45:06 PM. (outdated)

```
%pyspark
grouped_avgTime_bymonth =  roadtraffic['avgMeasuredTime'].groupby(roadtraffic['months'])
grouped_avgTime_bymonth.apply(get_stats).unstack()
```

FINISHED

```
            count     max        mean  min
months
02      1910192.0  3587.0  103.188559  0.0
03      3485620.0  3648.0  104.820731  0.0
04      3705591.0  3595.0  105.554581  0.0
05      3681921.0  3656.0  109.032780  0.0
06       793808.0  3456.0  104.187155  0.0
08      3608396.0  3585.0  107.636983  0.0
09      3527637.0  3572.0  109.661455  0.0
```

Took 4 sec. Last updated by anonymous at March 31 2017, 12:45:28 PM. (outdated)

```
%sql
select * from road_traffic limit 15 --see the first 15 rows in the table
```

FINISHED

status	avgMeasuredTime	avgSpeed	extID	medianM
OK	141	93	672	141
OK	146	90	672	146
OK	155	84	672	155
OK	149	88	672	149
OK	146	90	672	146
OK	148	88	672	148
OK	150	87	672	150
OK	155	84	672	155

Took 1 sec. Last updated by anonymous at March 31 2017, 12:55:06 PM.

```
%sql
select count(_id) from road_traffic --count total number of rows
    in the table
```

FINISHED

count(_id)
20713165

Took 18 sec. Last updated by anonymous at March 31 2017, 12:50:56 PM. (outdated)

```
%sql
select distinct status from road_traffic--to see what distinct
    values are in status column
```

FINISHED

status
OK

Took 19 sec. Last updated by anonymous at March 31 2017, 12:53:08 PM. (outdated)

```
%sql
select distinct extID
from road_traffic
order by extID --to see what distinct values are in extID column
```

READY

eID
610
611
612
613
614

```
%sql
select count(eID) from (select distinct extID as eID from road_traffic group by extID) --
    number of distinct values in extID column
```

FINISHED

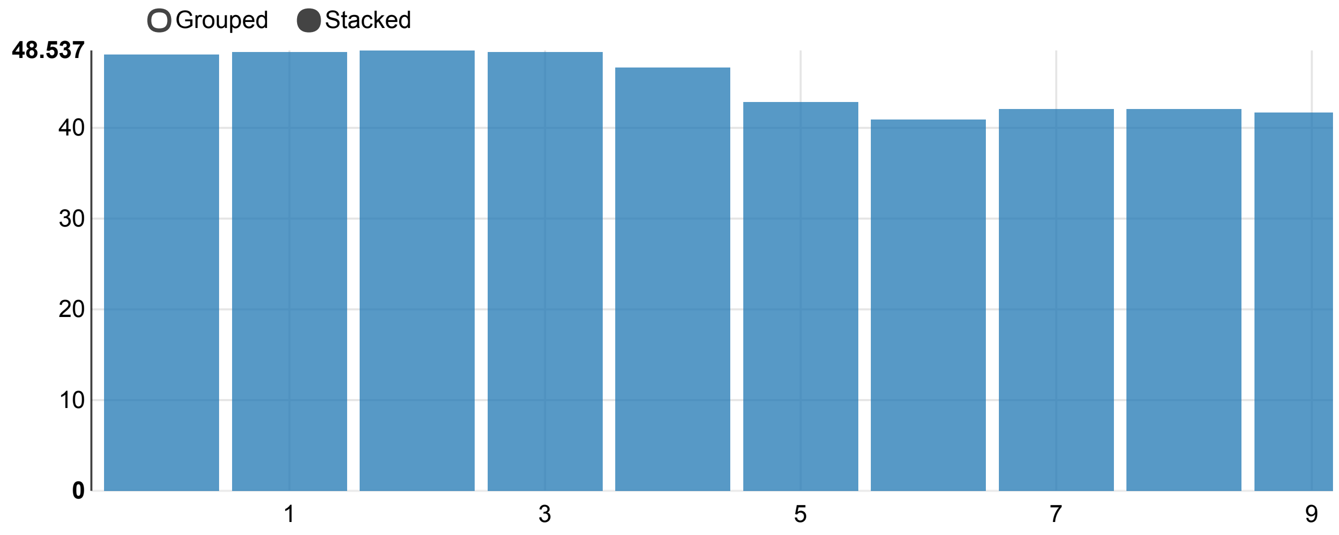
count(eID)
449

Took 18 sec. Last updated by anonymous at March 31 2017, 12:54:23 PM.

```
%sql
select avg(avgSpeed), hour(timestamp) as hour from road_traffic
group by hour(timestamp)
order by hour(timestamp)
```

FINISHED

settings



Took 1 min 35 sec. Last updated by anonymous at March 31 2017, 1:03:43 PM. (outdated)

```
%sql
select avg(vehicleCount), hour(timestamp) as hour from road_traffic
group by hour(timestamp)
order by hour(timestamp)
```

