Transportation Usage Forecasting

This project involves predicting the number of passengers on a new transit system based on historical data.

Loading packages

In [1]:
```
library(ggplot2)
library(lattice)
library(caret)
library(dplyr)
library(stringr)
library(lubridate)
library(readr)
library(tidyr)
options(repos='https://cran.cnr.berkeley.edu/')
install.packages('EnvStats')
install.packages('aTSA')
library(forecast)
library(EnvStats)
library(aTSA)
```

...

Data Exploration

Load data set.

In [2]:
```
df <- read_csv('C:/Datasets/TrafficTrain.csv')
head(df,10)
```

```
Parsed with column specification:
cols(
  ID = col_integer(),
  Datetime = col_character(),
  Count = col_integer()
)
```

ID	Datetime	Count
0	25-08-2012 00:00	8
1	25-08-2012 01:00	2
2	25-08-2012 02:00	6
3	25-08-2012 03:00	2
4	25-08-2012 04:00	2
5	25-08-2012 05:00	2
6	25-08-2012 06:00	2
7	25-08-2012 07:00	2
8	25-08-2012 08:00	6
9	25-08-2012 09:00	2

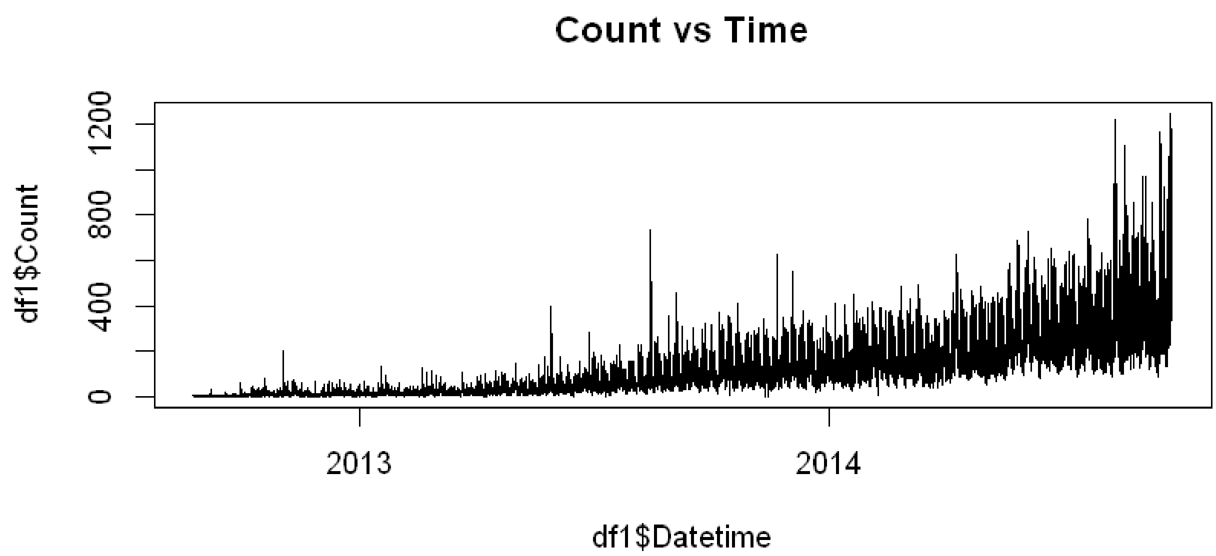
The Datetime column contains strings. These will be separated and a true date time column is generated in order to calculate day of the week.

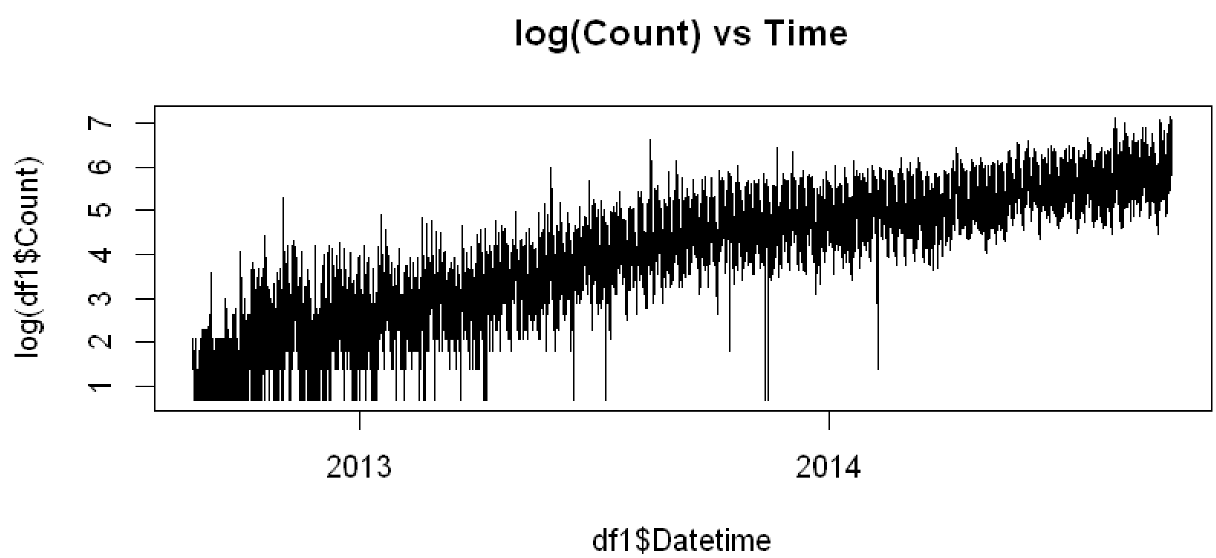
In [3]:
```
df1 <- separate(df, Datetime, into=c('d','m','y','hr','min'),sep='[- :]')
df1 <- transform(df1, d=as.numeric(d),m=as.numeric(m),y=as.numeric(y),hr=as.numeric(
df1$Datetime <- with(df1,make_datetime(y,m,d,hr,min))
df1$wd <- wday(df1$Datetime)
head(df1,10)
```

ID	d	m	y	hr	min	Count	Datetime	wd
0	25	8	2012	0	0	8	2012-08-25 00:00:00	7
1	25	8	2012	1	0	2	2012-08-25 01:00:00	7
2	25	8	2012	2	0	6	2012-08-25 02:00:00	7
3	25	8	2012	3	0	2	2012-08-25 03:00:00	7
4	25	8	2012	4	0	2	2012-08-25 04:00:00	7
5	25	8	2012	5	0	2	2012-08-25 05:00:00	7
6	25	8	2012	6	0	2	2012-08-25 06:00:00	7
7	25	8	2012	7	0	2	2012-08-25 07:00:00	7
8	25	8	2012	8	0	6	2012-08-25 08:00:00	7
9	25	8	2012	9	0	2	2012-08-25 09:00:00	7

Plotting of the time series shows that it is a multiplicative time series, since the amplitude of the periodic oscillations is proportional to the average count.

