Analysis of Crime Database in San Francisco

Sonam Malhotra & Sooraj Shetty

3/27/2017

## Description

Our database is extracted from San Francisco Police department. The sudden growth in the population has brought an inequality in terms of living, housing shortages leading to no scarcity of crime in the city This dataset has the incidents reported from various counties of San Francisco. To protect the privacy of victims, other information like address is provided at the block level. The objective of the project is to predict the requirement of police officers on a specific time in a county where the percentage of crime is high in a day with the help of historical data. This data can also be used to define the zones in the county where it is safe to live or any additional need of police patrolling to prevent crimes.

## Preliminary Results

Before performing exploratory anlysis on the data, the data has to be cleaned and the relevant information has to be extracted for the analysis

library(RCurl)

## Warning: package 'RCurl' was built under R version 3.3.3

## Loading required package: bitops

## Warning: package 'bitops' was built under R version 3.3.2

library(plyr)

## Warning: package 'plyr' was built under R version 3.3.3

library(forecast)

## Warning: package 'forecast' was built under R version 3.3.3

library(ggplot2)

## Warning: package 'ggplot2' was built under R version 3.3.3

library(tseries)

## Warning: package 'tseries' was built under R version 3.3.3

source(url("http://lib.stat.cmu.edu/general/tsa2/Rcode/itall.R"))

## itall has been installed

mymodel <- read.csv("train.csv",header = TRUE)

In our dataset the Date column has the information pertaining to when the crime has occured. However, it is not present in a desirable format. Hence, we had to extract this information to perform further analysis.

mymodel$Dates <- as.POSIXct(mymodel$Dates, format = "%Y-%m-%d %H:%M:%S")  
mymodel$Date <- (format(mymodel$Dates, "%d"))  
mymodel$Year <- format(mymodel$Dates, "%Y")  
mymodel$months <- (format(mymodel$Dates, "%m"))  
mymodel$Hours <- (format(mymodel$Dates, "%H"))

Converting the data columns into factor will enable better analysis. Hence, we convert the data columns into factor

mymodel$Category <- as.factor(mymodel$Category)  
mymodel$DayOfWeek <- as.factor(mymodel$DayOfWeek)  
mymodel$PdDistrict <- as.factor(mymodel$PdDistrict)  
mymodel$Resolution <- as.factor(mymodel$Resolution)  
mymodel$Address <- as.factor(mymodel$Address)  
str(mymodel)

## 'data.frame': 878049 obs. of 13 variables:  
## $ Dates : POSIXct, format: "2015-05-13 23:53:00" "2015-05-13 23:53:00" ...  
## $ Category : Factor w/ 39 levels "ARSON","ASSAULT",..: 38 22 22 17 17 17 37 37 17 17 ...  
## $ Descript : Factor w/ 879 levels "ABANDONMENT OF CHILD",..: 867 811 811 405 405 407 740 740 405 405 ...  
## $ DayOfWeek : Factor w/ 7 levels "Friday","Monday",..: 7 7 7 7 7 7 7 7 7 7 ...  
## $ PdDistrict: Factor w/ 10 levels "BAYVIEW","CENTRAL",..: 5 5 5 5 6 3 3 1 7 2 ...  
## $ Resolution: Factor w/ 17 levels "ARREST, BOOKED",..: 1 1 1 12 12 12 12 12 12 12 ...  
## $ Address : Factor w/ 23228 levels "0 Block of HARRISON ST",..: 19791 19791 22698 4267 1844 1506 13323 18055 11385 17659 ...  
## $ X : num -122 -122 -122 -122 -122 ...  
## $ Y : num 37.8 37.8 37.8 37.8 37.8 ...  
## $ Date : chr "13" "13" "13" "13" ...  
## $ Year : chr "2015" "2015" "2015" "2015" ...  
## $ months : chr "05" "05" "05" "05" ...  
## $ Hours : chr "23" "23" "23" "23" ...

However, date is needed in numeric format and hence, we convert the columns to numeric for the year, date, months and hours

mymodel$Year <- as.numeric(mymodel$Year)  
mymodel$Date <- as.numeric(mymodel$Date)  
mymodel$months <- as.numeric(mymodel$months)  
mymodel$Hours <- as.numeric(mymodel$Hours)

Frequency of each element of the dataset provides an insight about the dataset. Higher the frequency higher the occurence. As an example, higher the frequency of any particular crime, higher is it's frequency. This allows us to recognize which areas are affected by which kind of crime. Hence, we calculate the frequency of the each category in the dataset.

mymodel <- na.omit(mymodel)  
category <- data.frame(table(mymodel$Category))  
head(category)

## Var1 Freq  
## 1 ARSON 1513  
## 2 ASSAULT 76871  
## 3 BAD CHECKS 406  
## 4 BRIBERY 289  
## 5 BURGLARY 36749  
## 6 DISORDERLY CONDUCT 4319

To modify the data for better analysis, we add a column as CategoryMap which will have factor values for category

mymodel$CategoryMap <- mymodel$Category  
  
levels(mymodel$CategoryMap) <- gsub("ARSON", 1, levels(mymodel$CategoryMap))  
levels(mymodel$CategoryMap) <- gsub("ASSAULT", 2, levels(mymodel$CategoryMap))  
levels(mymodel$CategoryMap) <- gsub("BAD CHECKS", 3, levels(mymodel$CategoryMap))  
levels(mymodel$CategoryMap) <- gsub("BRIBERY", 4, levels(mymodel$CategoryMap))  
levels(mymodel$CategoryMap) <- gsub("BURGLARY", 5, levels(mymodel$CategoryMap))  
levels(mymodel$CategoryMap) <- gsub("DISORDERLY CONDUCT", 6, levels(mymodel$CategoryMap))  
levels(mymodel$CategoryMap) <- gsub("DRIVING UNDER THE INFLUENCE", 7, levels(mymodel$CategoryMap))  
levels(mymodel$CategoryMap) <- gsub("DRUG/NARCOTIC", 8, levels(mymodel$CategoryMap))  
levels(mymodel$CategoryMap) <- gsub("DRUNKENNESS", 9, levels(mymodel$CategoryMap))  
levels(mymodel$CategoryMap) <- gsub("EMBEZZLEMENT", 10, levels(mymodel$CategoryMap))  
levels(mymodel$CategoryMap) <- gsub("EXTORTION", 11,levels(mymodel$CategoryMap))  
levels(mymodel$CategoryMap) <- gsub("FAMILY OFFENSES", 12,levels(mymodel$CategoryMap))  
levels(mymodel$CategoryMap) <- gsub("FRAUD", 13, levels(mymodel$CategoryMap))  
levels(mymodel$CategoryMap) <- gsub("FORGERY/COUNTERFEITING", 14,levels(mymodel$CategoryMap))  
levels(mymodel$CategoryMap) <- gsub("GAMBLING", 15, levels(mymodel$CategoryMap))  
levels(mymodel$CategoryMap) <- gsub("KIDNAPPING", 16, levels(mymodel$CategoryMap))  
levels(mymodel$CategoryMap) <- gsub("LARCENY/THEFT", 17, levels(mymodel$CategoryMap))  
levels(mymodel$CategoryMap) <- gsub("LIQUOR LAWS", 18,levels(mymodel$CategoryMap))  
levels(mymodel$CategoryMap) <- gsub("LOITERING", 19, levels(mymodel$CategoryMap))  
levels(mymodel$CategoryMap) <- gsub("MISSING PERSON", 20, levels(mymodel$CategoryMap))  
levels(mymodel$CategoryMap) <- gsub("NON-CRIMINAL", 21, levels(mymodel$CategoryMap))  
levels(mymodel$CategoryMap) <- gsub("OTHER OFFENSES", 22, levels(mymodel$CategoryMap))  
levels(mymodel$CategoryMap) <- gsub("PORNOGRAPHY/OBSCENE MAT", 23, levels(mymodel$CategoryMap))  
levels(mymodel$CategoryMap) <- gsub("PROSTITUTION", 24, levels(mymodel$CategoryMap))  
levels(mymodel$CategoryMap) <- gsub("RECOVERED VEHICLE", 25, levels(mymodel$CategoryMap))  
levels(mymodel$CategoryMap) <- gsub("ROBBERY", 26, levels(mymodel$CategoryMap))  
levels(mymodel$CategoryMap) <- gsub("RUNAWAY", 27, levels(mymodel$CategoryMap))  
levels(mymodel$CategoryMap) <- gsub("SECONDARY CODES", 28, levels(mymodel$CategoryMap))  
levels(mymodel$CategoryMap) <- gsub("SEX OFFENSES FORCIBLE", 29, levels(mymodel$CategoryMap))  
levels(mymodel$CategoryMap) <- gsub("SEX OFFENSES NON FORCIBLE", 30,levels(mymodel$CategoryMap))  
levels(mymodel$CategoryMap) <- gsub("STOLEN PROPERTY", 31,levels(mymodel$CategoryMap))  
levels(mymodel$CategoryMap) <- gsub("SUICIDE", 32, levels(mymodel$CategoryMap))  
levels(mymodel$CategoryMap) <- gsub("SUSPICIOUS OCC", 33, levels(mymodel$CategoryMap))  
levels(mymodel$CategoryMap) <- gsub("TREA", 34,levels(mymodel$CategoryMap))  
levels(mymodel$CategoryMap) <- gsub("TRESPASS", 35,levels(mymodel$CategoryMap))  
levels(mymodel$CategoryMap) <- gsub("VANDALISM", 36, levels(mymodel$CategoryMap))  
levels(mymodel$CategoryMap) <- gsub("VEHICLE THEFT", 37, levels(mymodel$CategoryMap))  
levels(mymodel$CategoryMap) <- gsub("WARRANTS", 38, levels(mymodel$CategoryMap))  
levels(mymodel$CategoryMap) <- gsub("WEAPON LAWS", 39, levels(mymodel$CategoryMap))

We further add a column as DayOfWeekMap which will have factor values for day of the week. This will help in classifying the days as wekday or weekend.

mymodel$DayOfWeekMap <- mymodel$DayOfWeek  
  
levels(mymodel$DayOfWeekMap) <- gsub("Sunday", 1, levels(mymodel$DayOfWeekMap))  
levels(mymodel$DayOfWeekMap) <- gsub("Saturday", 1, levels(mymodel$DayOfWeekMap))  
levels(mymodel$DayOfWeekMap) <- gsub("Monday", 0, levels(mymodel$DayOfWeekMap))  
levels(mymodel$DayOfWeekMap) <- gsub("Tuesday", 0, levels(mymodel$DayOfWeekMap))  
levels(mymodel$DayOfWeekMap) <- gsub("Wednesday", 0, levels(mymodel$DayOfWeekMap))  
levels(mymodel$DayOfWeekMap) <- gsub("Thursday", 0, levels(mymodel$DayOfWeekMap))  
levels(mymodel$DayOfWeekMap) <- gsub("Friday", 1, levels(mymodel$DayOfWeekMap))