FINISHED

settings

All fields:

avg(vehicleCount) | hour

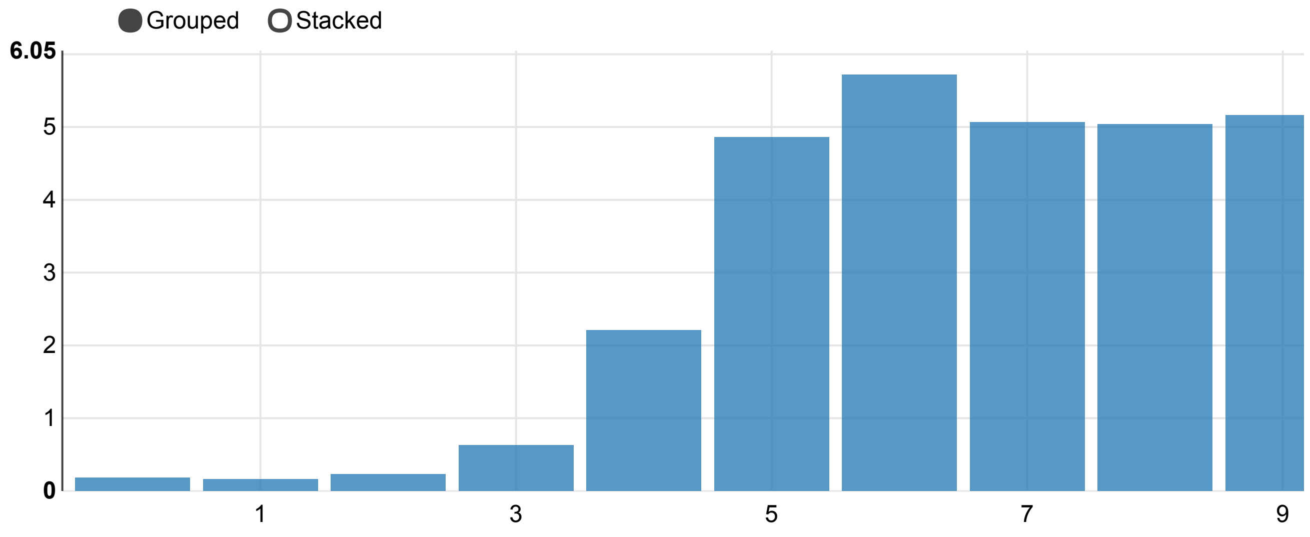
Keys

hour ✖

Groups

Values

avg(vehicleCount) SUM ✖



Took 1 min 41 sec. Last updated by anonymous at April 16 2017, 10:36:47 PM. (outdated)

```
%sql
select avgSpeed, vehicleCount
from road_traffic
```

FINISHED

settings ▲

All fields:

avgSpeed vehicleCount

xAxis

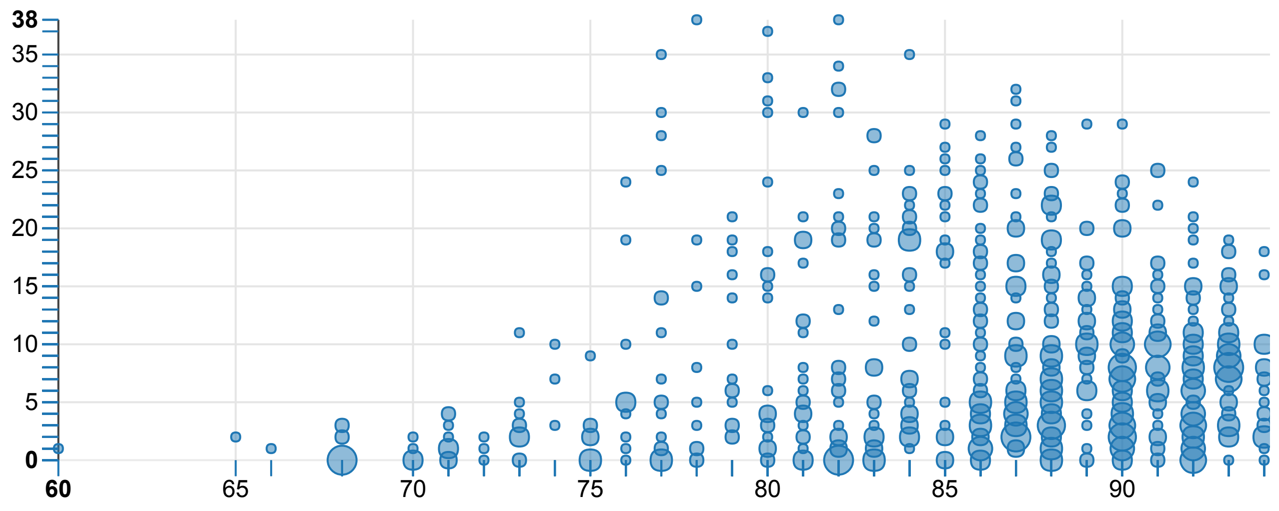
avgSpeed ✖

yAxis

vehicleCount ✖

group

size

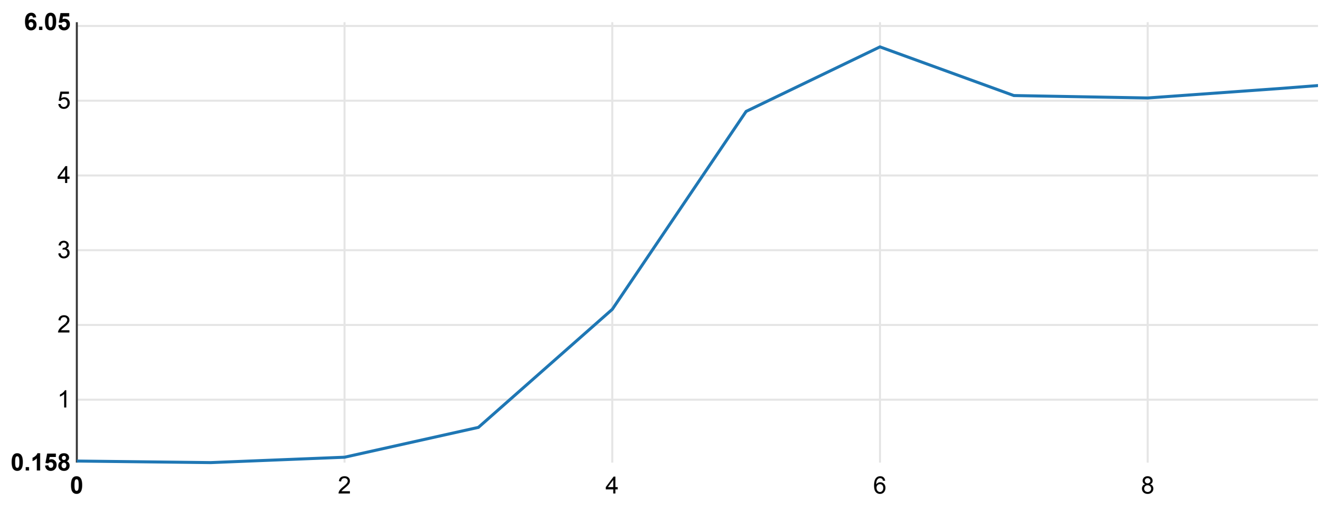


```
Results are limited by 1000.
```

Took 0 sec. Last updated by anonymous at March 31 2017, 1:01:10 PM. (outdated)

```
%sql
select avg(vehiclecount), hour(timestamp) as hour
from road_traffic
group by hour(timestamp)
order by hour(timestamp)
```

FINISHED



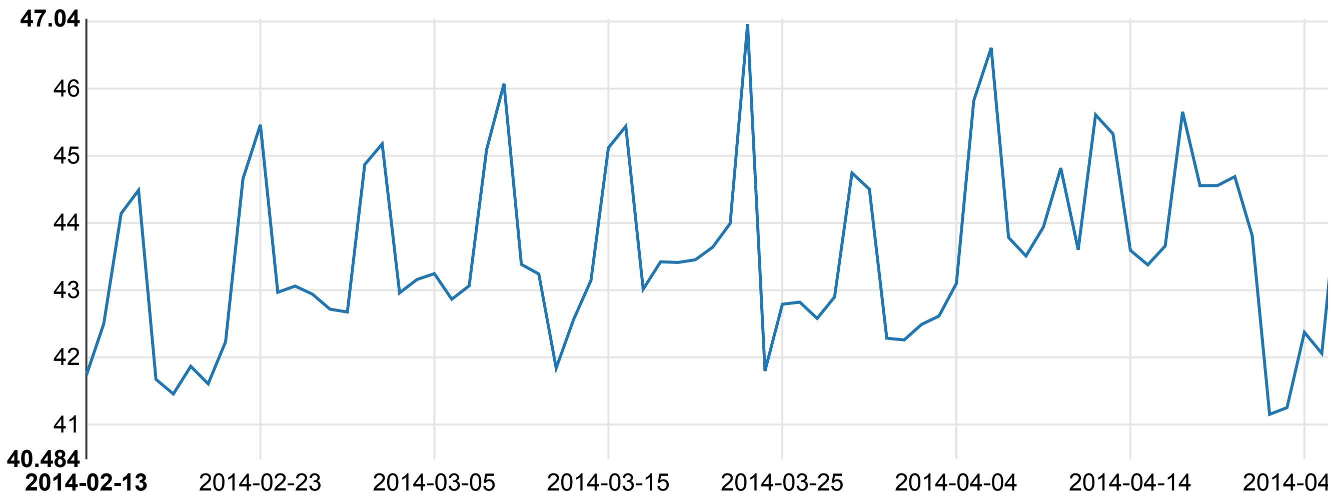
Took 2 min 59 sec. Last updated by anonymous at March 31 2017, 1:05:12 PM. (outdated)

```
%sql
select date(timestamp) date, avg(avgSpeed)
from road_traffic
group by date(timestamp)
```

FINISHED

```
5 order by date(timestamp)
```

settings



Took 1 min 30 sec. Last updated by anonymous at March 31 2017, 1:11:35 PM. (outdated)

```
%sql
select *
from road_traffic
order by road_traffic.timestamp
```

