Analysis of Traffic Through Santa Rosa Fastrak

Clark Fitzgerald, Oliver Paisley

March 9, 2015

Abstract

This is a paper about cleaning and analyzing data generated by the California highway toll system called Fastrak. There are three main sections to this paper. The first section is an extensive preparation and cleaning of the data. The second section is a time series analysis of the data, including predicition and a spectral analysis. The third section consists of a discussion of the analysis in section two and a conclusion of our findings.

Introduction

Fastrak is the California highway toll system. Thousands of cars pass through Fastrak stations each day, and the data generated is incredibly valuable. By analyzing the number of cars that pass through Fastrak we can gain insight and knowledge into the behavior of traffic flow throughout California. This information would be useful for any entity interested in traffic flow, of which there are many. For example, any type of road work project would benefit from this knowledge, as they could determine when the optimal time is to work.

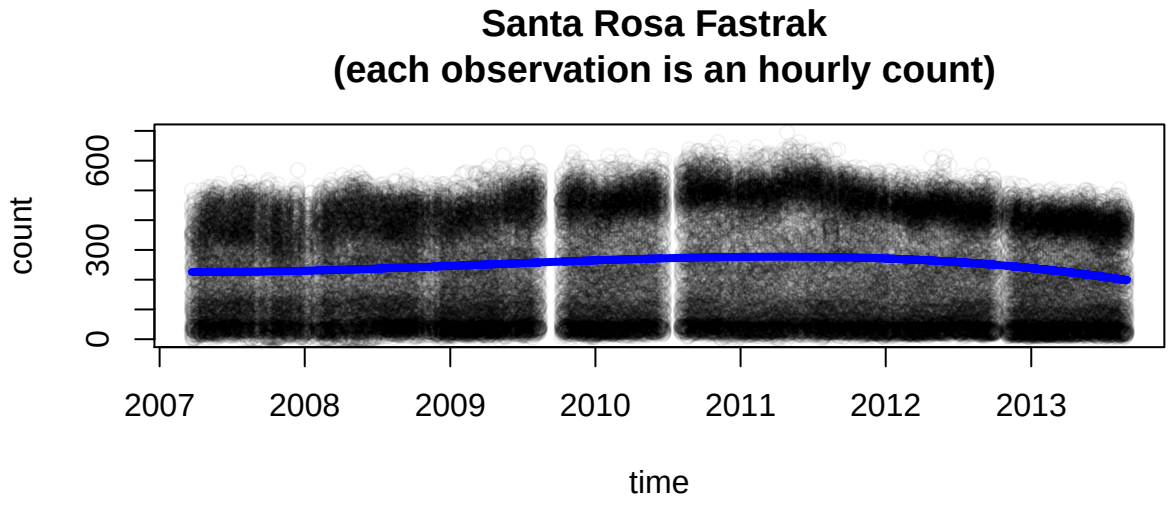
This paper is organized into three main sections. The data was quite dirty, so the first section is an exhaustive cleaning and preparation of the data. The second section includes an analyis of our residual series, and the third and final section is a discussion of the analysis done and an overall conclusion.

Data Description

The data consists of counts of the number of cars that pass through a given Fastrak station per hour. The data was acquired through the Caltrans Performance Measurement System, which is apart of the California Department of Transportation. We have counts for roughly 6 years worth of data across 150 stations, and there are about 8 million observations.

Here is what the data looks like for the Fastrak station in Santa Rosa (station 4300).

time	station	count	year	month	weekday	hour
2007-03-23 03:00:00	4300	32	2007	March	Friday	03
2007-03-23 04:00:00	4300	35	2007	March	Friday	04
2007-03-23 05:00:00	4300	103	2007	March	Friday	05
2007-03-23 06:00:00	4300	231	2007	March	Friday	06
2007-03-23 07:00:00	4300	357	2007	March	Friday	07
2007-03-23 08:00:00	4300	411	2007	March	Friday	08



This plot indicates that a cubic polynomial will adequately approximate the shape of the long term trend. A cubic polynomial fit is preferable to a fit based on the categorical variable *year* since the cubic polynomial is smooth, and therefore it will not contribute to any discontinuities in the time series.

Based on experience and common sense, we know that there are a few elements that primarily determine traffic flow. These elements include:

- The time of day.
- The day of the week.

There is also the possibility of monthly or annual trends. Based on the above, we have developed the following general linear model:

$$count = time + time^2 + time^3 + month + weekday + hour + weekday : hour + \epsilon$$

time is a continuous variable, whereas *hour*, *weekday*, and *month* are categorical variables. $weekday : hour$ represents the interaction between *weekday* and *hour*. Note that the three terms in the *time* polynomial are orthogonal, which is the default in the **poly()** function.

We can represent our model in terms of the classic linear time series decomposition.

Let $Y_t = count$,
$m_t = time + time^2 + time^3$,
$s_t = month + weekday + hour + weekday : hour$, and
$X_t = \epsilon$.

This allows us to write our model as

$$Y_t = m_t + s_t + X_t.$$

We can now do our time series analyis on the residual series (X_t).

Data Analysis

Preparation

Significant time was spent preparing this data. There were three major issues:

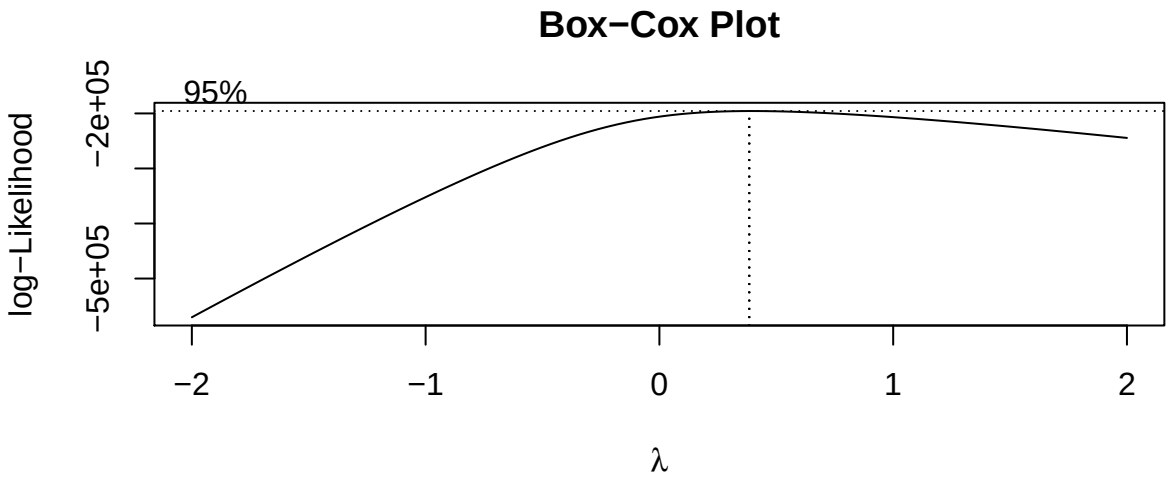
- Missing data.
- Too many counts of zero.
- A month long period where the traffic was double what it should have been across all stations.