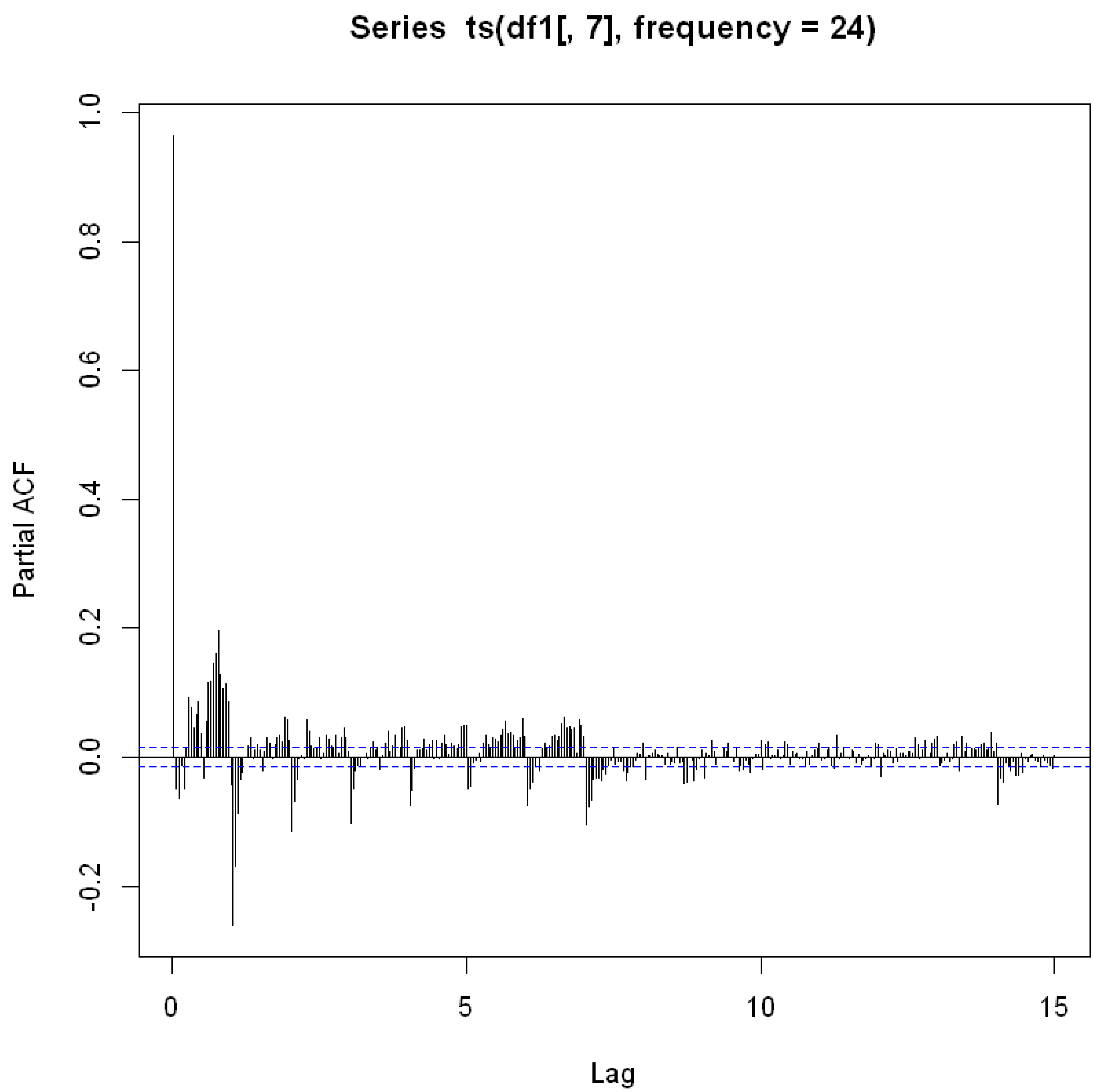
```
In [4]: par(mfrow=c(2,1))
        plot(df1$Datetime,df1$Count, type='l', main = 'Count vs Time')
        plot(df1$Datetime,log(df1$Count), type='l', main = 'log(Count) vs Time')
```





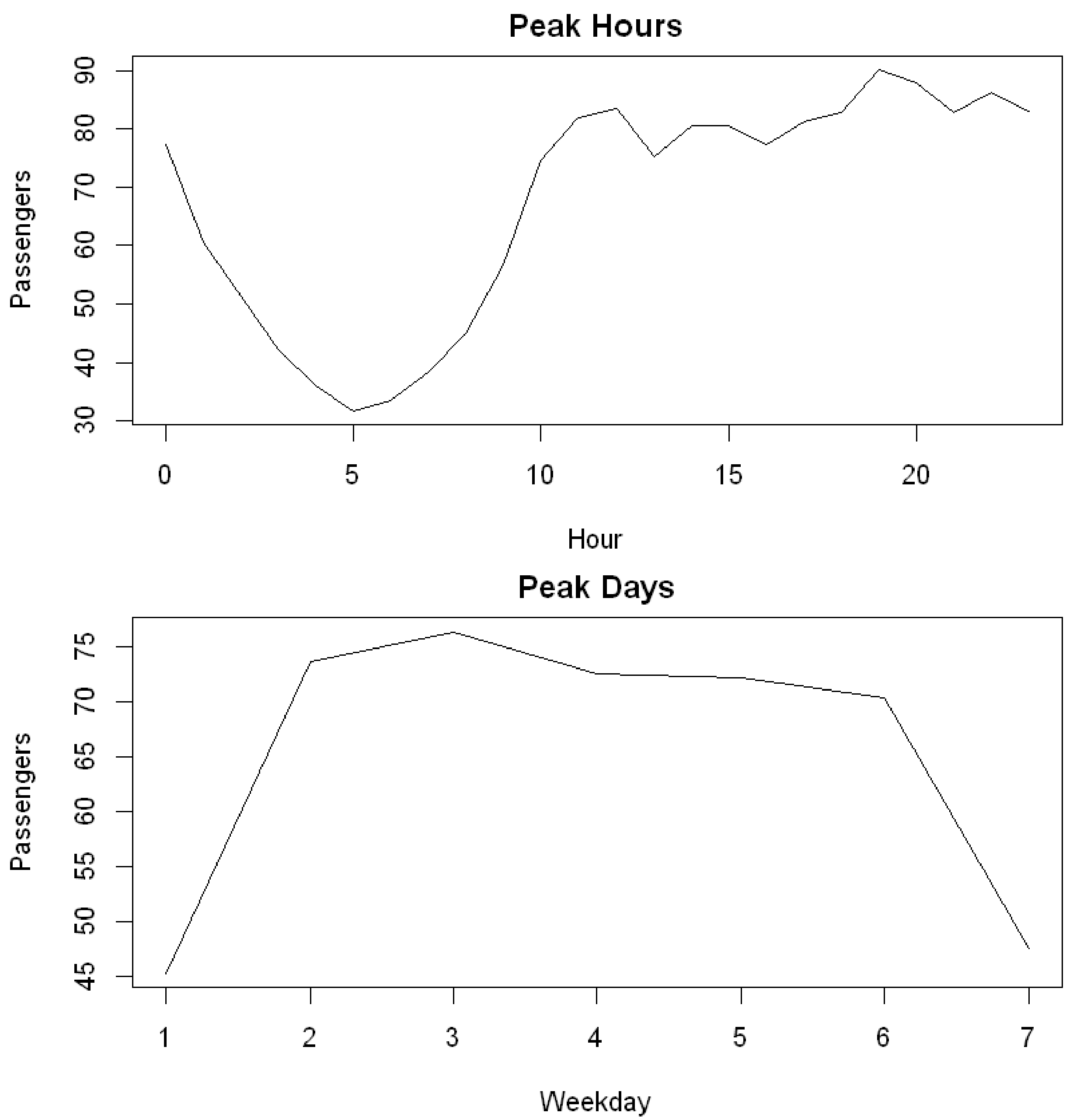
Partial autocorrelation shows strongest seasonality with period of 1 day and 7 days.