FINISHED

settings

All fields:

status avgMeasuredTime avgSpeed extID medianMeasuredTime TIMESTAMP vehicleCount _id REPORT_ID

Keys

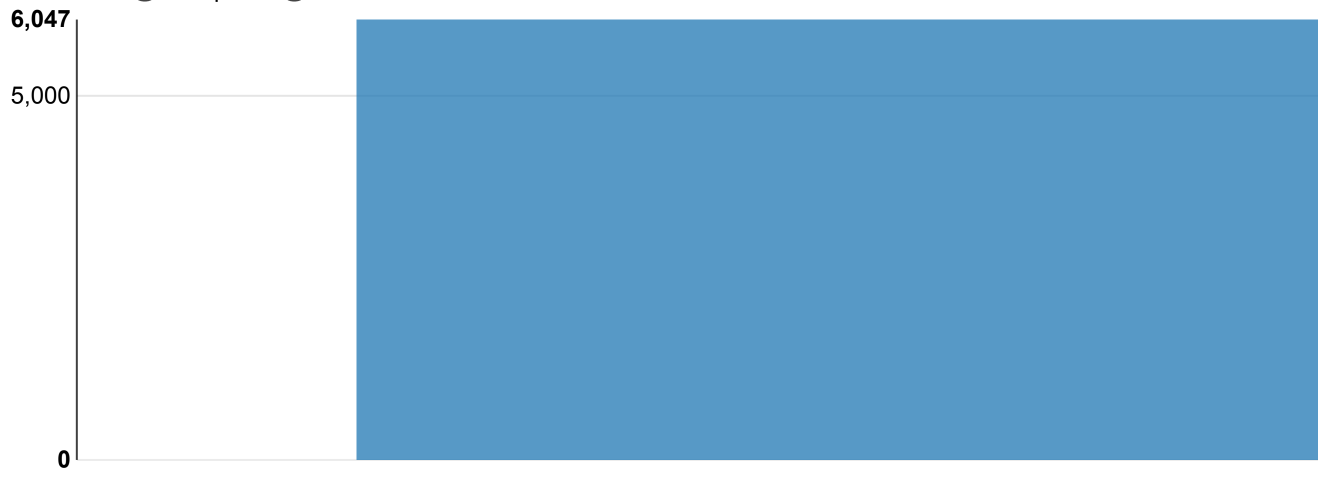
status ✖

Groups

Values

vehicleCount SUM ✖

● Grouped ○ Stacked



```
Results are limited by 1000.
```

Took 1 min 43 sec. Last updated by anonymous at March 31 2017, 1:15:18 PM. (outdated)

ERROR

```
%pyspark

avgSpeed_corr = lambda x: x.corrwith(x['avgSpeed'])
by_hour =roadtraffic.groupby(lambda x: x.hour)
by_year.apply(avgSpeed_corr)
import statsmodels.api as sm
def regression(data, yvar, xvars):
  Y = data[yvar]
  X = data[xvars]
  X['intercept'] = 1.
  result = sm.OLS(Y,X).fit()
  return result.params

by_year.apply(regression,'AAPL',['SPX'])

```

Took 0 sec. Last updated by anonymous at March 31 2017, 1:17:39 PM. (outdated)

FINISHED

```
%pyspark
grouped = roadtraffic.groupby('hour')
get_wavg = lambda g: np.average(g['avgSpeed'], weights=g['vehicleCount'])
grouped.apply(get_wavg)
```

```
07    49.978347
```

```
08    51.197955
09    50.778213
10    50.563207
11    50.482665
12    49.883909
13    47.909229
14    47.336310
15    50.606387
16    53.290080
17    54.765656
18    55.576947
19    55.817952
20    55.548410
21    56.290607
22    57.216225
23    58.958561
dtype: float64
```

Took 12 sec. Last updated by anonymous at March 31 2017, 1:26:11 PM.

ERROR

```
%pyspark
import statsmodels.api as sm

def regression(data, yvar, xvars):
  Y = data[yvar]
  X = data[xvars]
  X['intercept'] = 1.
  result = sm.OLS(Y,X).fit()
  return result.params
by_hour.apply(regression,'avgSpeed',['vehicleCount'])
```

Took 16 sec. Last updated by anonymous at March 31 2017, 1:25:27 PM.