Now having a preview of our processed data

head(mymodel)

## Dates Category Descript  
## 1 2015-05-13 23:53:00 WARRANTS WARRANT ARREST  
## 2 2015-05-13 23:53:00 OTHER OFFENSES TRAFFIC VIOLATION ARREST  
## 3 2015-05-13 23:33:00 OTHER OFFENSES TRAFFIC VIOLATION ARREST  
## 4 2015-05-13 23:30:00 LARCENY/THEFT GRAND THEFT FROM LOCKED AUTO  
## 5 2015-05-13 23:30:00 LARCENY/THEFT GRAND THEFT FROM LOCKED AUTO  
## 6 2015-05-13 23:30:00 LARCENY/THEFT GRAND THEFT FROM UNLOCKED AUTO  
## DayOfWeek PdDistrict Resolution Address X  
## 1 Wednesday NORTHERN ARREST, BOOKED OAK ST / LAGUNA ST -122.4259  
## 2 Wednesday NORTHERN ARREST, BOOKED OAK ST / LAGUNA ST -122.4259  
## 3 Wednesday NORTHERN ARREST, BOOKED VANNESS AV / GREENWICH ST -122.4244  
## 4 Wednesday NORTHERN NONE 1500 Block of LOMBARD ST -122.4270  
## 5 Wednesday PARK NONE 100 Block of BRODERICK ST -122.4387  
## 6 Wednesday INGLESIDE NONE 0 Block of TEDDY AV -122.4033  
## Y Date Year months Hours CategoryMap DayOfWeekMap  
## 1 37.77460 13 2015 5 23 38 0  
## 2 37.77460 13 2015 5 23 22 0  
## 3 37.80041 13 2015 5 23 22 0  
## 4 37.80087 13 2015 5 23 17 0  
## 5 37.77154 13 2015 5 23 17 0  
## 6 37.71343 13 2015 5 23 17 0

names(mymodel)

## [1] "Dates" "Category" "Descript" "DayOfWeek"   
## [5] "PdDistrict" "Resolution" "Address" "X"   
## [9] "Y" "Date" "Year" "months"   
## [13] "Hours" "CategoryMap" "DayOfWeekMap"

crime\_database<- mymodel

## Exploratory Data Analysis

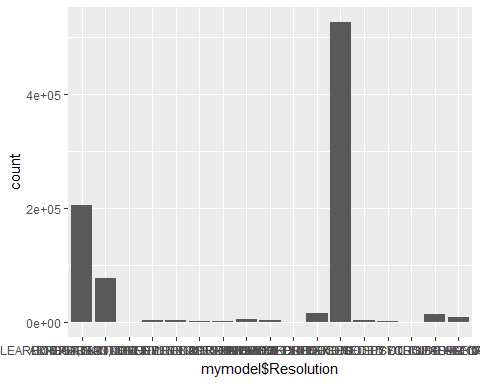
# Data distribution

How is the data distributed? This is a primary question that arises while performing EDA on the dataset. The graph show that the distribution of variable month and year is normal

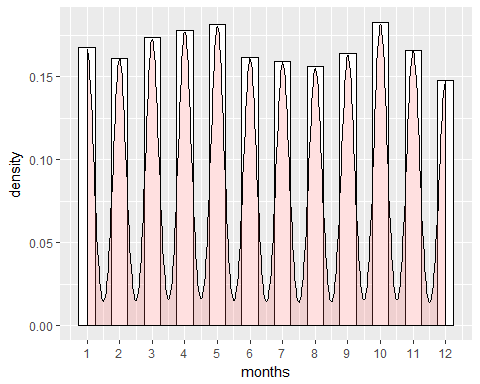
summary(mymodel$Category)

## ARSON ASSAULT   
## 1513 76871   
## BAD CHECKS BRIBERY   
## 406 289   
## BURGLARY DISORDERLY CONDUCT   
## 36749 4319   
## DRIVING UNDER THE INFLUENCE DRUG/NARCOTIC   
## 2267 53970   
## DRUNKENNESS EMBEZZLEMENT   
## 4279 1166   
## EXTORTION FAMILY OFFENSES   
## 256 491   
## FORGERY/COUNTERFEITING FRAUD   
## 10609 16679   
## GAMBLING KIDNAPPING   
## 146 2340   
## LARCENY/THEFT LIQUOR LAWS   
## 174894 1903   
## LOITERING MISSING PERSON   
## 1225 25986   
## NON-CRIMINAL OTHER OFFENSES   
## 92297 126179   
## PORNOGRAPHY/OBSCENE MAT PROSTITUTION   
## 22 7483   
## RECOVERED VEHICLE ROBBERY   
## 3138 22998   
## RUNAWAY SECONDARY CODES   
## 1946 9985   
## SEX OFFENSES FORCIBLE SEX OFFENSES NON FORCIBLE   
## 4388 148   
## STOLEN PROPERTY SUICIDE   
## 4539 508   
## SUSPICIOUS OCC TREA   
## 31413 6   
## TRESPASS VANDALISM   
## 7326 44721   
## VEHICLE THEFT WARRANTS   
## 53779 42213   
## WEAPON LAWS   
## 8555

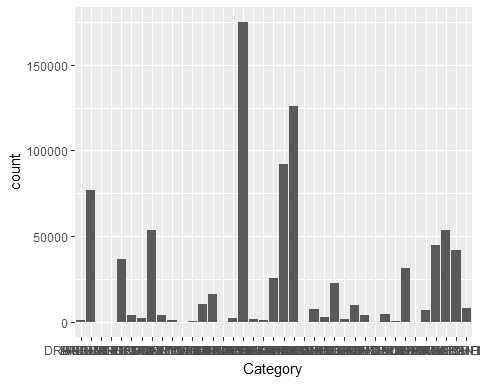
qplot(mymodel$Resolution,data = mymodel)



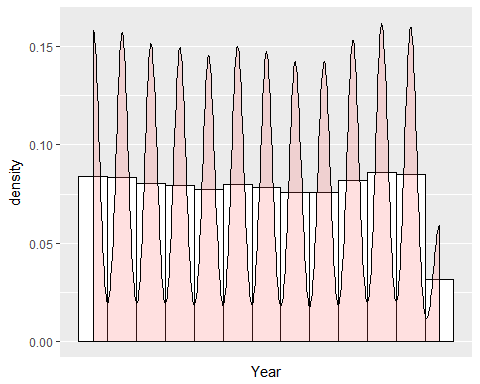
ggplot(mymodel,aes(x=months)) + scale\_x\_continuous(breaks=c(1:12)) +geom\_histogram(aes(y= ..density..),  
 binwidth=.5,  
 colour="black", fill="white") +  
 geom\_density(alpha=.2, fill="#FF6666")



qplot(Category,data = mymodel)

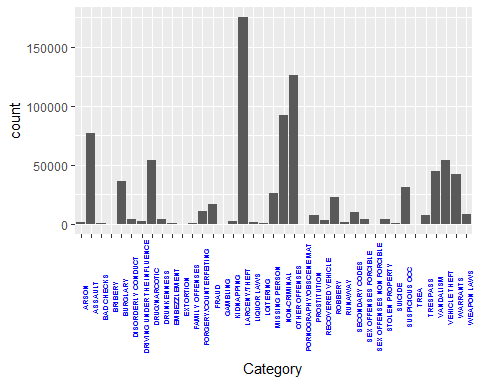


ggplot(mymodel,aes(x= Year)) + scale\_x\_continuous(breaks=c(1:39)) +  
 geom\_histogram(aes(y= ..density..), # Histogram with density instead of count on y-axis  
 binwidth=1,  
 colour="black",fill="white") +  
 geom\_density(alpha=.2, fill="#FF6666") # Overlay with transparent density plot



The graph shows the most number of categories with frequency counts.

qplot(Category,data=mymodel) +   
 theme(axis.text.x=element\_text(face="bold",color="blue",size=5,angle=90))

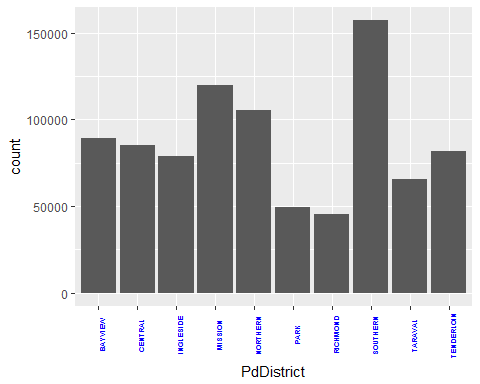


a <- ggplot(mymodel,aes(x=Category,fill=Category))  
a <- a + coord\_flip()  
a <- a + stat\_count(width=1,colour="white",geom="bar")  
a



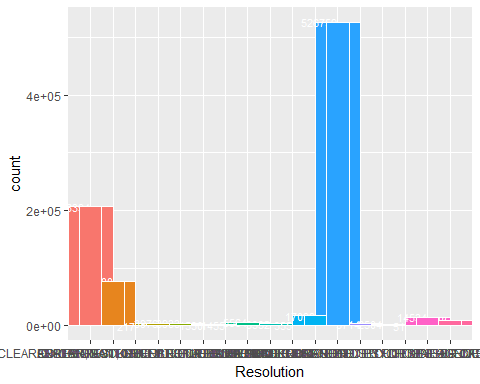
This graph shows that the distribuiton of crime is normally throughout the location. Significant areas can be observed with higher number of crimes.

qplot(PdDistrict,data=mymodel) +   
 theme(axis.text.x=element\_text(face="bold",color="blue",size=5,angle=90))



## This graph shows the resolution/outcome of crimes with number of counts  
b <- ggplot(mymodel,aes(x=Resolution,fill=Resolution))  
b <- b + geom\_bar(width=2,colour="white")   
b <- b + geom\_text(aes(y=..count..,label=..count..),  
 stat="count",color="white",  
 hjust=1.0,size=3)  
b <- b + theme(legend.position="none")  
b <- b + stat\_count(width=1,colour="white",geom="bar")  
b

## Warning: position\_stack requires non-overlapping x intervals

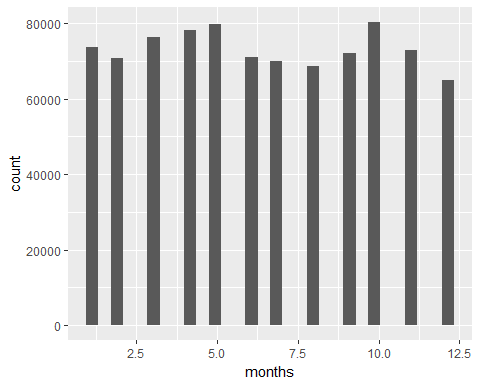


The above plot shows that the data is distributed normally with a peak around the year 1990-2000

We create the log transformed graph for this variable for further looking into the distribution. The log transformed graph suggests that the data is distributed normally.