In [5]:
```
TS <- ts(df1[,7],frequency=168)
pacf(ts(df1[,7],frequency=24),lag=360)
```



Plotting of the geometric mean of the passenger count grouped by hour of the day and day of the week shows seasonal trends.

In [6]:
```
par(mfrow=c(2,1),mar=c(4,4,2,2))
plot(0:23,aggregate(df1$Count,list(df1$hr),geoMean)[,2], type='l',xlab='Hour',ylab='
plot(1:7,aggregate(df1$Count,list(df1$wd),geoMean)[,2], type='l',xlab='Weekday',ylab
```

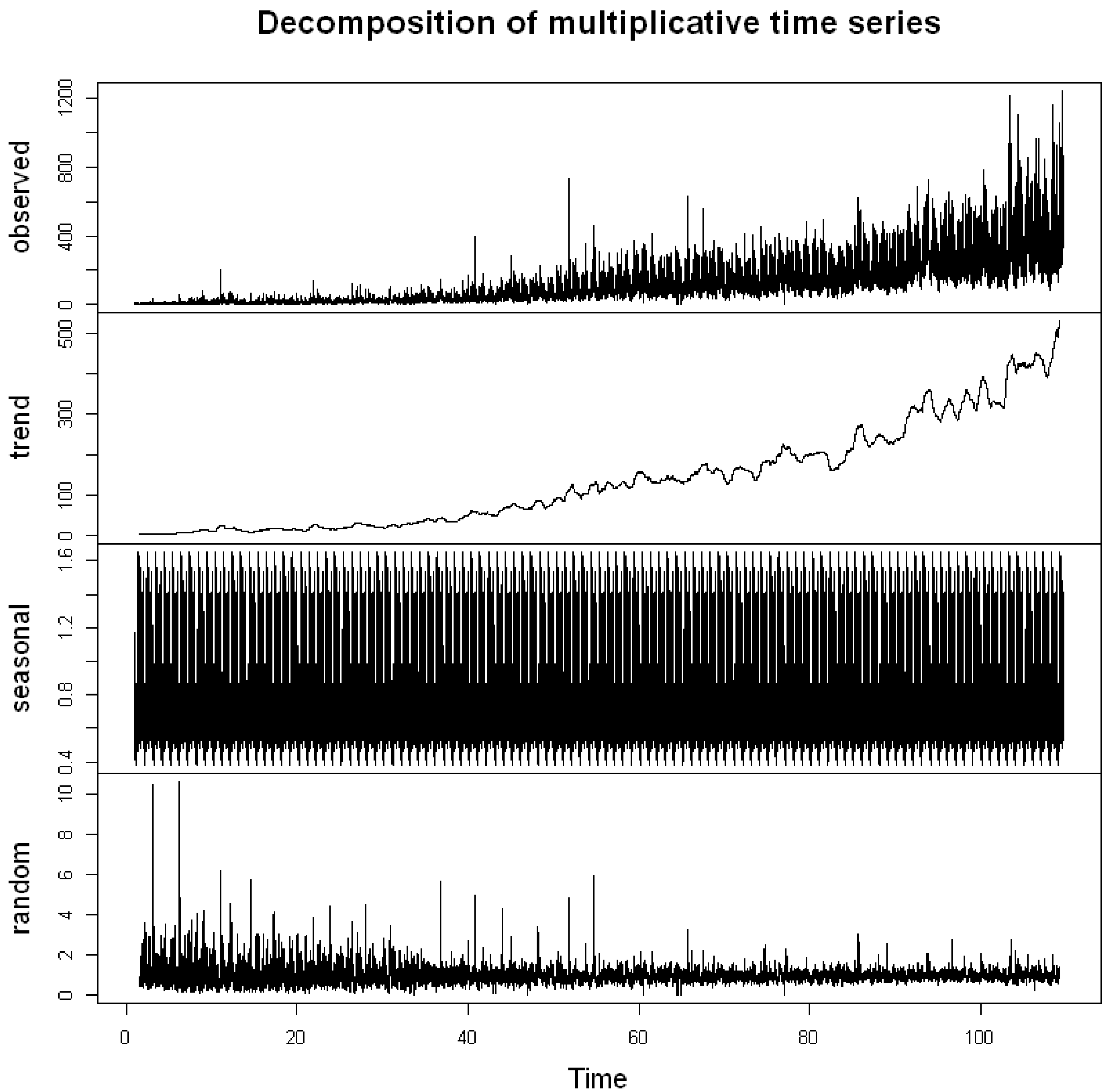


There are the fewest passengers in the early morning, and on Saturdays and Sundays.

Time Series Decomposition

The time series is decomposed into trend, seasonality, and noise as follows.

```
In [7]: decomposedTS <- decompose(TS,type='mult')
        plot(decomposedTS)
```



A table of the three components is generated.

In [8]:
```
stlTS <- stl(log(TS),s.window='periodic')
components <- exp(stlTS$time.series)
head(components,10)
```

seasonal	trend	remainder
1.2736905	3.224755	1.9477329
1.0090744	3.217392	0.6160314
0.8224904	3.210046	2.2725276
0.6620160	3.202717	0.9432851
0.5498908	3.195404	1.1382241
0.4604410	3.188109	1.3624573
0.4437631	3.180829	1.4168976
0.4971437	3.173567	1.2676531
0.5680883	3.166321	3.3356500
0.6607981	3.159092	0.9580738

Then the remainder is checked to see if it is stationary.