FINISHED

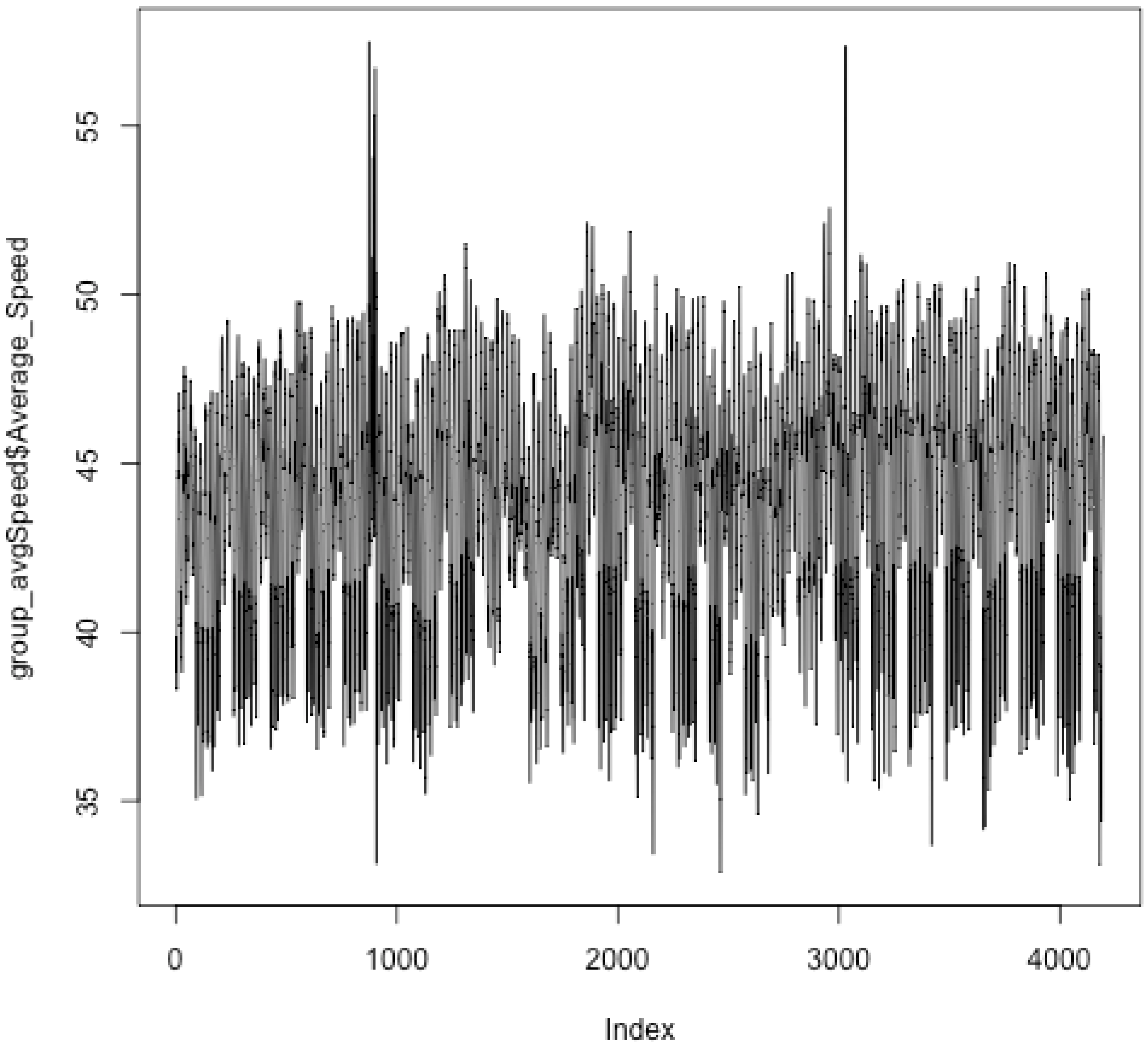
```
%r
setwd("~/Downloads/traffic_feb_june")
files = list.files(getwd())
library(readr)
library(dplyr)
tbl = lapply(files, read_csv) %>% bind_rows()
summary(tbl)
```

```
 Mode  :character   Median :  80.0   Median : 43.00   Median : 825.0
                    Mean   : 106.8   Mean   : 43.94   Mean   : 827.9
                    3rd Qu.: 116.0   3rd Qu.: 57.00   3rd Qu.: 938.0
                    Max.   :3656.0   Max.   :150.00   Max.   :1058.0
 medianMeasuredTime   TIMESTAMP                     vehicleCount
 Min.   :   0.0     Min.   :2014-02-13 11:30:00   Min.   :  0.000
 1st Qu.:  53.0     1st Qu.:2014-03-30 06:30:00   1st Qu.:  0.000
 Median :  80.0     Median :2014-05-11 09:55:00   Median :  1.000
 Mean   : 106.8     Mean   :2014-05-30 14:03:59   Mean   :  3.182
 3rd Qu.: 116.0     3rd Qu.:2014-08-18 12:35:00   3rd Qu.:  4.000
 Max.   :3656.0     Max.   :2014-09-30 23:55:00   Max.   :121.000
      _id              REPORT_ID
 Min.   :  189942   Min.   :158324
 1st Qu.: 5568083   1st Qu.:184621
 Median :10944545   Median :190770
 Mean   :13157424   Mean   :190461
 3rd Qu.:22800528   3rd Qu.:197977
 Max.   :28064531   Max.   :210199
 Max.    .28064321    Max.    .210199
```

Took 1 min 31 sec. Last updated by anonymous at April 23 2017, 11:26:27 PM.