We dealt with these issues by writing a small library of tested functions to prepare the data in a disciplined, repeatable way.

Since we have an abundance of data, we dropped all of the observations that gave us major issues. It would be certainly be possible to interpolate or backcast to fill in the missing data as well.

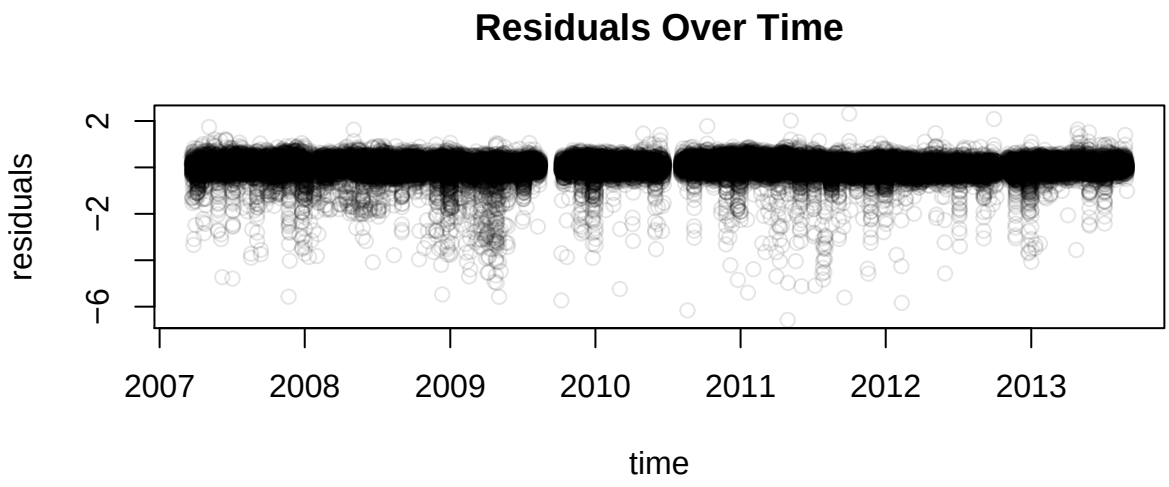
Variance Stabilizing Transform

The Box-Cox method indicates using the cube root function on the counts as a variance stabilizing transformation. Therefore, we will be using this transformation for further analysis.

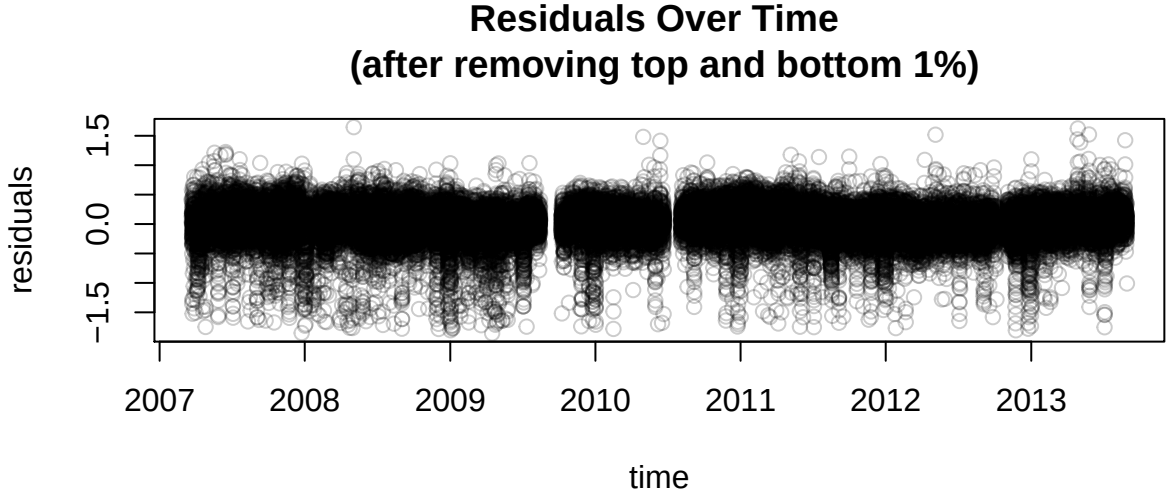


Outlier Detection and Removal

Our initial plot identified quite a few suspicious observations where the sensor may have stopped working.



There are quite a lot of long tails. We are going to remove the bottom 1% of residuals in hopes of having more consistency. For symmetry we are also going to remove the top 1%.



This removed about 500 values below the bottom threshold and about 5 values above the top threshold. There are still tails, but they are not as severe.