In [9]:
```
stationaryTS <- components[,3]
trendTS <- components[,2]
seasonTS <- components[,1]
adf.test(stationaryTS,30)
```

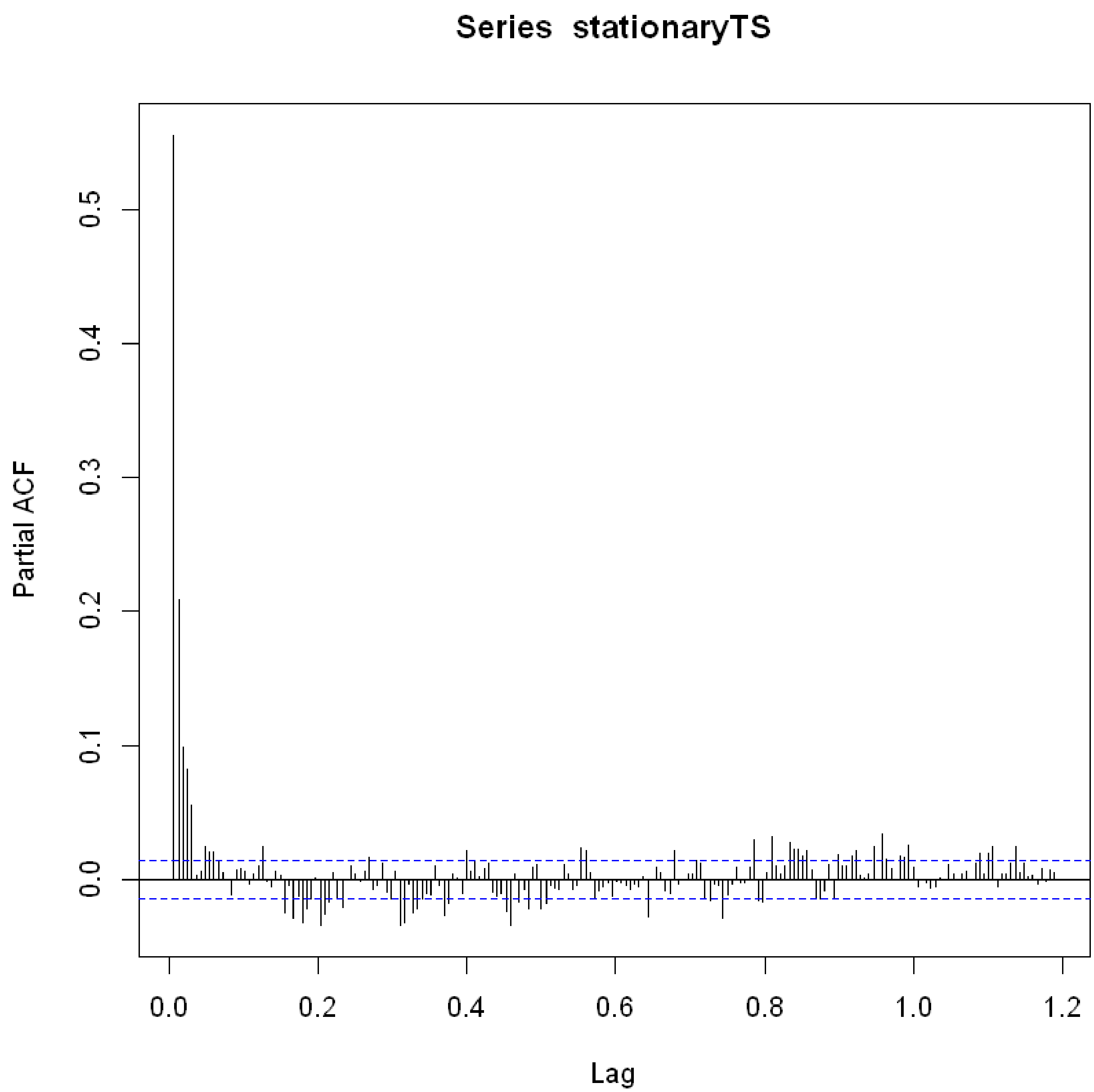
```
Augmented Dickey-Fuller Test
alternative: stationary

Type 1: no drift no trend
      lag    ADF p.value
 [1,]   0 -26.26    0.01
 [2,]   1 -17.64    0.01
 [3,]   2 -14.32    0.01
 [4,]   3 -12.11    0.01
 [5,]   4 -10.71    0.01
 [6,]   5 -10.07    0.01
 [7,]   6  -9.47    0.01
 [8,]   7  -8.78    0.01
 [9,]   8  -8.27    0.01
[10,]   9  -7.73    0.01
[11,]  10  -7.32    0.01
[12,]  11  -7.04    0.01
[13,]  12  -6.79    0.01
[14,]  13  -6.61    0.01
[15,]  14  -6.38    0.01
[16,]  15  -6.10    0.01
[17,]  16  -5.87    0.01
[18,]  17  -5.71    0.01
[19,]  18  -5.52    0.01
[20,]  19  -5.31    0.01
[21,]  20  -5.05    0.01
[22,]  21  -4.96    0.01
[23,]  22  -4.85    0.01
[24,]  23  -4.69    0.01
[25,]  24  -4.59    0.01
[26,]  25  -4.64    0.01
[27,]  26  -4.53    0.01
[28,]  27  -4.58    0.01
[29,]  28  -4.52    0.01
[30,]  29  -4.55    0.01
Type 2: with drift no trend
      lag   ADF p.value
 [1,]   0 -72.3    0.01
 [2,]   1 -51.6    0.01
 [3,]   2 -43.7    0.01
 [4,]   3 -38.3    0.01
 [5,]   4 -34.8    0.01
 [6,]   5 -33.6    0.01
 [7,]   6 -32.5    0.01
 [8,]   7 -30.8    0.01
 [9,]   8 -29.5    0.01
[10,]   9 -28.2    0.01
[11,]  10 -27.2    0.01
[12,]  11 -26.5    0.01
[13,]  12 -26.0    0.01
[14,]  13 -25.8    0.01
[15,]  14 -25.2    0.01
```

```
[16,]  15 -24.5    0.01
[17,]  16 -24.0    0.01
[18,]  17 -23.7    0.01
[19,]  18 -23.2    0.01
[20,]  19 -22.7    0.01
[21,]  20 -21.8    0.01
[22,]  21 -21.6    0.01
[23,]  22 -21.4    0.01
[24,]  23 -21.0    0.01
[25,]  24 -20.7    0.01
[26,]  25 -21.0    0.01
[27,]  26 -20.8    0.01
[28,]  27 -21.2    0.01
[29,]  28 -21.2    0.01
[30,]  29 -21.6    0.01
Type 3: with drift and trend
      lag   ADF p.value
 [1,]   0 -73.0    0.01
 [2,]   1 -52.2    0.01
 [3,]   2 -44.2    0.01
 [4,]   3 -38.8    0.01
 [5,]   4 -35.3    0.01
 [6,]   5 -34.2    0.01
 [7,]   6 -33.0    0.01
 [8,]   7 -31.3    0.01
 [9,]   8 -30.0    0.01
[10,]   9 -28.7    0.01
[11,]  10 -27.7    0.01
[12,]  11 -27.1    0.01
[13,]  12 -26.6    0.01
[14,]  13 -26.4    0.01
[15,]  14 -25.8    0.01
[16,]  15 -25.1    0.01
[17,]  16 -24.6    0.01
[18,]  17 -24.3    0.01
[19,]  18 -23.8    0.01
[20,]  19 -23.3    0.01
[21,]  20 -22.4    0.01
[22,]  21 -22.2    0.01
[23,]  22 -22.0    0.01
[24,]  23 -21.6    0.01
[25,]  24 -21.3    0.01
[26,]  25 -21.6    0.01
[27,]  26 -21.4    0.01
[28,]  27 -21.8    0.01
[29,]  28 -21.9    0.01
[30,]  29 -22.3    0.01
----
Note: in fact, p.value = 0.01 means p.value <= 0.01
```