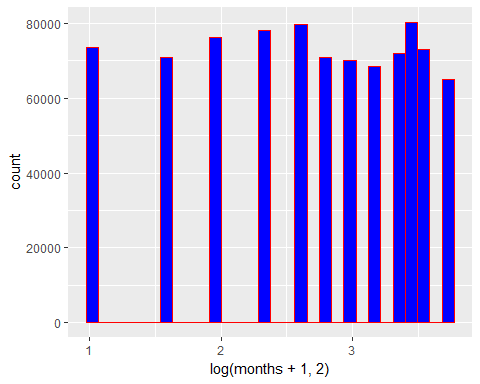
qplot(months,data=mymodel)

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



qplot(x=log(months+1,2),data=mymodel,color=I('red'),fill=I('blue'))

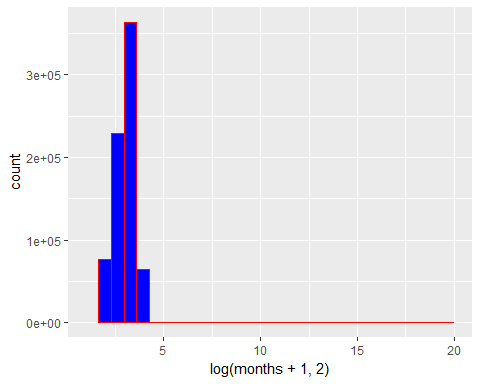
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



qplot(x=log(months+1,2),data=mymodel,color=I('red'),fill=I('blue'),xlim=c(1,20))

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

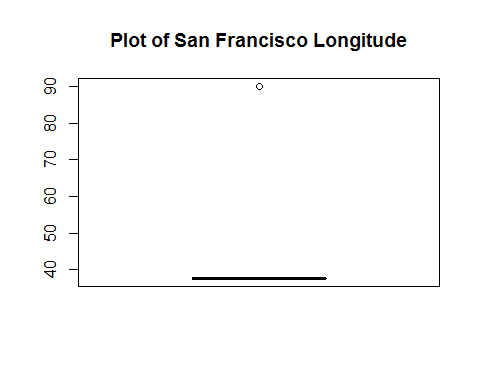
## Warning: Removed 1 rows containing missing values (geom\_bar).



# Checking the dataset for anomalies/outliers.

This allows us to identify the data which does not come together with majority of data. Of course, there are anomalies in the data, removing which will enable us to create the anomaly free data

boxplot(mymodel$Y, main = "Plot of San Francisco Longitude")



long <- which(mymodel$Y == 90, arr.ind = T)  
nrow(long)

## NULL

length(long)

## [1] 67

mymodel <- mymodel[-long,]  
nrow(mymodel)

## [1] 877935

train <- mymodel

## Unsupervised learning

library(arules)

## Warning: package 'arules' was built under R version 3.3.2

## Loading required package: Matrix

## Warning: package 'Matrix' was built under R version 3.3.2

##   
## Attaching package: 'arules'

## The following objects are masked from 'package:base':  
##   
## abbreviate, write

u1 <- mymodel[c("Year","months","Hours","CategoryMap","DayOfWeekMap")]   
  
u2 <- data.frame(as.factor(u1$Year),as.factor(u1$months),as.factor(u1$Hours),as.factor(u1$CategoryMap),as.factor(u1$DayOfWeekMap))  
  
colnames(u2) <- c("Year","months","Hours","CategoryMap","DayOfWeekMap")  
  
u3 <- as(u2,"transactions")

# look at the first five transactions  
  
inspect(u3[1:5])

## items transactionID  
## [1] {Year=2015,   
## months=5,   
## Hours=23,   
## CategoryMap=38,   
## DayOfWeekMap=0} 1  
## [2] {Year=2015,   
## months=5,   
## Hours=23,   
## CategoryMap=22,   
## DayOfWeekMap=0} 2  
## [3] {Year=2015,   
## months=5,   
## Hours=23,   
## CategoryMap=22,   
## DayOfWeekMap=0} 3  
## [4] {Year=2015,   
## months=5,   
## Hours=23,   
## CategoryMap=17,   
## DayOfWeekMap=0} 4  
## [5] {Year=2015,   
## months=5,   
## Hours=23,   
## CategoryMap=17,   
## DayOfWeekMap=0} 5

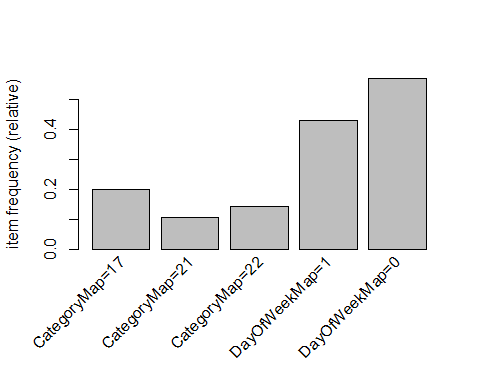
# Look at the frequency   
  
itemFrequency(u3[1:100, 1:13])

## Year=2003 Year=2004 Year=2005 Year=2006 Year=2007 Year=2008 Year=2009   
## 0 0 0 0 0 0 0   
## Year=2010 Year=2011 Year=2012 Year=2013 Year=2014 Year=2015   
## 0 0 0 0 0 1

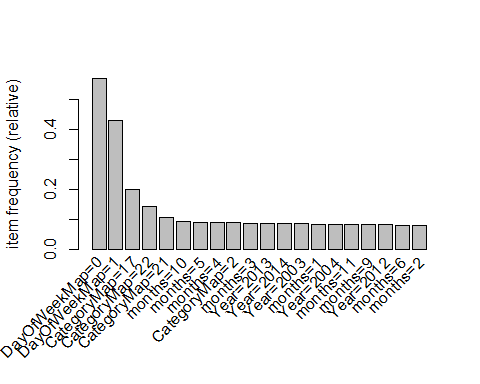
# plot the frequency  
head(u3)

## transactions in sparse format with  
## 6 transactions (rows) and  
## 90 items (columns)

itemFrequencyPlot(u3,support=0.1)



itemFrequencyPlot(u3,topN=20)



# Generate a set of 50 or so (non-redundant) rules.  
  
crime.rules<-apriori(u3)

## Apriori  
##   
## Parameter specification:  
## confidence minval smax arem aval originalSupport maxtime support minlen  
## 0.8 0.1 1 none FALSE TRUE 5 0.1 1  
## maxlen target ext  
## 10 rules FALSE  
##   
## Algorithmic control:  
## filter tree heap memopt load sort verbose  
## 0.1 TRUE TRUE FALSE TRUE 2 TRUE  
##   
## Absolute minimum support count: 87793   
##   
## set item appearances ...[0 item(s)] done [0.00s].  
## set transactions ...[90 item(s), 877935 transaction(s)] done [0.16s].  
## sorting and recoding items ... [5 item(s)] done [0.02s].  
## creating transaction tree ... done [0.23s].  
## checking subsets of size 1 2 done [0.00s].  
## writing ... [0 rule(s)] done [0.00s].  
## creating S4 object ... done [0.04s].

summary(crime.rules)

## set of 0 rules

Improving model performance.

crime.rules.2 <- apriori(u3, parameter = list(support = 0.01, confidence = 0.5, minlen = 1))

## Apriori  
##   
## Parameter specification:  
## confidence minval smax arem aval originalSupport maxtime support minlen  
## 0.5 0.1 1 none FALSE TRUE 5 0.01 1  
## maxlen target ext  
## 10 rules FALSE  
##   
## Algorithmic control:  
## filter tree heap memopt load sort verbose  
## 0.1 TRUE TRUE FALSE TRUE 2 TRUE  
##   
## Absolute minimum support count: 8779   
##   
## set item appearances ...[0 item(s)] done [0.00s].  
## set transactions ...[90 item(s), 877935 transaction(s)] done [0.15s].  
## sorting and recoding items ... [65 item(s)] done [0.02s].  
## creating transaction tree ... done [0.57s].  
## checking subsets of size 1 2 3 done [0.01s].  
## writing ... [63 rule(s)] done [0.00s].  
## creating S4 object ... done [0.06s].

crime.rules.2

## set of 63 rules

# converting the rule set to a data frame  
Rulesdataframe<- as(crime.rules.2, "data.frame")  
str(Rulesdataframe)

## 'data.frame': 63 obs. of 4 variables:  
## $ rules : Factor w/ 63 levels "{} => {DayOfWeekMap=0}",..: 1 2 32 27 8 5 16 63 9 33 ...  
## $ support : num 0.5704 0.0113 0.016 0.014 0.0147 ...  
## $ confidence: num 0.57 0.594 0.636 0.551 0.561 ...  
## $ lift : num 1 1.041 1.114 1.283 0.984 ...

Pruning redundant rules.

subset.matrix <- is.subset(crime.rules.2, crime.rules.2)  
subset.matrix[lower.tri(subset.matrix, diag=T)] <- NA  
redundant <- colSums(subset.matrix, na.rm=T) >= 1  
which(redundant)