Interpretation

Inspecting the corresponding ANOVA table tells us much about the relative sources of variation in the count data. To determine the sources of variation in our count data we will look at the corresponding $R_a^2 dj$ value and ANOVA table of our model. From **R** we determined that $R_a^2 dj$ is equal to 0.972. Therefore, this model

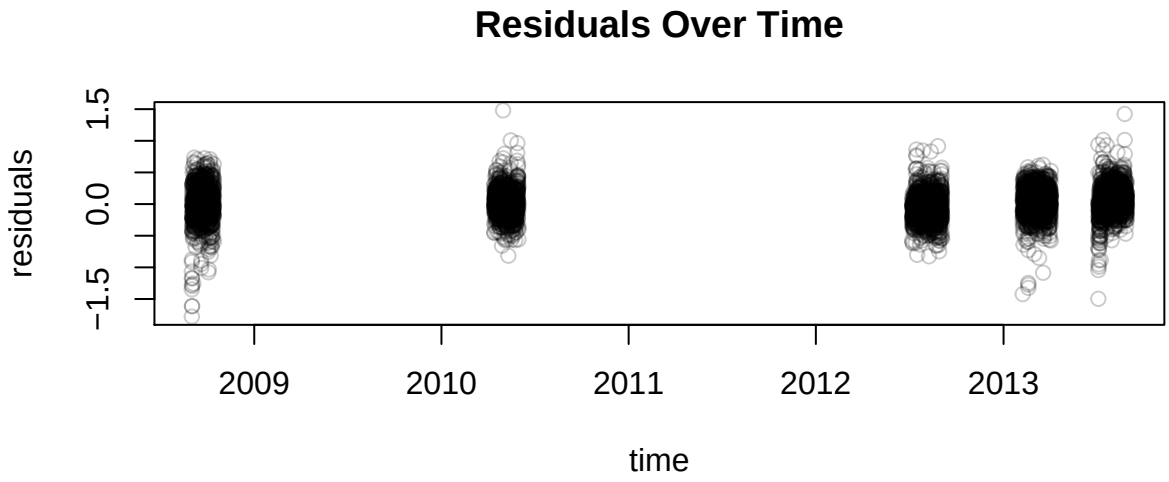
This model accounts for 97% of the variation in the cube root of counts.

	Df	Sum Sq	Mean Sq	F value	Pr(>F)	percent
hour	23	140140.3886	6093.06037	68109.3102	0	88.7670198

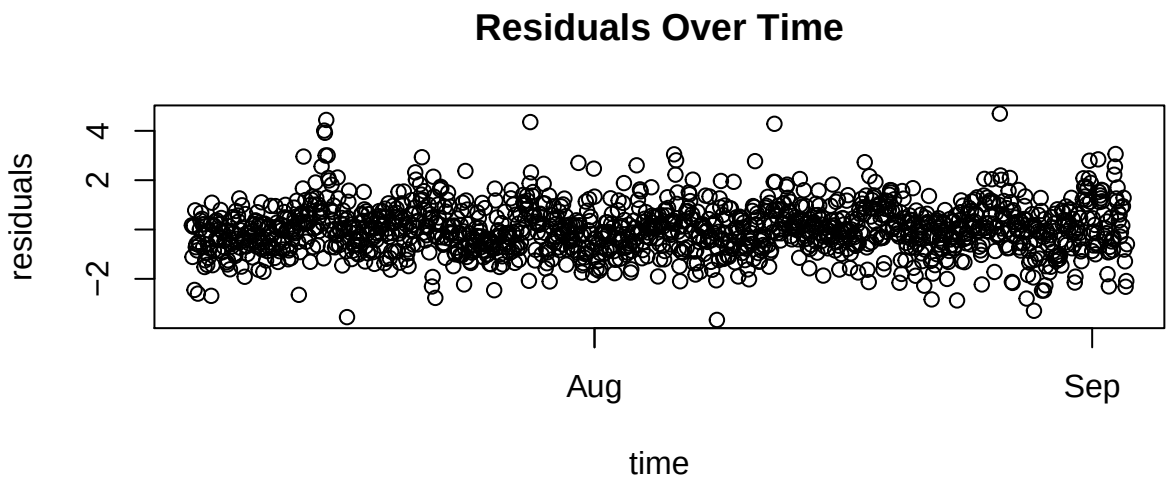
	Df	Sum Sq	Mean Sq	F value	Pr(>F)	percent
weekday	6	5972.1541	995.35901	11126.2997	0	3.7828518
month	11	217.7513	19.79558	221.2785	0	0.1379270
poly(time, 3)	3	1539.9660	513.32199	5738.0043	0	0.9754375
hour:weekday	138	5601.9786	40.59405	453.7675	0	3.5483771
Residuals	49208	4402.1488	0.08946	NA	NA	2.7883869

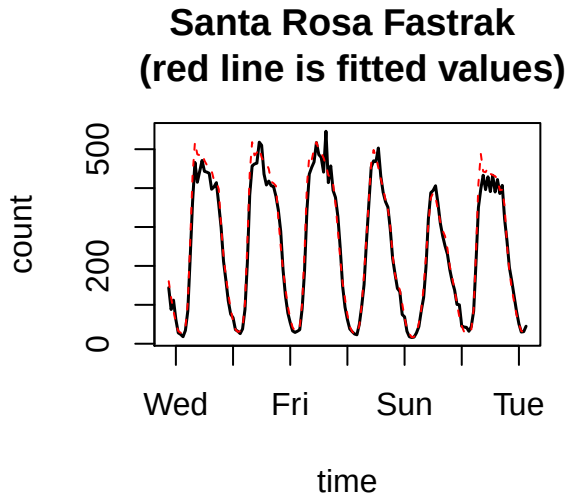
We can see that *hour* alone accounts for about 89% of the variation, while *month* and the long term trend of time account for only 0.14% and 1% of the variation, respectively. This tells us that traffic through Fastrak stations changes mostly based on the time of day, but remains relatively constant throughout a long period of time.

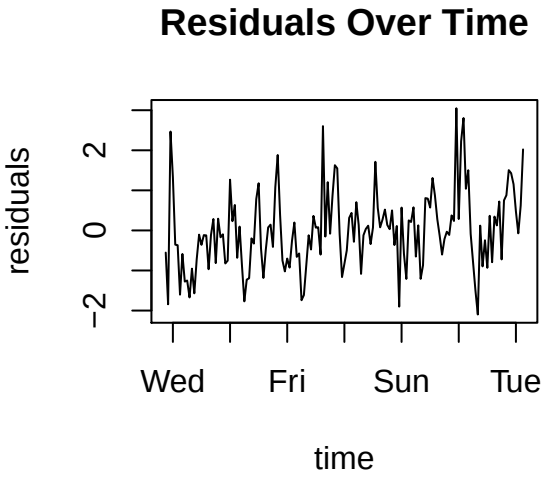
To deal with the issue of missing data and counts of zero we found as many consistently spaced residuals that were longer than 1000 on the filtered data. There were five groups.