The p values are <0.05, indicating the remainder is stationary and can be modeled with ARIMA.

In [10]:
```
pacf(stationaryTS,200)
```



The partial autocorrelation plot (x axis is in weeks) shows no large correlations except at small lag times. This means most of the seasonality has been removed during decomposition.

Time Series Modeling

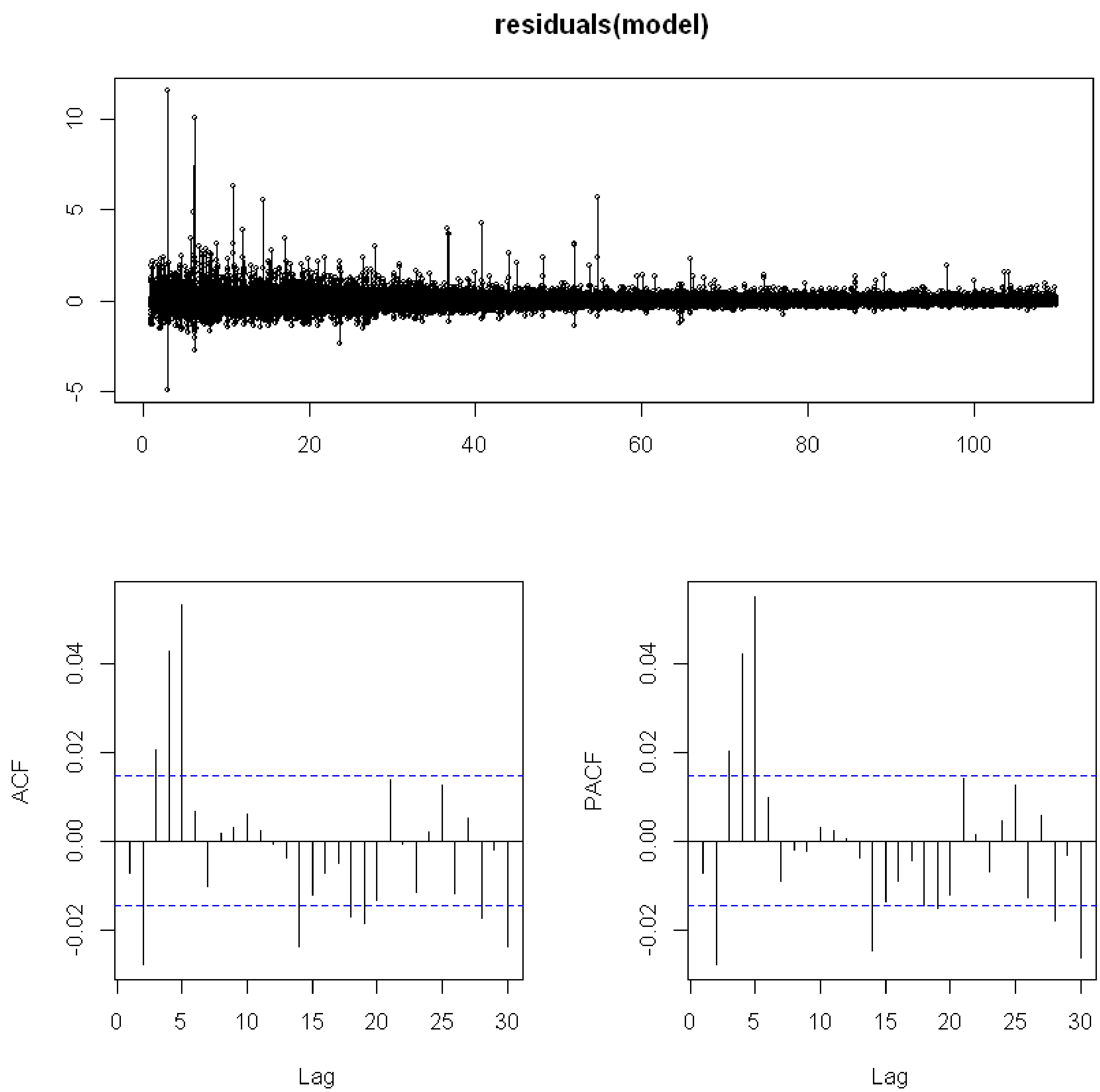
The ARIMA order parameters are determined automatically below for the stationary component.

In [11]:
```
model <- auto.arima(stationaryTS, seasonal=FALSE)
model
tsdisplay(residuals(model),lag.max=30)
```

```
Series: stationaryTS
ARIMA(2,1,1)

Coefficients:
         ar1     ar2      ma1
      0.3663  0.1428  -0.9305
s.e.  0.0112  0.0102   0.0077

sigma^2 estimated as 0.1499:  log likelihood=-8596.8
AIC=17201.61   AICc=17201.61   BIC=17232.86
```



Once again, the PACF (x in hours) shows no daily seasonality (no peaks at 24 hours).

The ARIMA model is used to forecast the noise.