## {CategoryMap=13,DayOfWeekMap=0}   
## 2   
## {Hours=7,DayOfWeekMap=0}   
## 3   
## {CategoryMap=26,DayOfWeekMap=0}   
## 5   
## {CategoryMap=20,DayOfWeekMap=0}   
## 6   
## {Year=2015,DayOfWeekMap=0}   
## 8   
## {CategoryMap=33,DayOfWeekMap=0}   
## 9   
## {Hours=8,DayOfWeekMap=0}   
## 10   
## {Hours=9,DayOfWeekMap=0}   
## 11   
## {CategoryMap=5,DayOfWeekMap=0}   
## 12   
## {Hours=10,DayOfWeekMap=0}   
## 13   
## {Hours=11,DayOfWeekMap=0}   
## 14   
## {CategoryMap=38,DayOfWeekMap=0}   
## 15   
## {Hours=23,DayOfWeekMap=0}   
## 16   
## {Hours=13,DayOfWeekMap=0}   
## 17   
## {Hours=21,DayOfWeekMap=0}   
## 18   
## {Hours=14,DayOfWeekMap=0}   
## 19   
## {Hours=20,DayOfWeekMap=0}   
## 20   
## {CategoryMap=36,DayOfWeekMap=0}   
## 21   
## {Hours=0,DayOfWeekMap=0}   
## 22   
## {Hours=22,DayOfWeekMap=0}   
## 23   
## {Hours=15,DayOfWeekMap=0}   
## 24   
## {Hours=19,DayOfWeekMap=0}   
## 25   
## {Hours=16,DayOfWeekMap=0}   
## 26   
## {Hours=12,DayOfWeekMap=0}   
## 27   
## {Hours=17,DayOfWeekMap=0}   
## 28   
## {CategoryMap=37,DayOfWeekMap=0}   
## 29   
## {CategoryMap=8,DayOfWeekMap=0}   
## 30   
## {Hours=18,DayOfWeekMap=0}   
## 31   
## {months=12,DayOfWeekMap=0}   
## 32   
## {Year=2010,DayOfWeekMap=0}   
## 33   
## {Year=2011,DayOfWeekMap=0}   
## 34   
## {Year=2007,DayOfWeekMap=0}   
## 35   
## {months=8,DayOfWeekMap=0}   
## 36   
## {Year=2009,DayOfWeekMap=0}   
## 37   
## {Year=2006,DayOfWeekMap=0}   
## 38   
## {months=7,DayOfWeekMap=0}   
## 39   
## {Year=2008,DayOfWeekMap=0}   
## 40   
## {Year=2005,DayOfWeekMap=0}   
## 41   
## {months=2,DayOfWeekMap=0}   
## 42   
## {months=6,DayOfWeekMap=0}   
## 43   
## {Year=2012,DayOfWeekMap=0}   
## 44   
## {months=9,DayOfWeekMap=0}   
## 45   
## {months=11,DayOfWeekMap=0}   
## 46   
## {Year=2004,DayOfWeekMap=0}   
## 47   
## {months=1,DayOfWeekMap=0}   
## 48   
## {Year=2003,DayOfWeekMap=0}   
## 49   
## {Year=2014,DayOfWeekMap=0}   
## 50   
## {Year=2013,DayOfWeekMap=0}   
## 51   
## {months=3,DayOfWeekMap=0}   
## 52   
## {CategoryMap=2,DayOfWeekMap=0}   
## 53   
## {months=4,DayOfWeekMap=0}   
## 54   
## {months=5,DayOfWeekMap=0}   
## 55   
## {months=10,DayOfWeekMap=0}   
## 56   
## {CategoryMap=21,DayOfWeekMap=0}   
## 57   
## {CategoryMap=22,DayOfWeekMap=0}   
## 58   
## {CategoryMap=17,DayOfWeekMap=0}   
## 59   
## {Year=2014,CategoryMap=17,DayOfWeekMap=0}   
## 60   
## {Year=2013,CategoryMap=17,DayOfWeekMap=0}   
## 61   
## {months=4,CategoryMap=17,DayOfWeekMap=0}   
## 62   
## {months=10,CategoryMap=17,DayOfWeekMap=0}   
## 63

# remove redundant rules  
rules.pruned <- crime.rules.2[!redundant]  
inspect(rules.pruned)

## lhs rhs support confidence lift   
## [1] {} => {DayOfWeekMap=0} 0.57037708 0.5703771 1.000000  
## [2] {Hours=2} => {DayOfWeekMap=1} 0.01396800 0.5511708 1.282918  
## [3] {Hours=1} => {DayOfWeekMap=1} 0.01633834 0.5481504 1.275887

Conviction:

conviction <- interestMeasure(rules.pruned, "conviction", transactions=u3)  
rules.conviction<-as(rules.pruned, "data.frame")  
rules.conviction<-data.frame(rules.conviction, conviction)  
rules.conviction

## rules support confidence lift conviction  
## 1 {} => {DayOfWeekMap=0} 0.57037708 0.5703771 1.000000 1.000000  
## 4 {Hours=2} => {DayOfWeekMap=1} 0.01396800 0.5511708 1.282918 1.270811  
## 7 {Hours=1} => {DayOfWeekMap=1} 0.01633834 0.5481504 1.275887 1.262316

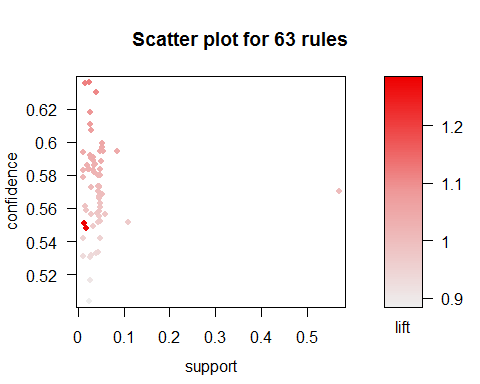
* Visualize your 50 association rules. Where do the best and worst end up in your plot?

library(arulesViz)

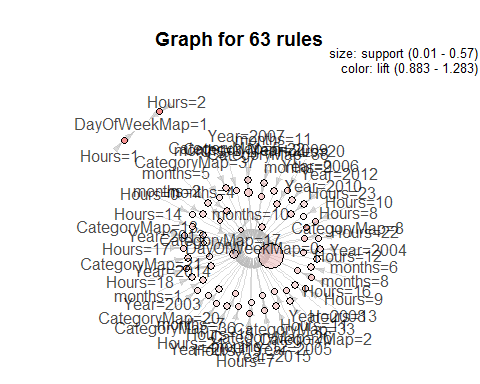
## Warning: package 'arulesViz' was built under R version 3.3.3

## Loading required package: grid

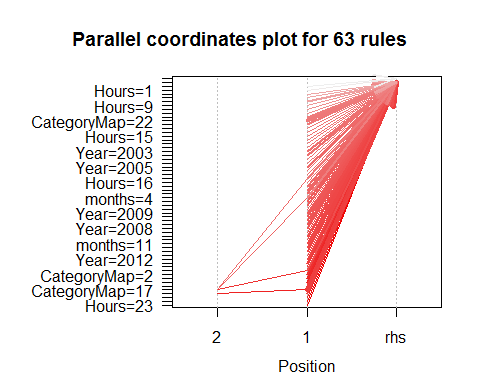
plot(crime.rules.2)



plot(crime.rules.2, method="graph", control=list(type="items"))



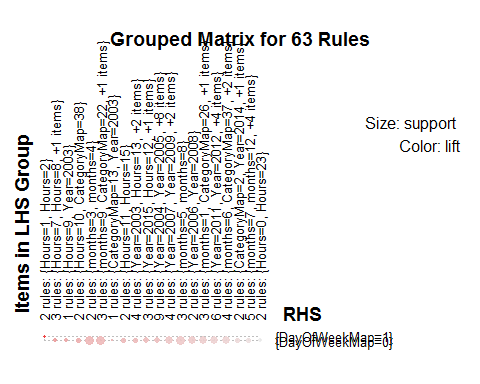
plot(crime.rules.2, method="paracoord", control=list(reorder=TRUE))



plot(crime.rules.2, method="grouped", control=list(reorder=TRUE))

## Warning: Unknown control parameters: reorder

## Available control parameters (with default values):  
## main = Grouped Matrix for 63 Rules  
## k = 20  
## rhs\_max = 10  
## lhs\_items = 2  
## aggr.fun = function (x, na.rm = FALSE) UseMethod("median")  
## col = c("#EE0000FF", "#EE0303FF", "#EE0606FF", "#EE0909FF", "#EE0C0CFF", "#EE0F0FFF", "#EE1212FF", "#EE1515FF", "#EE1818FF", "#EE1B1BFF", "#EE1E1EFF", "#EE2222FF", "#EE2525FF", "#EE2828FF", "#EE2B2BFF", "#EE2E2EFF", "#EE3131FF", "#EE3434FF", "#EE3737FF", "#EE3A3AFF", "#EE3D3DFF", "#EE4040FF", "#EE4444FF", "#EE4747FF", "#EE4A4AFF", "#EE4D4DFF", "#EE5050FF", "#EE5353FF", "#EE5656FF", "#EE5959FF", "#EE5C5CFF", "#EE5F5FFF", "#EE6262FF", "#EE6666FF", "#EE6969FF", "#EE6C6CFF", "#EE6F6FFF", "#EE7272FF", "#EE7575FF", "#EE7878FF", "#EE7B7BFF", "#EE7E7EFF", "#EE8181FF", "#EE8484FF", "#EE8888FF", "#EE8B8BFF", "#EE8E8EFF", "#EE9191FF", "#EE9494FF", "#EE9797FF", "#EE9999FF", "#EE9B9BFF", "#EE9D9DFF", "#EE9F9FFF", "#EEA0A0FF", "#EEA2A2FF", "#EEA4A4FF", "#EEA5A5FF", "#EEA7A7FF", "#EEA9A9FF", "#EEABABFF", "#EEACACFF", "#EEAEAEFF", "#EEB0B0FF", "#EEB1B1FF", "#EEB3B3FF", "#EEB5B5FF", "#EEB7B7FF", "#EEB8B8FF", "#EEBABAFF", "#EEBCBCFF", "#EEBDBDFF", "#EEBFBFFF", "#EEC1C1FF", "#EEC3C3FF", "#EEC4C4FF", "#EEC6C6FF", "#EEC8C8FF", "#EEC9C9FF", "#EECBCBFF", "#EECDCDFF", "#EECFCFFF", "#EED0D0FF", "#EED2D2FF", "#EED4D4FF", "#EED5D5FF", "#EED7D7FF", "#EED9D9FF", "#EEDBDBFF", "#EEDCDCFF", "#EEDEDEFF", "#EEE0E0FF", "#EEE1E1FF", "#EEE3E3FF", "#EEE5E5FF", "#EEE7E7FF", "#EEE8E8FF", "#EEEAEAFF", "#EEECECFF", "#EEEEEEFF")  
## reverse = TRUE  
## xlab = NULL  
## ylab = NULL  
## legend = Size: support Color: lift  
## spacing = -1  
## panel.function = function (row, size, shading, spacing) { size[size == 0] <- NA shading[is.na(shading)] <- 1 grid.circle(x = c(1:length(size)), y = row, r = size/2 \* (1 - spacing), default.units = "native", gp = gpar(fill = shading, col = shading, alpha = 0.9)) }  
## gp\_main = list(cex = 1.2, fontface = "bold", font = 2)  
## gp\_labels = list(cex = 0.8)  
## gp\_labs = list(cex = 1.2, fontface = "bold", font = 2)  
## gp\_lines = list(col = "gray", lty = 3)  
## newpage = TRUE  
## interactive = FALSE  
## max.shading = NA  
## verbose = FALSE



## Supervised learning

Rearranging the data

u1 <- mymodel[c("Hours","CategoryMap")]   
u2 <- data.frame(as.numeric(u1$Hours),as.numeric(u1$CategoryMap))  
colnames(u2) <- c("Hours","CategoryMap")  
table(u2$CategoryMap)