We are going to focus on the third group for the rest of the analysis as it seems to be the most consistent. For ease of further analysis we center and scale the residuals. We also confirm our fit worked correctly by plotting one week's worth of data. The model seems to have fit the data quite well.

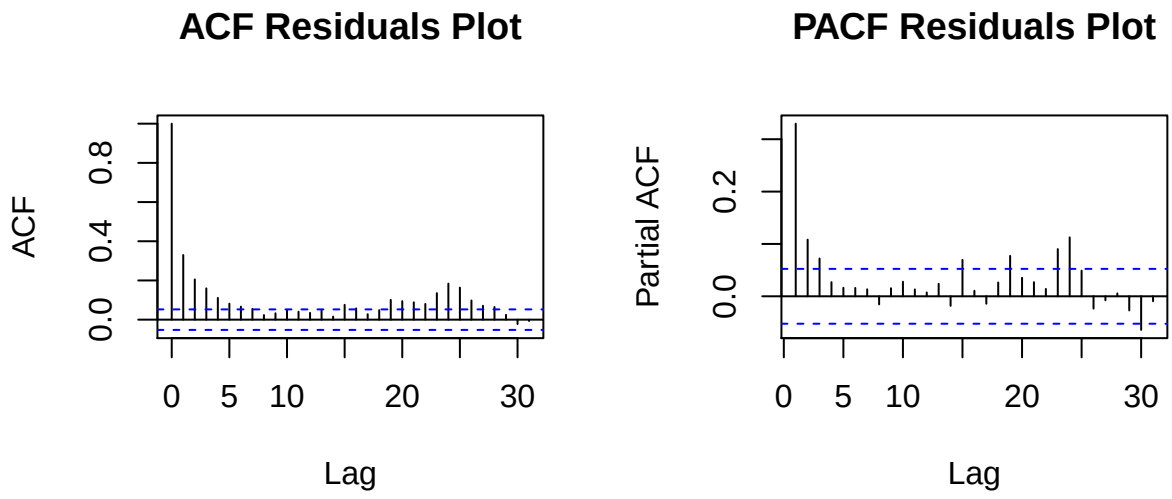




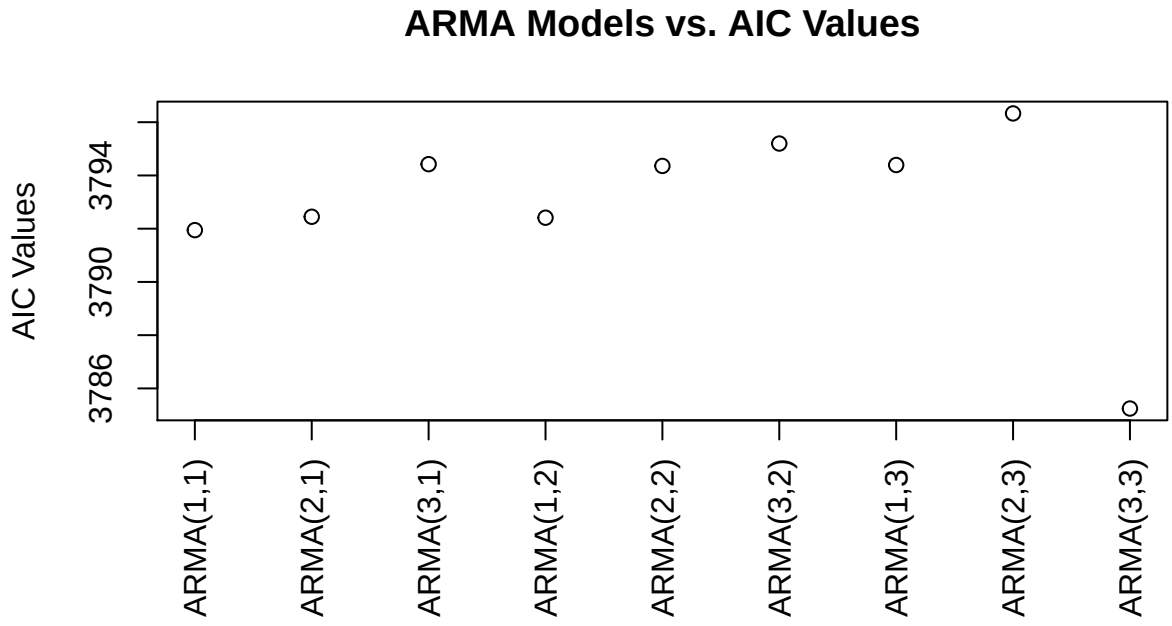


Time Series Analysis

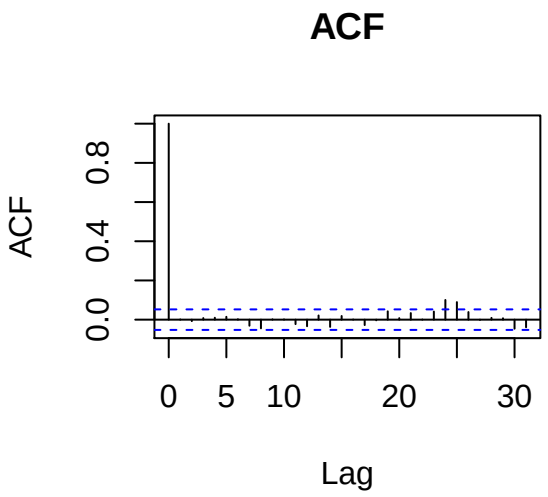
Now that all of the cleaning and processing is out of the way, we can finally begin a time series analysis on our residual series X. We will begin by looking at ACF and PACF plots.

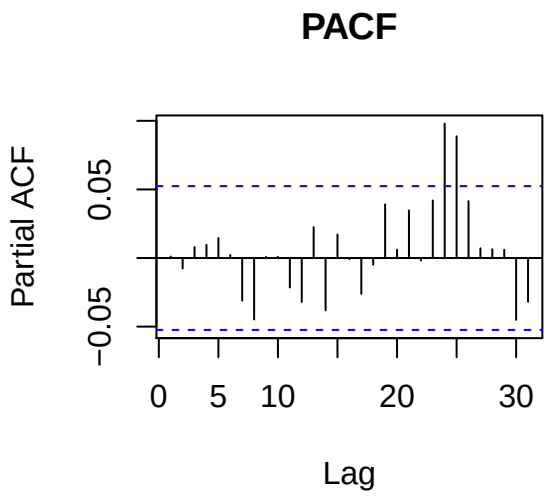


We see spikes at 24, which makes sense because our data is hourly counts. Thus a lag of 24 would represent the following day. To find the best ARMA model we searched over a grid and used AIC as a determining factor.