In [12]:
```
model <- arima(stationaryTS, order =c(2,1,1))
prediction <- forecast(model,5112)
```

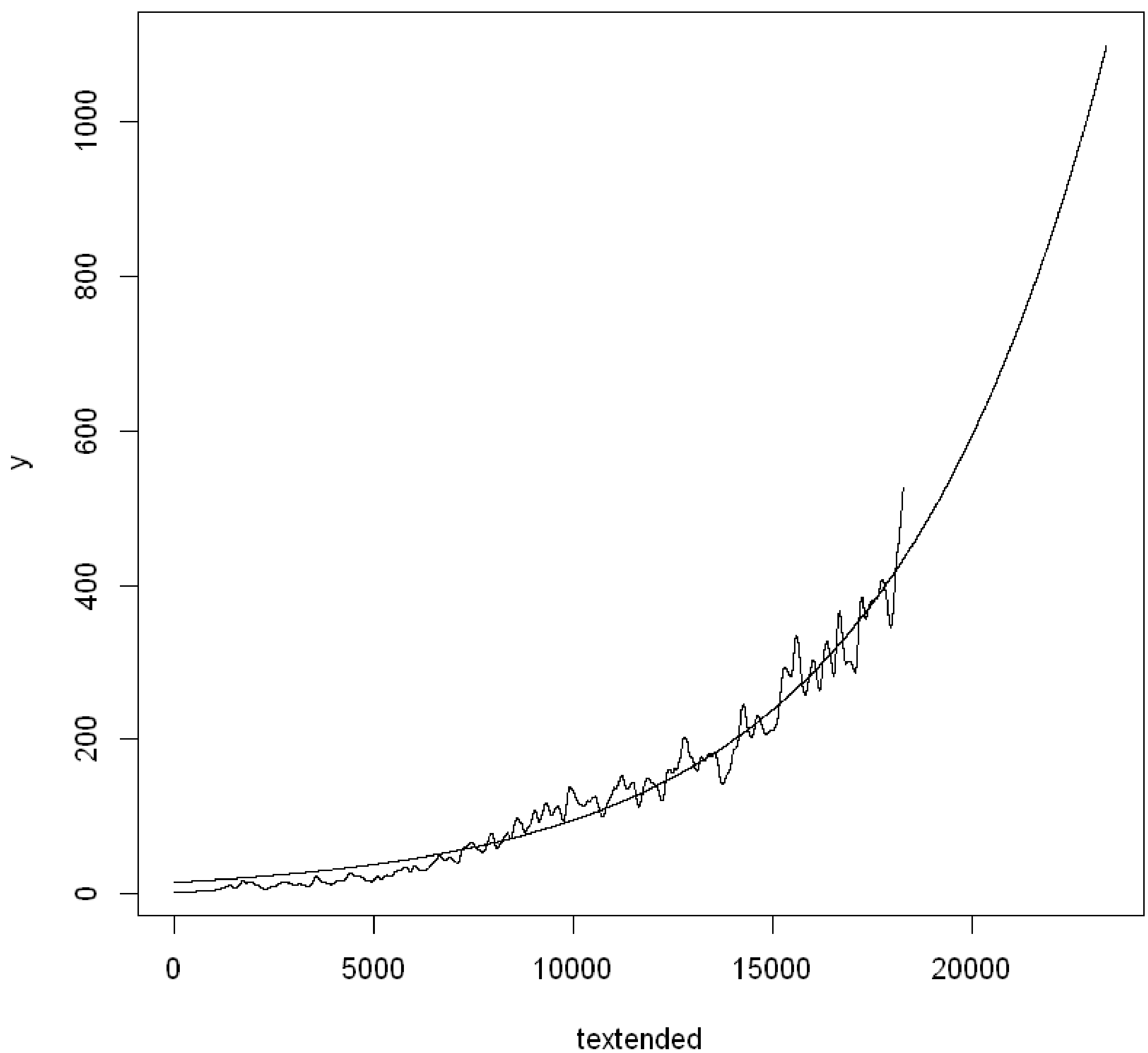
```
Forecast for univariate time series:
      Lead Forecast   S.E    Lower Upper
18289    1    0.847 0.387  0.08852  1.61
18290    2    0.889 0.422  0.06140  1.72
18291    3    0.922 0.446  0.04714  1.80
18292    4    0.940 0.458  0.04170  1.84
18293    5    0.951 0.466  0.03740  1.86
18294    6    0.957 0.472  0.03290  1.88
18295    7    0.962 0.476  0.02797  1.90
18296    8    0.964 0.480  0.02265  1.91
18297    9    0.965 0.484  0.01707  1.91
18298   10    0.966 0.487  0.01134  1.92
18299   11    0.967 0.490  0.00552  1.93
18300   12    0.967 0.494 -0.00034  1.93
18301   13    0.967 0.497 -0.00621  1.94
18302   14    0.967 0.500 -0.01208  1.95
18303   15    0.968 0.503 -0.01793  1.95
18304   16    0.968 0.506 -0.02376  1.96
18305   17    0.968 0.509 -0.02955  1.96
```

Next, the trend component is modeled using non-linear least squares regression (a simple exponential model).

In [13]:
```
t <- 1:length(trendTS)
typeof(trendTS)
expmodel <- nls(trendTS~b*(a^t))
A<-coef(expmodel)[[2]]
B<-coef(expmodel)[[1]]
textended <-1:(length(trendTS)+5112)
y <- B*(A^textended)
plot(textended,y,type='l')
lines(t,trendTS)
```

'double'

```
Warning message in nls(trendTS ~ b * (a^t)):
"No starting values specified for some parameters.
Initializing 'b', 'a' to '1.'.
Consider specifying 'start' or using a selfStart model"
```



Finally, the seasonality component can be forecasted by just cycling through it repeatedly. However, first it must be determined at which point in the cycle the data starts and ends.

In [14]:
```
head(df1,10)
tail(df1,10)
```

ID	d	m	y	hr	min	Count	Datetime	wd
0	25	8	2012	0	0	8	2012-08-25 00:00:00	7
1	25	8	2012	1	0	2	2012-08-25 01:00:00	7
2	25	8	2012	2	0	6	2012-08-25 02:00:00	7
3	25	8	2012	3	0	2	2012-08-25 03:00:00	7
4	25	8	2012	4	0	2	2012-08-25 04:00:00	7
5	25	8	2012	5	0	2	2012-08-25 05:00:00	7
6	25	8	2012	6	0	2	2012-08-25 06:00:00	7
7	25	8	2012	7	0	2	2012-08-25 07:00:00	7
8	25	8	2012	8	0	6	2012-08-25 08:00:00	7
9	25	8	2012	9	0	2	2012-08-25 09:00:00	7

	ID	d	m	y	hr	min	Count	Datetime	wd
18279	18278	25	9	2014	14	0	616	2014-09-25 14:00:00	5
18280	18279	25	9	2014	15	0	686	2014-09-25 15:00:00	5
18281	18280	25	9	2014	16	0	654	2014-09-25 16:00:00	5
18282	18281	25	9	2014	17	0	622	2014-09-25 17:00:00	5
18283	18282	25	9	2014	18	0	680	2014-09-25 18:00:00	5
18284	18283	25	9	2014	19	0	868	2014-09-25 19:00:00	5
18285	18284	25	9	2014	20	0	732	2014-09-25 20:00:00	5
18286	18285	25	9	2014	21	0	702	2014-09-25 21:00:00	5
18287	18286	25	9	2014	22	0	580	2014-09-25 22:00:00	5
18288	18287	25	9	2014	23	0	534	2014-09-25 23:00:00	5

The cycle length is 24 x 7 = 168 hours (data points) long. It begins on a Saturday, and the data ends on a Thursday. Therefore the repeated cycle must start on a Friday (the last 24 hours of the 168 hour cycle) and run until the 144th hour of the cycle.