FINISHED

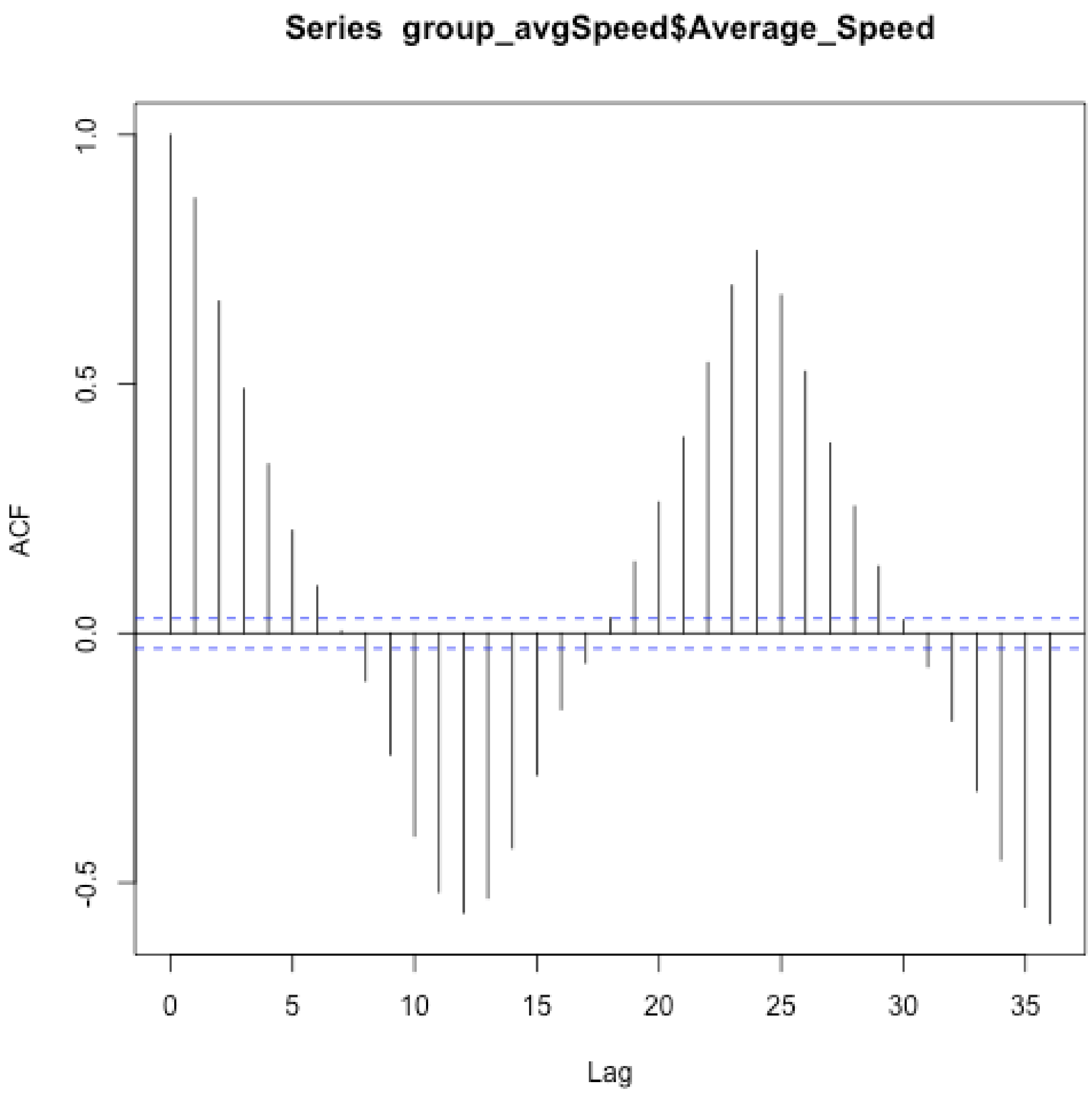
```
%r
tbl$hour<- as.POSIXlt(tbl$TIMESTAMP)$hour
tbl$Date<- as.Date(tbl$TIMESTAMP)

group_avgSpeed <- aggregate(tbl$avgSpeed, by = list(tbl$Date, tbl$hour), FUN = mean)
colnames(group_avgSpeed) <- c("Date", "Hour", "Average_Speed")
group_avgSpeed <- group_avgSpeed[with(group_avgSpeed, order(Date, Hour)), ]
group_avgSpeed$Date_Hour <- paste(as.character(group_avgSpeed$Date), paste(as.character(group_

plot(group_avgSpeed$Average_Speed, type = 'l')
```



Took 4 min 18 sec. Last updated by anonymous at April 23 2017, 11:31:51 PM.

```
%r
acf(group_avgSpeed$Average_Speed)
```