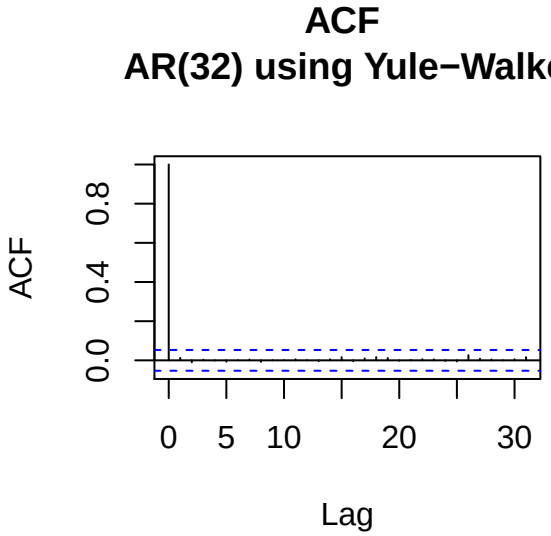


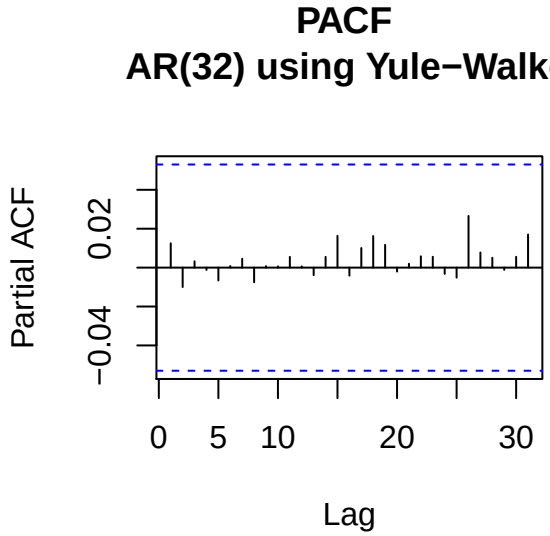
The AIC criteria chose an ARMA(6, 6) model. We will now check to see if the residuals resemble white noise.



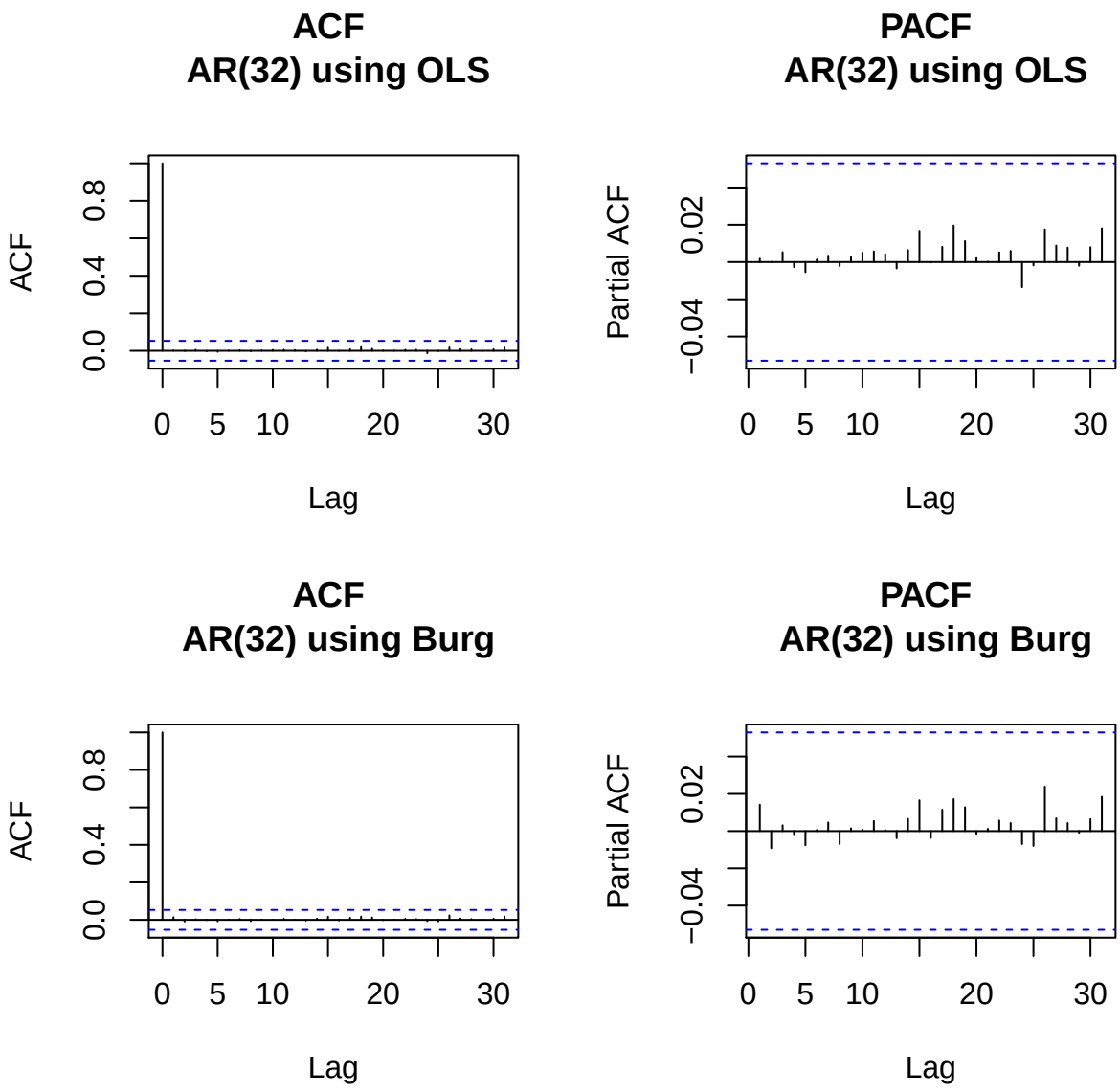


We still see spikes at a lag of 24. This definitely does not resemble white noise. We will now compare this with a larger AR model. The **ar()** function in **R** chose an AR(3) model based on AIC. The following graphs are from an AR(32) model with maximum order of 35.