##   
## 1 2 3 4 5 6 7 8 9 10   
## 1513 76867 406 289 36748 4317 2267 53970 4279 1166   
## 11 12 13 14 15 16 17 18 19 20   
## 256 491 10609 16679 146 2340 174879 1903 1225 25986   
## 21 22 23 24 25 26 27 28 29 30   
## 92293 126162 22 7483 3138 22997 1946 9985 4387 148   
## 31 32 33 34 35 36 37 38 39   
## 4538 508 31411 6 7325 44720 53770 42205 8555

class <- u2[,1]  
leaf <- u2[1:90,1:2]  
shuff<-runif(nrow(leaf))  
leaf<-leaf[order(shuff),]  
class <- class[order(shuff)]

Scaling the data

leaf.scaled<-as.data.frame(lapply(leaf, scale))  
head(leaf.scaled)

## Hours CategoryMap  
## 1 -0.6333208 -0.58833178  
## 2 0.6475527 -0.02623135  
## 3 -0.6333208 1.21038958  
## 4 -1.2737576 -0.13865144  
## 5 0.6475527 -0.58833178  
## 6 -0.6333208 1.66006992

summary(leaf.scaled)

## Hours CategoryMap   
## Min. :-1.273758 Min. :-2.2746   
## 1st Qu.:-0.633321 1st Qu.:-0.5883   
## Median : 0.007116 Median :-0.1387   
## Mean : 0.000000 Mean : 0.0000   
## 3rd Qu.: 0.647553 3rd Qu.: 0.3110   
## Max. : 1.928426 Max. : 1.8849

Normalizing the data

normalize<- function(x) {  
 return((x-min(x))/(max(x)-min(x)))  
}  
leaf.normalized<-as.data.frame(lapply(leaf,normalize))  
leaf.scaled.normalized<-as.data.frame(lapply(leaf.scaled, normalize))  
head(leaf.normalized)

## Hours CategoryMap  
## 1 0.2 0.4054054  
## 2 0.6 0.5405405  
## 3 0.2 0.8378378  
## 4 0.0 0.5135135  
## 5 0.6 0.4054054  
## 6 0.2 0.9459459

#Different sized datasets  
train <- 1:75  
train\_small <- 1:50  
test <- 76:90  
test\_large <- 51:90  
class <- as.factor(class)

## Decision Tree

library(C50)

## Warning: package 'C50' was built under R version 3.3.3

model <- C5.0(leaf.scaled.normalized[train,], class[train])  
model\_small <- C5.0(leaf.scaled.normalized[train\_small,], class[train\_small])  
model\_no\_scale <- C5.0(leaf.normalized[train,], class[train])  
model\_no\_norm <- C5.0(leaf.scaled[train,], class[train])  
model\_no\_ns <- C5.0(leaf[train,], class[train])  
  
summary(model)

##   
## Call:  
## C5.0.default(x = leaf.scaled.normalized[train, ], y = class[train])  
##   
##   
## C5.0 [Release 2.07 GPL Edition] Fri Apr 28 19:23:00 2017  
## -------------------------------  
##   
## Class specified by attribute `outcome'  
##   
## Read 75 cases (3 attributes) from undefined.data  
##   
## Decision tree:  
##   
## Hours <= 0.2:  
## :...Hours <= 0: 18 (12)  
## : Hours > 0: 19 (22)  
## Hours > 0.2:  
## :...Hours <= 0.4: 20 (18)  
## Hours > 0.4:  
## :...Hours <= 0.6: 21 (10)  
## Hours > 0.6:  
## :...Hours <= 0.8: 22 (5)  
## Hours > 0.8: 23 (8)  
##   
##   
## Evaluation on training data (75 cases):  
##   
## Decision Tree   
## ----------------   
## Size Errors   
##   
## 6 0( 0.0%) <<  
##   
##   
## (a) (b) (c) (d) (e) (f) <-classified as  
## ---- ---- ---- ---- ---- ----  
## 12 (a): class 18  
## 22 (b): class 19  
## 18 (c): class 20  
## 10 (d): class 21  
## 5 (e): class 22  
## 8 (f): class 23  
##   
##   
## Attribute usage:  
##   
## 100.00% Hours  
##   
##   
## Time: 0.0 secs

summary(model\_small)

##   
## Call:  
## C5.0.default(x = leaf.scaled.normalized[train\_small, ], y  
## = class[train\_small])  
##   
##   
## C5.0 [Release 2.07 GPL Edition] Fri Apr 28 19:23:00 2017  
## -------------------------------  
##   
## Class specified by attribute `outcome'  
##   
## Read 50 cases (3 attributes) from undefined.data  
##   
## Decision tree:  
##   
## Hours <= 0.2:  
## :...Hours <= 0: 18 (8)  
## : Hours > 0: 19 (14)  
## Hours > 0.2:  
## :...Hours <= 0.4: 20 (10)  
## Hours > 0.4:  
## :...Hours <= 0.6: 21 (9)  
## Hours > 0.6:  
## :...Hours <= 0.8: 22 (3)  
## Hours > 0.8: 23 (6)  
##   
##   
## Evaluation on training data (50 cases):  
##   
## Decision Tree   
## ----------------   
## Size Errors   
##   
## 6 0( 0.0%) <<  
##   
##   
## (a) (b) (c) (d) (e) (f) <-classified as  
## ---- ---- ---- ---- ---- ----  
## 8 (a): class 18  
## 14 (b): class 19  
## 10 (c): class 20  
## 9 (d): class 21  
## 3 (e): class 22  
## 6 (f): class 23  
##   
##   
## Attribute usage:  
##   
## 100.00% Hours  
##   
##   
## Time: 0.0 secs

summary(model\_no\_scale)

##   
## Call:  
## C5.0.default(x = leaf.normalized[train, ], y = class[train])  
##   
##   
## C5.0 [Release 2.07 GPL Edition] Fri Apr 28 19:23:00 2017  
## -------------------------------  
##   
## Class specified by attribute `outcome'  
##   
## Read 75 cases (3 attributes) from undefined.data  
##   
## Decision tree:  
##   
## Hours <= 0.2:  
## :...Hours <= 0: 18 (12)  
## : Hours > 0: 19 (22)  
## Hours > 0.2:  
## :...Hours <= 0.4: 20 (18)  
## Hours > 0.4:  
## :...Hours <= 0.6: 21 (10)  
## Hours > 0.6:  
## :...Hours <= 0.8: 22 (5)  
## Hours > 0.8: 23 (8)  
##   
##   
## Evaluation on training data (75 cases):  
##   
## Decision Tree   
## ----------------   
## Size Errors   
##   
## 6 0( 0.0%) <<  
##   
##   
## (a) (b) (c) (d) (e) (f) <-classified as  
## ---- ---- ---- ---- ---- ----  
## 12 (a): class 18  
## 22 (b): class 19  
## 18 (c): class 20  
## 10 (d): class 21  
## 5 (e): class 22  
## 8 (f): class 23  
##   
##   
## Attribute usage:  
##   
## 100.00% Hours  
##   
##   
## Time: 0.0 secs

summary(model\_no\_norm)

##   
## Call:  
## C5.0.default(x = leaf.scaled[train, ], y = class[train])  
##   
##   
## C5.0 [Release 2.07 GPL Edition] Fri Apr 28 19:23:00 2017  
## -------------------------------  
##   
## Class specified by attribute `outcome'  
##   
## Read 75 cases (3 attributes) from undefined.data  
##   
## Decision tree:  
##   
## Hours <= -0.6333208:  
## :...Hours <= -1.273758: 18 (12)  
## : Hours > -1.273758: 19 (22)  
## Hours > -0.6333208:  
## :...Hours <= 0.007115964: 20 (18)  
## Hours > 0.007115964:  
## :...Hours <= 0.6475527: 21 (10)  
## Hours > 0.6475527:  
## :...Hours <= 1.287989: 22 (5)  
## Hours > 1.287989: 23 (8)  
##   
##   
## Evaluation on training data (75 cases):  
##   
## Decision Tree   
## ----------------   
## Size Errors   
##   
## 6 0( 0.0%) <<  
##   
##   
## (a) (b) (c) (d) (e) (f) <-classified as  
## ---- ---- ---- ---- ---- ----  
## 12 (a): class 18  
## 22 (b): class 19  
## 18 (c): class 20  
## 10 (d): class 21  
## 5 (e): class 22  
## 8 (f): class 23  
##   
##   
## Attribute usage:  
##   
## 100.00% Hours  
##   
##   
## Time: 0.0 secs

summary(model\_no\_ns)

##   
## Call:  
## C5.0.default(x = leaf[train, ], y = class[train])  
##   
##   
## C5.0 [Release 2.07 GPL Edition] Fri Apr 28 19:23:00 2017  
## -------------------------------  
##   
## Class specified by attribute `outcome'  
##   
## Read 75 cases (3 attributes) from undefined.data  
##   
## Decision tree:  
##   
## Hours <= 19:  
## :...Hours <= 18: 18 (12)  
## : Hours > 18: 19 (22)  
## Hours > 19:  
## :...Hours <= 20: 20 (18)  
## Hours > 20:  
## :...Hours <= 21: 21 (10)  
## Hours > 21:  
## :...Hours <= 22: 22 (5)  
## Hours > 22: 23 (8)  
##   
##   
## Evaluation on training data (75 cases):  
##   
## Decision Tree   
## ----------------   
## Size Errors   
##   
## 6 0( 0.0%) <<  
##   
##   
## (a) (b) (c) (d) (e) (f) <-classified as  
## ---- ---- ---- ---- ---- ----  
## 12 (a): class 18  
## 22 (b): class 19  
## 18 (c): class 20  
## 10 (d): class 21  
## 5 (e): class 22  
## 8 (f): class 23  
##   
##   
## Attribute usage:  
##   
## 100.00% Hours  
##   
##   
## Time: 0.0 secs

## Time Series Analysis

# Autoregressive integrated moving average (ARIMA)

An autoregressive integrated moving average (ARIMA or ARMA) model combines an autoregressive component with a moving average component in to a single model.