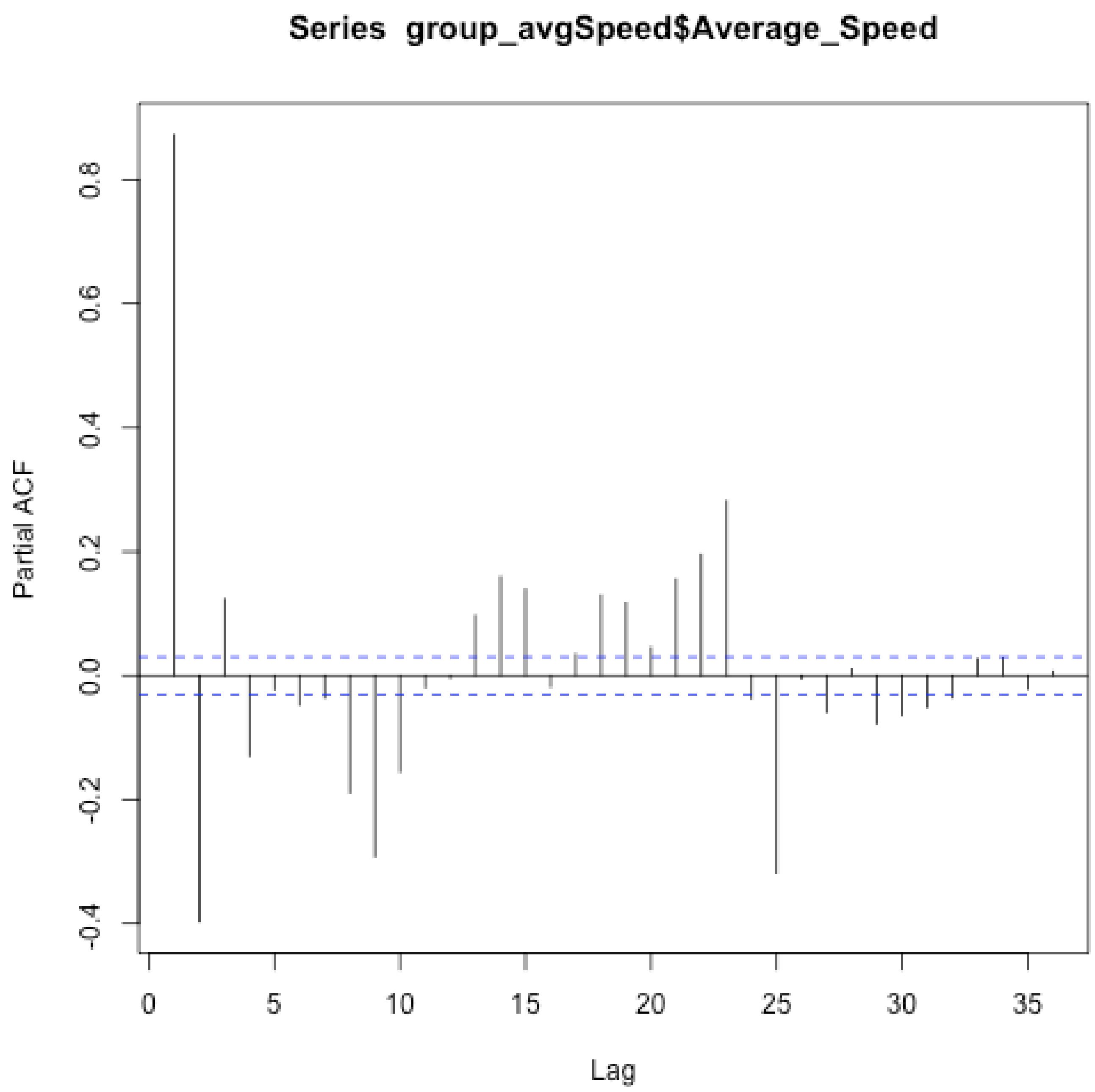
FINISHED



Took 0 sec. Last updated by anonymous at April 23 2017, 11:32:37 PM.

```
%r
pacf(group_avgSpeed$Average_Speed)
```