These residuals resemble white noise much more so than the ARMA(6, 6) model. For curiosity and fun, there are two other methods to explore. The default method is Yule-Walker, but there are also the Maximum Liklihood Estimation (MLE) and Ordinary Least Sqaures (OLS) methods to use. We will look at the OLS method next, and then finally the MLE method.



Both the OLS and MLE methods also chose an AR(32) model. It is surprising that although each method chose an AR(32) model, all of the coefficients are different. For example, here are the first four for each method:

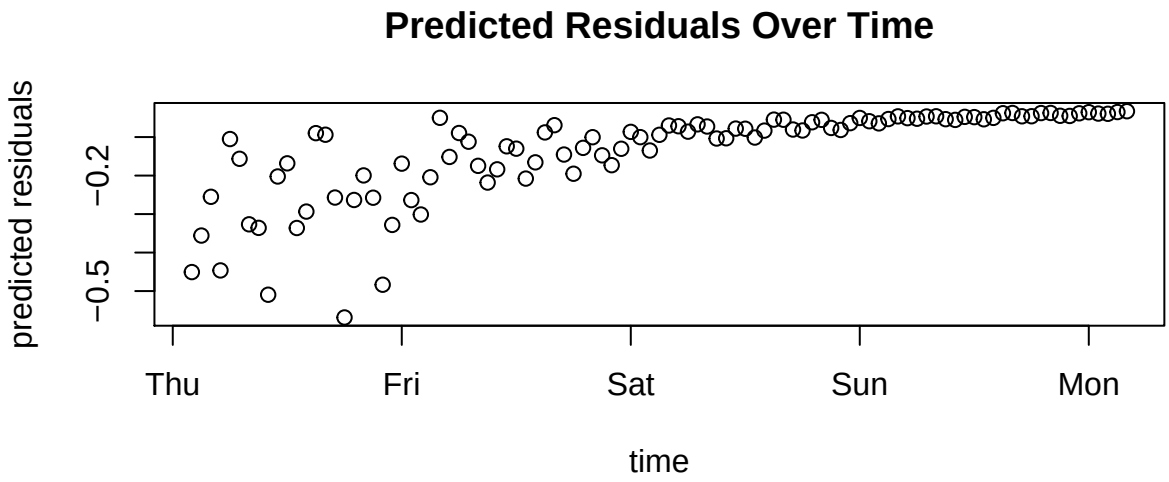
```
##   yule.walker         ols        burg
## 1 0.260738636 0.271636613 0.258953862
## 2 0.085682152 0.072707064 0.085358004
## 3 0.055885363 0.055592143 0.056362762
## 4 0.005726957 0.008108437 0.006372378
```

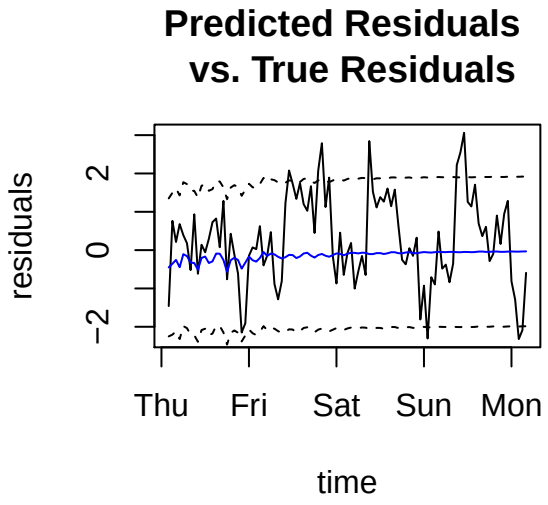
Prediction

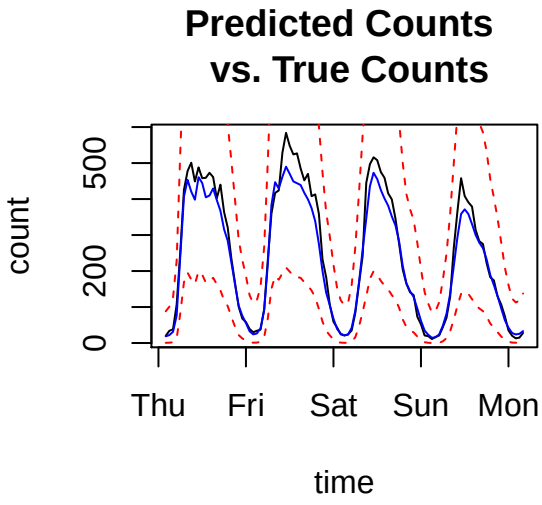
We will now do prediction. We will predict our residuals, and then put them back into the original scale by adding back on trend and seasonality and reversing the cube root transformation to determine our predictions for *count*. The adjusted R^2 for the ANOVA model is 0.972. We also observe that the residuals from this

model do not exhibit any trend or seasonality. We conclude that the ANOVA fit adequately explains the smooth component.

To evaluate the performance on the rough part we will use a fit on the first 1300 data points in the series to predict the remaining 99. We then look at the first 10 predictions compared to the actual values. Finally, we add back on trend and seasonality to see how our count prediction compared to the actual count values. The blue lines are predicted values, and the dashed lines are a 95% confidence interval. It seems like our predictions did quite well!







Spectral analysis

We will now do a spectral analysis on our residual series (X_t). We used the **spectrum()** function to look at the spectral density plots. The following are 3 **spectrum()** plots with different arguments.