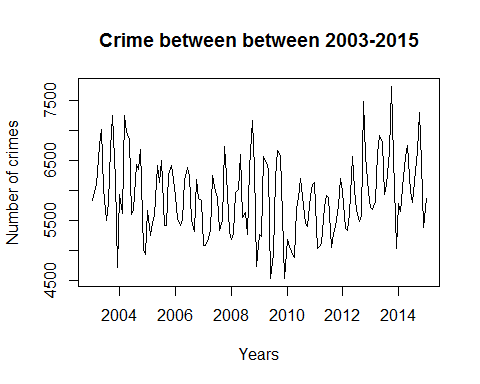
# Time series analysis & Predictiona Model in R

The ts() function will convert a numeric vector into an R time series object. The format is ts(vector, start=, end=, frequency=) where start and end are the times of the first and last observation and frequency is the number of observations per unit time (1=annual, 4=quartly, 12=monthly, etc.).

crimes<- count(mymodel,c("Year", "months"))  
head(crimes)

## Year months freq  
## 1 2003 1 5831  
## 2 2003 2 5963  
## 3 2003 3 6099  
## 4 2003 4 6750  
## 5 2003 5 7024  
## 6 2003 6 6045

# set the freq parameter to 12 to indicate monthly readings  
  
crime\_timeseries <- ts(crimes$freq, start = c(2003, 1), end = c(2015, 1), frequency = 12)   
  
plot(crime\_timeseries, xlab='Years' ,ylab='Number of crimes', main='Crime between between 2003-2015')



## ------------- USE ARIMA MODEL ---------------------  
#creating ranges of possible values for the order parameters p, d, and q.  
  
  
d <- 0 : 2  
p <- 0 : 6  
q <- 0 : 6  
  
crime\_models <- expand.grid(d = d, p = p, q = q)  
  
  
head(crime\_models, n = 4)

## d p q  
## 1 0 0 0  
## 2 1 0 0  
## 3 2 0 0  
## 4 0 1 0

getTSModelAIC <- function(ts\_mymodel, p, d, q) {  
 ts\_model <- arima(ts\_mymodel, order = c(p, d, q))  
 return(ts\_model$aic)  
 }  
  
  
getTSModelAICSafe <- function(ts\_mymodel, p, d, q) {   
 result = tryCatch({  
 getTSModelAIC(ts\_mymodel, p, d, q)  
 }, error = function(e) {  
 Inf  
 })  
 }  
   
 # PICK THE BEST MODEL THAT HAS THE SMALLEST AIC   
  
crime\_models$aic <- mapply(function(x, y, z)   
 getTSModelAICSafe(crime\_timeseries, x, y, z), crime\_models$p,   
 crime\_models$d, crime\_models$q)

## Warning in arima(ts\_mymodel, order = c(p, d, q)): possible convergence  
## problem: optim gave code = 1

## Warning in log(s2): NaNs produced

## Warning in arima(ts\_mymodel, order = c(p, d, q)): possible convergence  
## problem: optim gave code = 1  
  
## Warning in arima(ts\_mymodel, order = c(p, d, q)): possible convergence  
## problem: optim gave code = 1

## Warning in log(s2): NaNs produced  
  
## Warning in log(s2): NaNs produced

## Warning in arima(ts\_mymodel, order = c(p, d, q)): possible convergence  
## problem: optim gave code = 1  
  
## Warning in arima(ts\_mymodel, order = c(p, d, q)): possible convergence  
## problem: optim gave code = 1

subset(crime\_models,aic == min(aic))

## d p q aic  
## 141 2 4 6 2191.955

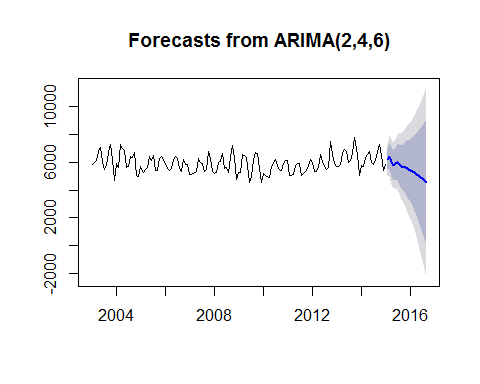
# ARIMA model for best p,d,q order model   
  
crime\_model <- arima(crime\_timeseries, order = c(2, 4, 6))  
summary(crime\_model)

##   
## Call:  
## arima(x = crime\_timeseries, order = c(2, 4, 6))  
##   
## Coefficients:

## Warning in sqrt(diag(x$var.coef)): NaNs produced

## ar1 ar2 ma1 ma2 ma3 ma4 ma5 ma6  
## 0.1575 -0.6102 -3.3149 4.4100 -3.7906 3.1633 -2.0128 0.5453  
## s.e. 0.0695 0.0711 0.0073 0.0449 0.0904 0.0773 0.0216 NaN  
##   
## sigma^2 estimated as 319944: log likelihood = -1110.38, aic = 2238.75  
##   
## Training set error measures:  
## ME RMSE MAE MPE MAPE MASE  
## Training set -29.80263 557.78 433.2029 -1.038267 7.440622 0.847122  
## ACF1  
## Training set -0.1080933

#------------------- Prediction ---------------------------------  
  
plot(forecast(crime\_model, 20))



# Seasonal Models

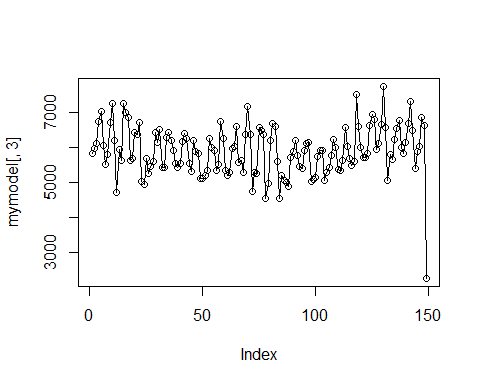
When there are patterns that repeat over known, fixed periods of time (i.e. day, week, month, quarter, year, etc.) within the data set it is considered to be seasonal variation. One has a model for the periodic fluctuations based on knowledge of the domain.

The seasonal ARIMA model incorporates both non-seasonal and seasonal factors in a multiplicative model.

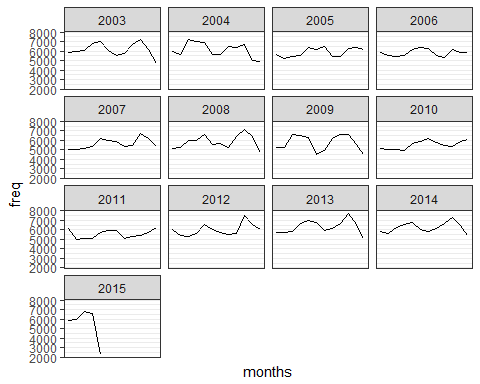
# Seasonal Models in R

Here we are using an ARIMA model to identify seasonality trends by looking for signficant seasonal differences.

mymodel <- count(mymodel, c("Year","months"))  
plot(mymodel[,3], type='o') #Aggregate the data by month



ggplot(mymodel, aes(x=months, y= freq))+  
stat\_summary(geom = 'line', fun.y='mean')+ # take the mean of eachmonth  
scale\_x\_discrete(breaks=seq(1,12,1), labels=seq(1,12,1))+  
theme\_bw()+   
facet\_wrap(~Year) # year by year



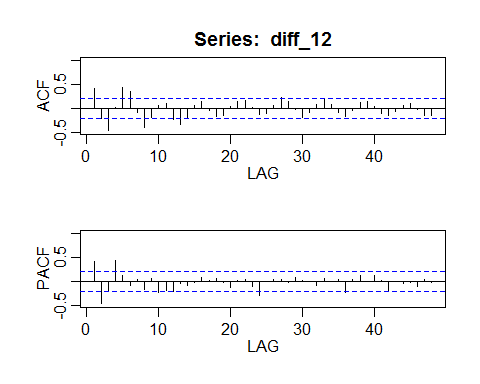
#Looking at the ACF and PACF.  
  
mydata<-ts(mymodel[1:100,][,3])  
mydata

## Time Series:  
## Start = 1   
## End = 100   
## Frequency = 1   
## [1] 5831 5963 6099 6750 7024 6045 5502 5798 6703 7259 6194 4713 5937 5626  
## [15] 7259 6985 6862 5611 5679 6435 6361 6695 5011 4944 5668 5251 5448 5585  
## [29] 6426 6134 6511 5421 5423 6276 6418 6180 5896 5537 5418 5524 6177 6393  
## [43] 6246 5523 5312 6183 5868 5832 5094 5093 5202 5336 6253 5984 5894 5331  
## [57] 5509 6733 6253 5326 5182 5284 5968 6028 6597 5556 5631 5275 6367 7173  
## [71] 6371 4736 5272 5237 6572 6472 6355 4543 4960 6199 6671 6593 5581 4537  
## [85] 5179 5063 4994 4890 5708 5888 6207 5758 5453 5395 5906 6098 6130 5029  
## [99] 5068 5123

diff\_12 <- diff(mydata, 12)  
diff\_12

## Time Series:  
## Start = 13   
## End = 100   
## Frequency = 1   
## [1] 106 -337 1160 235 -162 -434 177 637 -342 -564 -1183  
## [12] 231 -269 -375 -1811 -1400 -436 523 832 -1014 -938 -419  
## [23] 1407 1236 228 286 -30 -61 -249 259 -265 102 -111  
## [34] -93 -550 -348 -802 -444 -216 -188 76 -409 -352 -192  
## [45] 197 550 385 -506 88 191 766 692 344 -428 -263  
## [56] -56 858 440 118 -590 90 -47 604 444 -242 -1013  
## [67] -671 924 304 -580 -790 -199 -93 -174 -1578 -1582 -647  
## [78] 1345 1247 -441 -1218 -1198 325 1561 951 -34 74 233

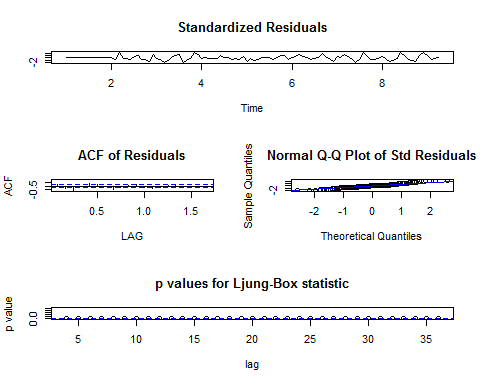
acf2(diff\_12, 48)



## ACF PACF  
## [1,] 0.41 0.41  
## [2,] -0.22 -0.47  
## [3,] -0.46 -0.20  
## [4,] 0.03 0.43  
## [5,] 0.44 0.12  
## [6,] 0.35 -0.08  
## [7,] -0.08 0.03  
## [8,] -0.40 -0.17  
## [9,] -0.18 0.07  
## [10,] 0.06 -0.24  
## [11,] 0.10 -0.19  
## [12,] -0.22 -0.22  
## [13,] -0.33 -0.04  
## [14,] -0.20 -0.09  
## [15,] 0.07 -0.02  
## [16,] 0.14 0.09  
## [17,] -0.05 0.01  
## [18,] -0.18 0.05  
## [19,] -0.14 -0.02  
## [20,] 0.05 -0.13  
## [21,] 0.15 0.02  
## [22,] 0.16 0.03  
## [23,] 0.03 -0.10  
## [24,] -0.13 -0.29  
## [25,] -0.10 0.00  
## [26,] 0.06 0.03  
## [27,] 0.23 0.03  
## [28,] 0.14 -0.02  
## [29,] -0.03 0.07  
## [30,] -0.18 0.01  
## [31,] -0.09 -0.01  
## [32,] 0.08 -0.10  
## [33,] 0.19 0.05  
## [34,] 0.08 -0.01  
## [35,] -0.08 0.03  
## [36,] -0.17 -0.22  
## [37,] -0.05 0.03  
## [38,] 0.13 0.12  
## [39,] 0.14 0.00  
## [40,] 0.03 0.11  
## [41,] -0.11 0.01  
## [42,] -0.15 -0.22  
## [43,] -0.07 -0.01  
## [44,] 0.06 -0.05  
## [45,] 0.11 -0.02  
## [46,] -0.02 -0.12  
## [47,] -0.15 0.03  
## [48,] -0.15 -0.03

mydata<-ts(mydata, freq=12)  
ace1<-sarima(mydata, 1,0,0,2,1,0,12)