Then the stationary noise, trend, and seasonality components can be multiplied together to generate the final prediction. But, first the model will be evaluated against actual data.

In [87]:

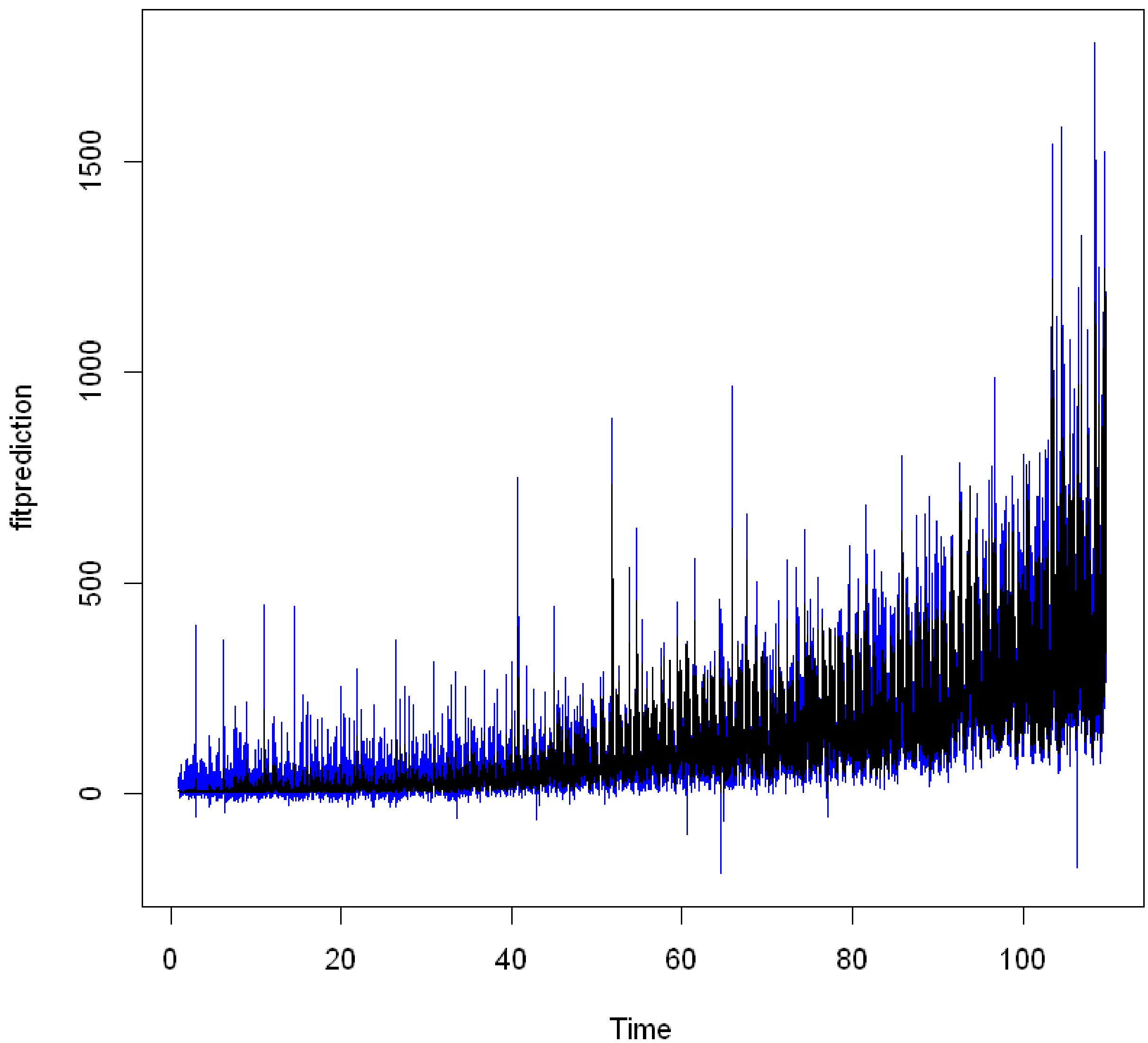
```
tfit <-1:length(trendTS)
stationaryfit <-residuals(model)+stationaryTS
fitprediction <- B*(A^tfit)*stationaryfit*seasonTS
plot(fitprediction,type='l',col='blue')
lines(TS)
'RMSE'
sqrt(sum((fitprediction-TS)^2)/(length(TS)-(168+2+4)))
'R^2'
1-sum((fitprediction-TS)^2)/sum((TS-mean(TS))^2)
```

'RMSE'

41.716041603315

'R^2'

0.926811058564574



The fit is decent. We will continue generating the prediction using this model.