The increase as the frequency tends to zero is evidence of long-term memory in the traffic counts. By visual inspection, there seems to be a maximum spectrum value around a frequency of 0.04. We found the exact value programatically (it was around 0.039) and determined 1 divided by that number was around 25.6. This

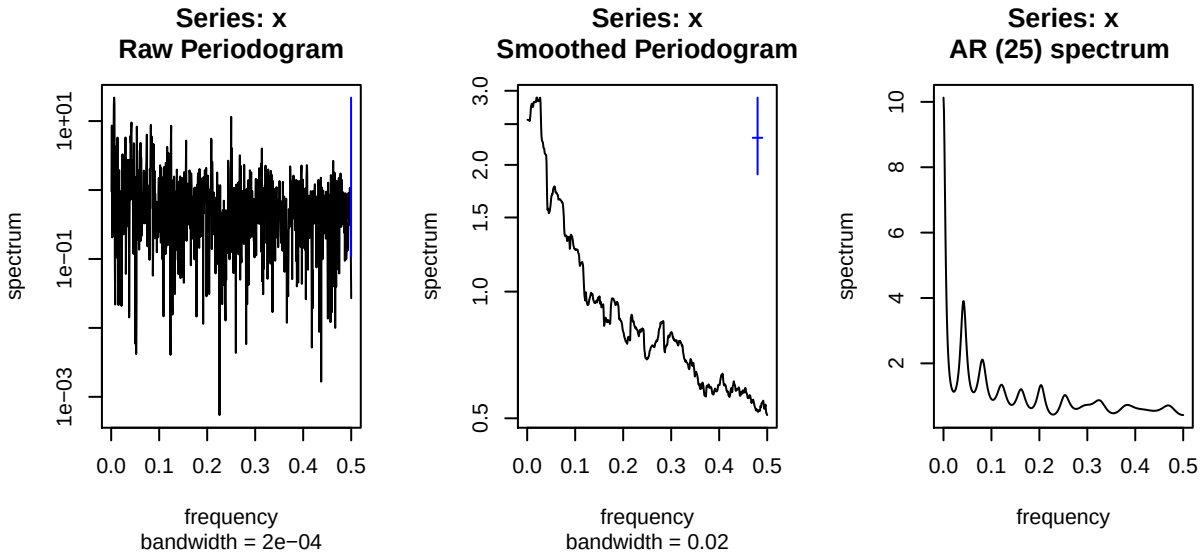


Figure 1: Left plot is default. Middle is span=100. Right is log='no' and method='ar'.

indicates the same information we learned from the ACF and PACF plots, that is, traffic through the Fastrak in Santa Rosa is mostly affected by the traffic conditions 1 day prior.

Discussion

At just about 8 million observations, the initial size of the data set presented some interesting computational challenges. We used the *R* package `biglm` along with chunking to fit a linear ANOVA model with 5000 rows, and it took 5 days to complete on the statistics department server (gumbel). Additionally, it used all 256GB of available memory. It is possible that different chunk sizes would have performed better. We also tried using the `lme4` package, which solves the problem with sparse matrices. This is ideal, but it failed to fit the full model, even with 256GB of memory on the department server.

After subsetting our data down to just the Fastrak station in Santa Rosa we had a much easier time fitting a model. We fit a generalized linear model, and then represented our model using the classical time series decomposition $Y_t = m_t + s_t + X_t$. This allowed us to just focus on the residuals of our linear model, which is the residual series (X_t). Before we could begin any time series analysis, though, we had to deal with one more issue: dirty data. We had missing values, too many zeros, and a month long strech where the counts were double what they should have been. In the end of our cleaning process we ended up with 5 consistent groups of about 1500 observations each. We ended up choosing to focus our analysison group 3, since it seemed to be the most consistent of the 5.

Once we had our clean data of about 1500 observations we could begin our time series analysis. We found through inspection of our ACF and PACF plots that we had spikes at a lag of 24. This was not surprising, as our data is hourly counts of traffic. Thus a lag of 24 would represent a day. Our spectral analysis produced similar results, although it gave us a value of 25, rather than 24. This value is close enough to 24 to conclude it represents 1 day.

Conclusion

Our first conclusion is that Fastrak data is very dirty! Much of the work that went into this paper involved solely cleaning and preparing the data. It reinforces the notion that a data analyst spends most of their time

on cleaning, rather than actual analysis. Our next conclusion is that, unsurprisingly, traffic flow through the Santa Rosa Fastrak is based primarily on the time of day. The time of day is the biggest factor when determining how much traffic there will be, whereas the given month has little to no affect.

References

TODO: . References: Name all resources you have used. Do not copy from the web and existing papers. You can and should use other resources, but they have to be clearly identified. Line-by-line copying from existing contributions will be considered plagiarism.