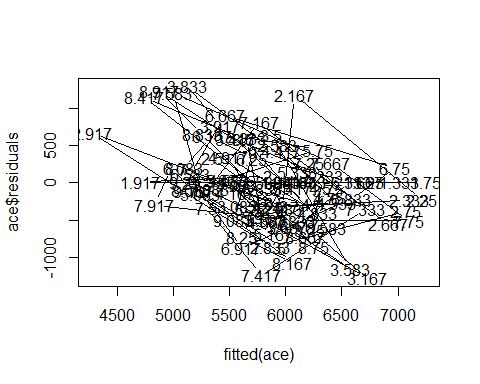
## initial value 6.442927   
## iter 2 value 6.277628  
## iter 3 value 6.273156  
## iter 4 value 6.272725  
## iter 5 value 6.272723  
## iter 6 value 6.272723  
## iter 6 value 6.272723  
## iter 6 value 6.272723  
## final value 6.272723   
## converged  
## initial value 6.373315   
## iter 2 value 6.369545  
## iter 3 value 6.369096  
## iter 4 value 6.369009  
## iter 5 value 6.368994  
## iter 6 value 6.368993  
## iter 6 value 6.368993  
## iter 6 value 6.368993  
## final value 6.368993   
## converged



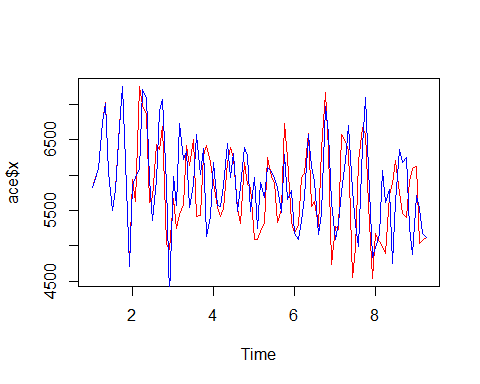
ace<-Arima(mydata,order=c(1, 0, 0),  
 seasonal=list(order=c(2, 1, 0), period=12))  
  
ace

## Series: mydata   
## ARIMA(1,0,0)(2,1,0)[12]   
##   
## Coefficients:  
## ar1 sar1 sar2  
## 0.4213 -0.2858 -0.3744  
## s.e. 0.0958 0.1194 0.1406  
##   
## sigma^2 estimated as 339116: log likelihood=-685.82  
## AIC=1379.63 AICc=1380.11 BIC=1389.54

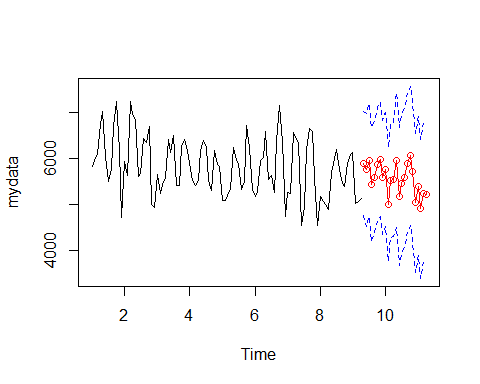
plot(fitted(ace), ace$residuals)



plot(ace$x, col='red')  
lines(fitted(ace), col='blue')



# Now, we have a reasonable prediction, we can forecast the model, say 24 months into the future.  
  
sarima.for(mydata, 24, 1,0,0,2,1,0,12)



## $pred  
## Jan Feb Mar Apr May Jun Jul Aug  
## 9 5897.148 5756.684 5973.110 5432.301  
## 10 5765.488 4993.503 5529.664 5537.997 5972.551 5177.381 5461.873 5585.953  
## 11 5405.542 4905.619 5250.381 5212.845   
## Sep Oct Nov Dec  
## 9 5596.200 5865.561 5994.268 5588.680  
## 10 5901.027 6063.650 5733.392 5042.389  
## 11   
##   
## $se  
## Jan Feb Mar Apr May Jun Jul Aug  
## 9 568.8625 613.7687 620.8282 621.9790  
## 10 622.2047 622.2047 622.2048 622.2048 737.8071 755.0902 757.8889 758.3473  
## 11 758.4372 758.4372 758.4372 758.4372   
## Sep Oct Nov Dec  
## 9 622.1677 622.1987 622.2038 622.2046  
## 10 758.4224 758.4348 758.4368 758.4371  
## 11

predict(ace, n.ahead=24)

## $pred  
## Jan Feb Mar Apr May Jun Jul Aug  
## 9 5961.243 5873.560 6097.938 5536.388  
## 10 5893.035 5103.857 5637.696 5648.759 6131.128 5374.081 5662.193 5764.840  
## 11 5604.671 5095.196 5447.188 5411.272   
## Sep Oct Nov Dec  
## 9 5686.537 5954.222 6108.799 5726.376  
## 10 6075.853 6242.978 5929.157 5248.093  
## 11   
##   
## $se  
## Jan Feb Mar Apr May Jun Jul Aug  
## 9 582.3368 631.9049 640.3017 641.7805  
## 10 642.0991 642.0991 642.0991 642.0991 765.0485 784.8601 788.3243 788.9375  
## 11 789.0698 789.0698 789.0698 789.0698   
## Sep Oct Nov Dec  
## 9 642.0426 642.0891 642.0974 642.0988  
## 10 789.0463 789.0656 789.0691 789.0697  
## 11

# Trend Analysis

Trend Analysis is the practice of collecting information and attempting to spot a pattern, or trend, in the information. Typically this involves analyzing the variance for a change over time. The null hypothesis: is that there is no trend. Many techniques can be used to identify trends, we'll use an ARMA model again.

# Trend Analysis in R

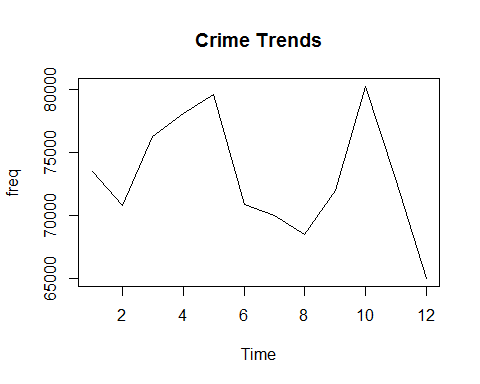
#----------- CRIME TRENDS --------  
crimeT <- count(mymodel, c("months"))

## Using freq as weighting variable

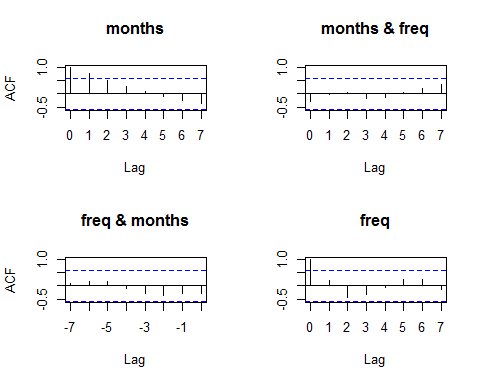
head(crimeT)

## months freq  
## 1 1 73534  
## 2 2 70811  
## 3 3 76279  
## 4 4 78084  
## 5 5 79640  
## 6 6 70882

plot(crimeT,type='l',xlab='Time',main='Crime Trends')



acf(crimeT)



var(crimeT)

## months freq  
## months 13.000 -5060.955  
## freq -5060.955 21529623.841

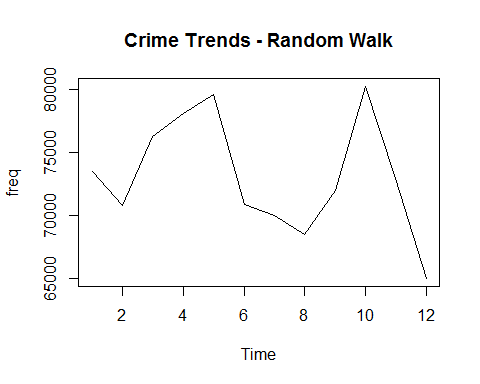
#------------- Random Walk -------------  
crime\_rw<- count(mymodel, c("months"))

## Using freq as weighting variable

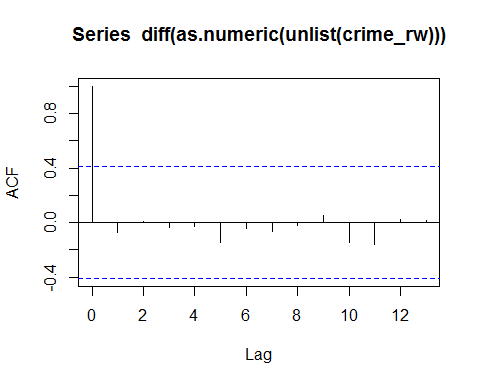
head(crime\_rw)

## months freq  
## 1 1 73534  
## 2 2 70811  
## 3 3 76279  
## 4 4 78084  
## 5 5 79640  
## 6 6 70882

plot(crime\_rw,type='l',xlab='Time',main='Crime Trends - Random Walk')



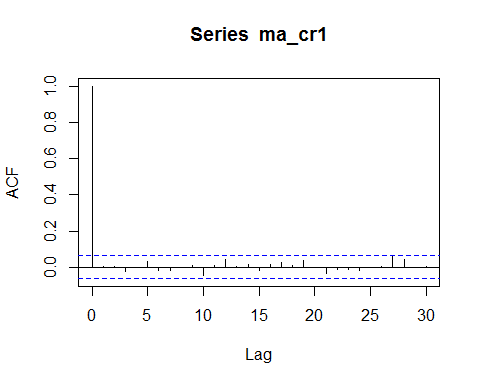
acf(diff(as.numeric(unlist(crime\_rw))))