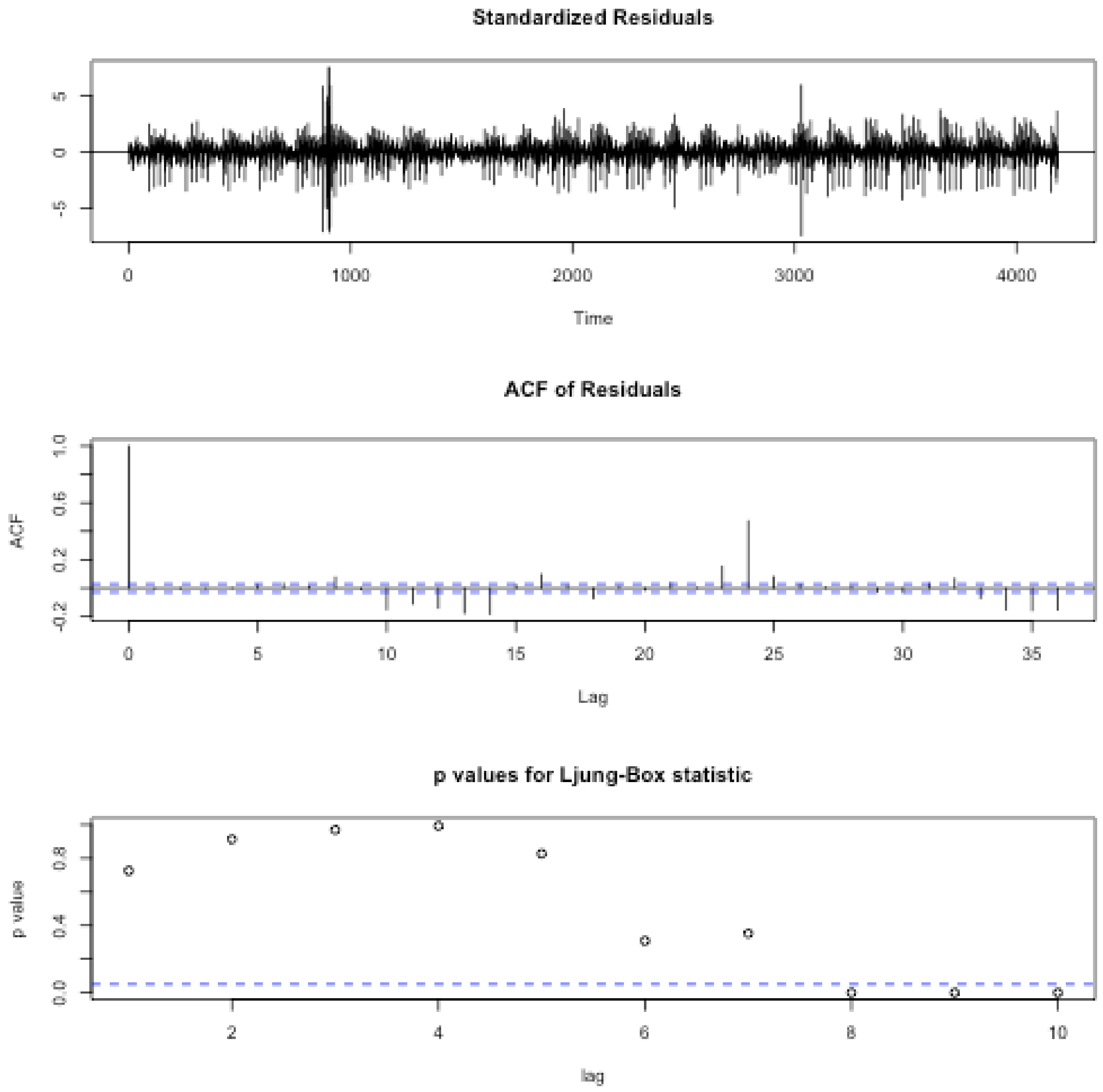
FINISHED



Took 1 sec. Last updated by anonymous at April 23 2017, 11:32:52 PM.

```
%r
library(forecast)
train_avgSpeed <- group_avgSpeed[-c(4187:4198),]
fit_avgSpeed <- arima(train_avgSpeed$Average_Speed,order = c(4,2,7))
tsdiag(fit_avgSpeed)
```

FINISHED



Took 4 sec. Last updated by anonymous at April 23 2017, 11:36:04 PM.

```
%r
summary(fit_avgSpeed)
```

FINISHED

```
Call:
arima(x = train_avgSpeed$Average_Speed, order = c(4, 2, 7))
Coefficients:
         ar1      ar2     ar3     ar4      ma1     ma2      ma3      ma4
      1.1006  -1.0084  0.3534  0.1155  -1.7851  1.2711  -0.4577  -0.4285
s.e.  0.0891   0.1336  0.1305  0.0682   0.0878  0.1929   0.1847   0.0964
         ma5     ma6     ma7
      0.2477  0.0539  0.1061
s.e.  0.0459  0.0522  0.0218
sigma2 estimated as 2.471:  log likelihood = -7833.14,  aic = 15690.29
Training set error measures:
```

```
                         ME      RMSE       MAE         MPE      MAPE       MASE
Training set -0.005745859 1.571492 1.082008 -0.1138453 2.518093 0.9114698
                      ACF1
Training set -0.005423384
```

Took 0 sec. Last updated by anonymous at April 23 2017, 11:36:08 PM.

FINISHED

```
%r
pred_speed<- predict(fit_avgSpeed, n.ahead=12)
TS_result<-cbind(pred_speed$pred[1:12], group_avgSpeed$Average_Speed[c(4187:4198)])
colnames(TS_result) <- c("Predicted", "Actual")
TS_result
TS_result<-as.data.frame(TS_result)
plot(TS_result$Actual,type ='l', col = 'red', ylim=c(25,50), xlab = 'Time', ylab = 'Average Sp
lines(TS_result$Predicted,type ='l', col = 'blue')
legend('topright', names(TS_result) , lty=1, col=c('blue', 'red'), bty='n', cex=.75)
```

```
   Predicted   Actual
```

[1,] 38.79589 38.83837 [2,] 37.89249 36.03330 [3,] 37.53761 34.39095 [4,] 38.60748 37.37238 [5,] 40.15553 41.74689 [6,] 40.58624 42.28561 [7,] 39.73291 43.07860 [8,] 38.92659 43.89440 [9,] 39.12721 44.53928 [10,] 39.80582 45.30594 [11,] 39.86345 45.46067 [12,] 39.11689 45.83157