Appendix

```
# To produce the report
library(knitr)

# Box cox transformation
library(MASS)

# Semitransparent plotting with alpha
library(scales)

# Tested helper functions
source('../functions.R')

# Contains the `fastrak` data frame with 8 million rows
load('../fastrak.Rda')

sr1 = getstation(4300, fastrak)

kable(head(sr1))

with(sr1, plot(time, count, col=alpha('black', 0.2)))
fit0 = lm(count ~ poly(time, 3), sr1)
lines(sr1$time, predict(fit0), lwd=4, col='blue')

boxcox(lm(count ~ hour + weekday + hour:weekday + month + poly(time, 3), sr1))
f = count^(1/3) ~  hour + weekday + hour:weekday + month + poly(time, 3)

fit1 = lm(f, sr1)
res1 = residuals(fit1)
plot(sr1$time, res1, col=alpha('black', 0.1))

(cutoff = quantile(res1, 0.01))
sr2     = sr1[(res1 > cutoff) & (res1 < -cutoff), ]
dim(sr2)

fit2    = lm(f, sr2)
sr2$res = residuals(fit2)

with(sr2, plot(time, res, col=alpha('black', 0.2)))
```

```r
summary(fit2)$adj.r.squared

a2 = anova(fit2)
a2$percent = 100 * a2[, 'Sum Sq'] / sum(a2[, 'Sum Sq'])
kable(a2)


diffsr = diff(sr2$time)
islong = longrun(diffsr, 1, 1000)
long   = rle(islong)
sr3    = sr2[islong, ]

# Infer the time spaced groups
a         = rle(as.numeric(diff(sr3$time)))
a$values  = 1:((length(a$values) + 1) / 2)
a$lengths = 1 + a$lengths[a$lengths != 1]
sr3$group = inverse.rle(a)

# We can add the fitted values in for further analysis
# Exponentiating since we fitted the cube root
sr3$fitted = predict(fit2, sr3) ** 3

dim(sr3)
## [1] 6228   11
with(sr3, plot(time, res, col=alpha('black', 0.2)))
# save(sr3, file='cleaned.Rda')


# Time Series Analysis
acf(X)
pacf(X)

# Parameters to search over:
#ap = expand.grid(ar=3:6, ma=3:6)
ap = expand.grid(ar=1:3, ma=1:3)
getaic = function(ar, ma, x=X){
    arima(x, order=c(ar, 0, ma), optim.control=list(maxit=1000))$aic
}
aicvals = mapply(getaic, ap$ar, ap$ma)
plot(aicvals)

ap[which.min(aicvals), ]
arma1 = arima(X, order=c(6, 0, 6), optim.control=list(maxit=1000))
arma1$aic

acf(residuals(arma1))
pacf(residuals(arma1))

ar1 = ar(X, method='yw', order.max=35)
ar1$order

r1 = na.omit(ar1$resid)
```

```r
acf(r1)
pacf(r1)

# We don't need to fit an intercept because the data is centered
ar2 = ar(X, order.max=35, method='ols', intercept=FALSE)
ar2$order
## [1] 32
r2 = na.omit(ar2$resid)
acf(r2)
pacf(r2)

ar3 = ar(X, order.max=35, method='burg', intercept=FALSE)
ar3$order
## [1] 32
r3 = na.omit(ar3$resid)
acf(r3)
pacf(r3)

first4 = data.frame('yule-walker'=ar1$ar[1:4], 'ols'=ar2$ar[1:4],
           'burg'=ar3$ar[1:4])
first4


# Prediction
s = summary(fit2)
s$adj.r.squared
## [1] 0.9720136
kable(anova(fit2))

d = sr33$time[1301:1399]
ar1300 = ar(X[1:1300], order.max=35)

rough = predict(ar1300, n.ahead=99)
plot(d, rough$pred)

plot(d, X[1301:1399], type='l')

# Our predicitons
lines(d, rough$pred, col='blue')

# Add a line for 95% confidence
alpha = 0.05
zscale = qnorm(1 - alpha/2)

roughpreds = data.frame(time = d
                        , estimate = rough$pred
                        , upper = rough$pred + zscale * rough$se
                        , lower = rough$pred - zscale * rough$se
                        )

with(roughpreds, lines(time, upper, lty=2))
with(roughpreds, lines(time, lower, lty=2))
```

```
# Add back in the smooth component.
smoothpreds = roughpreds[, -1] + predict(fit2, sr33[1301:1399, ])

# Cube it to get back to the correct scale
smoothpreds = as.data.frame(smoothpreds ** 3)

# True counts
plot(d, sr33[1301:1399, 'count'], type='l')

# Our estimates
smoothpreds$time = d
with(smoothpreds, lines(time, estimate, col='blue'))
with(smoothpreds, lines(time, upper, lty=2, col='red'))
with(smoothpreds, lines(time, lower, lty=2, col='red'))


# Spectral Analysis
Xts = ts(X)
sp = spectrum(Xts, log='no', method='ar')
spectrum(Xts)
spectrum(Xts, span=100)

bigone = which.max(sp$spec[0.02 < sp$freq & sp$freq < 0.04])
freq = sp$freq[0.02 < sp$freq & sp$freq < 0.04][bigone]
1 / freq

#=============================================================

# Helper functions

runtests = TRUE

preclean = function(fastrak){
    # Perform all precleaning steps on the fastrak data

    # The zeros are most likely not valid readings.
    fastrak = fastrak[fastrak$count != 0, ]

    # The date range where the counts inexplicably doubled.
    a = as.POSIXct('2010-06-23')
    b = as.POSIXct('2010-08-04')
    toobig = with(fastrak, (a < time) & (time < b))

    fastrak = fastrak[!toobig, ]

    return(fastrak)
}

longrun = function(x, runtype, k){
    # Determine whether runtype in x occurs in a run of at least k.
    # Returns a logical vector which is TRUE if the corresponding
    # entry in x is in such a run.
    if(is.na(runtype)){
```

```
        isruntype = is.na(x)
    }
    else{
        isruntype = x == runtype
    }
    r1 = rle(isruntype)
    islong = (r1$lengths >= k)
    rep(islong, r1$lengths)
}

getstation = function(stationnumber, alldata){
    # Return a particular station from alldata sorted on time
    s = alldata[alldata$station == stationnumber, ]
    s[order(s$time), ]
}

makealldates = function(dframe, by='hour'){
    # Given a data frame that has a column `time`, this function
    # returns a data frame containing equi-spaced dates.
    # All columns other than date are filled in with NA.
}

addtimeparts = function(dframe){
    # Given a data frame that has a column `time`, this function
    # returns a data frame including factors for
    # year, month, dayofweek, hour
    out = dframe
    tm = out$time
    out$year = as.factor(format(tm, '%Y'))
    out$month = as.factor(months(tm))
    out$weekday = as.factor(weekdays(tm))
    out$hour = as.factor(format(tm, '%H'))
    out
}


#============================================================

# Test suite

if (runtests){
    library(RUnit)

    # longrun
    x1 = c(rep(0, 5), 1, rep(0, 4))
    checkEquals(longrun(x1, 0, 5), c(rep(TRUE, 5), rep(FALSE, 5)))
    xwithNA = c(rep(NA, 5), 1, rep(NA, 4))
    checkEquals(longrun(xwithNA, NA, 5), c(rep(TRUE, 5), rep(FALSE, 5)))

    # addtimeparts
    smalltime = data.frame(time=as.POSIXct('2000-01-01'))
    expected_addtime = data.frame(time=smalltime$time, year=as.factor(2000)
                                  , month=as.factor('January')
```

```
                                    , weekday=as.factor('Saturday')
                                    , hour=as.factor('00')
                                    )
    checkEquals(expected_addtime, addtimeparts(smalltime))

    # getstation
    alldata = data.frame(station = c(1, 1, 2), time = c(2, 1, 1))
    s1 = getstation(1, alldata)
    s1expected = data.frame(station = c(1, 1), time = c(1, 2))
    # MUST be integer row names
    row.names(s1expected) = 2:1
    checkEquals(s1, s1expected)
}
```