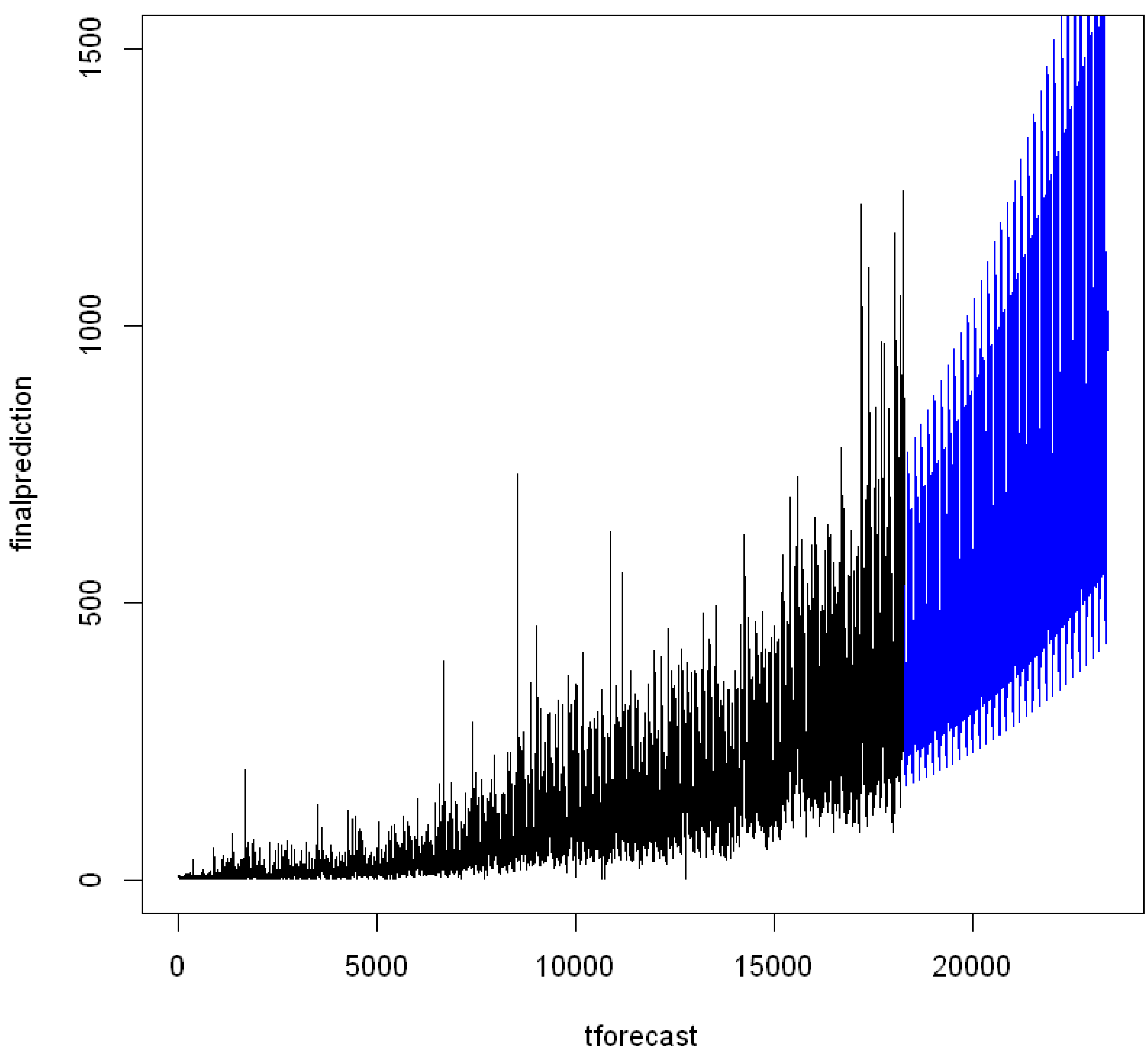
In [15]:
```
shiftedseason <- c(tail(seasonTS,24),seasonTS[1:144])

tforecast <-length(trendTS):(length(trendTS)+5112)

finalprediction <- B*(A^tforecast)*prediction[,2]*shiftedseason

plot(tforecast,finalprediction,type='l',xlim=c(0,23399),ylim=c(0,1500),col='blue')
lines(df$ID,df$Count)
```

```
Warning message in B * (A^tforecast) * prediction[, 2]:
"longer object length is not a multiple of shorter object length"Warning message in
B * (A^tforecast) * prediction[, 2] * shiftedseason:
"longer object length is not a multiple of shorter object length"
```



In [16]:
```
forecasttable <- as.data.frame(tforecast)
forecasttable$predict <- finalprediction
write.csv(forecasttable,'C:/Datasets/TrafficPredict.csv')
```

Cross Validation

Cross validation is be performed using subsets of the full data set as the training set and subsequent data points as the test set. The test set size is approximately half the training set size. The training set always starts at the beginning of the full data set. Cross validation is performed with 7 different splits, with the training set representing ~ 10, 20, 30,... 70% of the total data set respectively, where the test set is 5, 10, 15, ... 35% of the total data set that immediately follows the training set.

All the procedures performed above using the full set are written as a function.

In [120]:
```
CV <-function(dftrain, dftest){
    TS2 <- ts(dftrain,frequency=168)
    stlTS2 <- stl(log(TS2),s.window='periodic')
    components2 <- exp(stlTS2$time.series)
    stationaryTS2 <<- components2[,3]
    trendTS2 <- components2[,2]
    seasonTS2 <- components2[,1]

    model2 <- arima(stationaryTS2, order =c(2,1,1),)
    prediction2 <- forecast(model2,length(dftest))

    t2 <- 1:length(trendTS2)
    expmodel2 <- nls(trendTS2~b*(a^t2))
    A2 <-coef(expmodel2)[[2]]
    B2 <-coef(expmodel2)[[1]]

    shiftedseason2 <- c(tail(seasonTS2,24),seasonTS2[1:144])
    tforecast2 <-length(trendTS2):(length(trendTS2)+length(dftest))
    finalprediction2 <- B2*(A2^tforecast2)*prediction2[,2]*shiftedseason2
    plot(finalprediction2,type='l',col='blue')
    lines(dftest)
    return(c((sqrt(sum((finalprediction2-dftest)^2)/(length(dftest)-168-4-2))),(1-su
}
```

Cross validation is performed as follows. The number of samples int he training set follows a formula that ensures the weekdays are aligned the same way in the training sets as they are in the full set (start on Sunday, end on Thrusday).

In [129]:
```
r2<-c()
rmse <- c()
for(i in 1:7){
    number <- i*10*168+144
    scores <- CV(df1[1:number,7],df1[(number+1):(number+1+number%/%2),7])
    rmse <<- append(rmse, scores[1])
    r2 <<- append(r2, scores[2])
}
```

...

The results are shown below:

In [130]:
```
r2
```

-5.75552522261993 -0.0328853084873488 0.452468579606861 0.497498473369978 -14.7137622265676 -5.75166059164056 -0.405244170550902

In [131]:
```
rmse
```

30.8275942505101 15.0012949724756 23.8186426785431 46.541835945809 313.389806692917 255.955051640632 177.616306090138

The R squared values indicate that the fit is better in the middle than at the beginning or end, where the average of the actual results is a better fit than the forecast. This indicates large range deviations from the model (long range fluctuations), which may be random or may be due to an inadequate trend equation.

It is evident from the plots of predicted vs actual data that the predictions drift away from the actual data in these cases. For the first CV sets, there likely wasn't enough data to estimate the trend equation, since the data is relatively flat there. For the last CV sets, it looks like the actual behavior starts to deviate from the trend equation, but then starts getting closer to the trend equation at the end.

From the comparison of the trend equation and the actual trend component of the full time series, it is evident that there is some long range (perhaps random) fluctuation, which may be difficult to forecast no matter what equation is used to model the trend. Models with more parameters may make the prediction error worse due to overfitting. Since the current trend model only has 2 paramters and matches the curvature of the data better than a linear model (also 2 parameters) would, the model will not be adjusted any further. In addition, despite the poor R squared values, the current RMSE is still fairly good considering that by the end of the data set the passenger count can fluctuate up to about 1000.