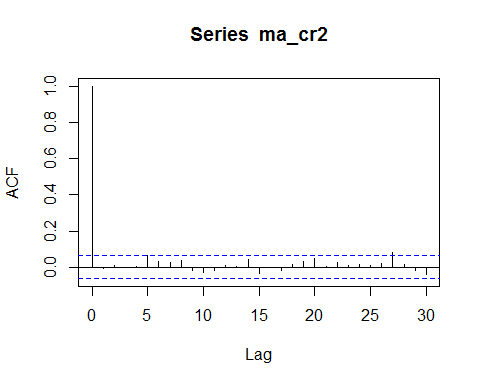
#----------------------------- ARMA model ----------------------  
  
# Moving Average Model  
ma\_cr1 <- arima.sim(model = list(crime\_rw, sd = 1.2), n = 1000)  
head(ma\_cr1, n = 8)

## Time Series:  
## Start = 1   
## End = 8   
## Frequency = 1   
## [1] -0.97700756 0.31973696 0.24106732 -0.03570051 0.77525226 -1.25094998  
## [7] 0.27299971 -0.51942474

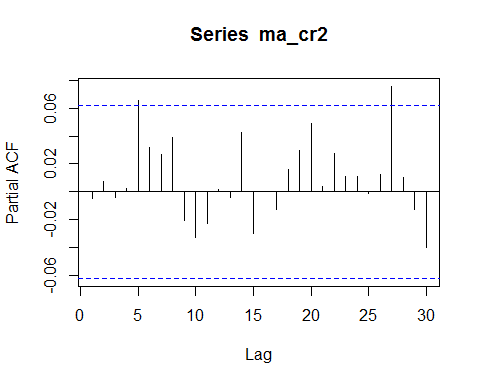
acf(ma\_cr1)



# Autoregressive model  
ma\_cr2 <- arima.sim(model = list(crime\_rw, sd = 1.2), n = 1000)  
acf(ma\_cr2)



pacf(ma\_cr2)



#--------------Dickey-Fuller for stationarity -----------------------  
crime\_rwS<- count(mymodel,c("Year", "months"))

## Using freq as weighting variable

adf.test(crime\_rwS$months, alternative = "stationary")

## Warning in adf.test(crime\_rwS$months, alternative = "stationary"): p-value  
## smaller than printed p-value

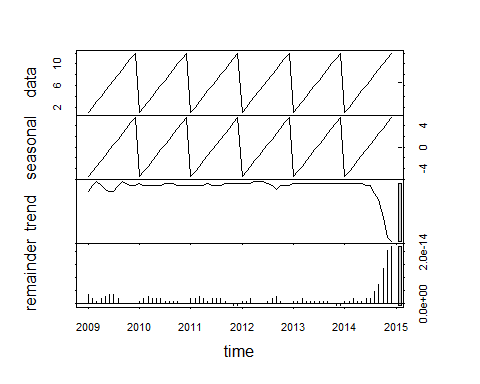
##   
## Augmented Dickey-Fuller Test  
##   
## data: crime\_rwS$months  
## Dickey-Fuller = -8.9938, Lag order = 5, p-value = 0.01  
## alternative hypothesis: stationary

#------------ Another unit root test : Philips-Perron test -------  
PP.test(crime\_rwS$months)

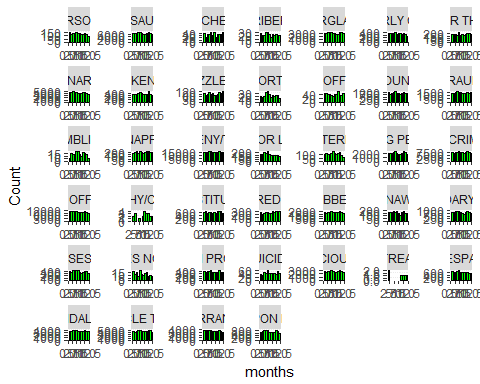
##   
## Phillips-Perron Unit Root Test  
##   
## data: crime\_rwS$months  
## Dickey-Fuller = -6.5431, Truncation lag parameter = 4, p-value =  
## 0.01

# ------------ Seasonal Trend Decomposition in R --------  
  
#The Seasonal Trend Decomposition using Loess (STL) is an algorithm that was developed   
#to help to divide up a time series into three components namely: the trend, seasonality and remainder.  
  
myts <- ts(crime\_rwS[,2], start=c(2009, 1), end=c(2014, 12), frequency=12)   
crime.stl <- stl(myts, s.window="periodic")  
plot(crime.stl)

## Warning in plot.window(xlim, ylim, log, ...): relative range of values = 17  
## \* EPS, is small (axis 2)



# ------------ Monthly Trend Decomposition in R --------  
  
ggplot(data=crime\_database, aes(x= months)) +  
 geom\_bar(colour="black", fill="green") +  
 ylab('Count') +  
 facet\_wrap(~Category, scales='free')



# ------------ Mapping in R --------  
  
library(rworldmap)

## Warning: package 'rworldmap' was built under R version 3.3.3

## Loading required package: sp

## Warning: package 'sp' was built under R version 3.3.3

## ### Welcome to rworldmap ###

## For a short introduction type : vignette('rworldmap')

library(ggmap)

## Warning: package 'ggmap' was built under R version 3.3.3

crime\_map<-crime\_database[1:2000,]  
  
  
qmplot(X, Y, data = crime\_map, colour = I('red'), size = I(3), darken = .3)

## Using zoom = 13...

## Map from URL : http://tile.stamen.com/toner-lite/13/1307/3165.png

## Map from URL : http://tile.stamen.com/toner-lite/13/1308/3165.png

## Map from URL : http://tile.stamen.com/toner-lite/13/1309/3165.png

## Map from URL : http://tile.stamen.com/toner-lite/13/1310/3165.png

## Map from URL : http://tile.stamen.com/toner-lite/13/1311/3165.png

## Map from URL : http://tile.stamen.com/toner-lite/13/1307/3166.png

## Map from URL : http://tile.stamen.com/toner-lite/13/1308/3166.png

## Map from URL : http://tile.stamen.com/toner-lite/13/1309/3166.png

## Map from URL : http://tile.stamen.com/toner-lite/13/1310/3166.png

## Map from URL : http://tile.stamen.com/toner-lite/13/1311/3166.png

## Map from URL : http://tile.stamen.com/toner-lite/13/1307/3167.png

## Map from URL : http://tile.stamen.com/toner-lite/13/1308/3167.png

## Map from URL : http://tile.stamen.com/toner-lite/13/1309/3167.png

## Map from URL : http://tile.stamen.com/toner-lite/13/1310/3167.png

## Map from URL : http://tile.stamen.com/toner-lite/13/1311/3167.png

## Map from URL : http://tile.stamen.com/toner-lite/13/1307/3168.png

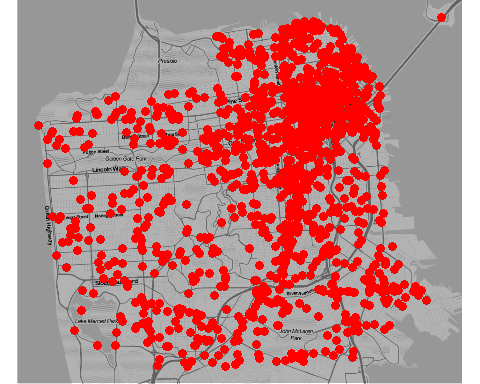
## Map from URL : http://tile.stamen.com/toner-lite/13/1308/3168.png

## Map from URL : http://tile.stamen.com/toner-lite/13/1309/3168.png

## Map from URL : http://tile.stamen.com/toner-lite/13/1310/3168.png

## Map from URL : http://tile.stamen.com/toner-lite/13/1311/3168.png

## Warning: `panel.margin` is deprecated. Please use `panel.spacing` property  
## instead

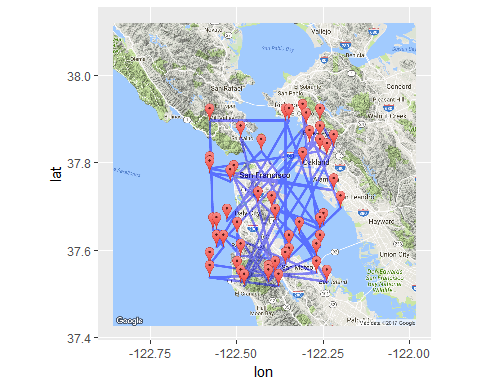


set.seed(500)  
  
df <- round(data.frame(  
 x = jitter(rep(-122.3951, 50), amount = .2),  
 y = jitter(rep( 37.7134, 50), amount = .2)  
), digits = 2)  
map <- get\_googlemap('San franscisco', markers = df, path = df, scale = 2)

## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=San+franscisco&zoom=10&size=640x640&scale=2&maptype=terrain&markers=37.61,-122.26%7C37.91,-122.31%7C37.7,-122.2%7C37.54,-122.41%7C37.59,-122.27%7C37.77,-122.51%7C37.55,-122.39%7C37.84,-122.22%7C37.9,-122.26%7C37.8,-122.31%7C37.52,-122.48%7C37.82,-122.24%7C37.85,-122.29%7C37.67,-122.53%7C37.89,-122.3%7C37.53,-122.49%7C37.67,-122.39%7C37.61,-122.35%7C37.58,-122.35%7C37.59,-122.49%7C37.65,-122.56%7C37.55,-122.27%7C37.54,-122.58%7C37.52,-122.38%7C37.86,-122.49%7C37.59,-122.27%7C37.64,-122.32%7C37.53,-122.24%7C37.57,-122.58%7C37.61,-122.54%7C37.65,-122.57%7C37.9,-122.58%7C37.9,-122.35%7C37.61,-122.56%7C37.71,-122.44%7C37.86,-122.26%7C37.83,-122.26%7C37.79,-122.58%7C37.74,-122.22%7C37.78,-122.58%7C37.76,-122.52%7C37.83,-122.43%7C37.65,-122.26%7C37.7,-122.4%7C37.66,-122.25%7C37.9,-122.36%7C37.57,-122.36%7C37.53,-122.41%7C37.64,-122.5%7C37.55,-122.5&path=37.61,-122.26%7C37.91,-122.31%7C37.7,-122.2%7C37.54,-122.41%7C37.59,-122.27%7C37.77,-122.51%7C37.55,-122.39%7C37.84,-122.22%7C37.9,-122.26%7C37.8,-122.31%7C37.52,-122.48%7C37.82,-122.24%7C37.85,-122.29%7C37.67,-122.53%7C37.89,-122.3%7C37.53,-122.49%7C37.67,-122.39%7C37.61,-122.35%7C37.58,-122.35%7C37.59,-122.49%7C37.65,-122.56%7C37.55,-122.27%7C37.54,-122.58%7C37.52,-122.38%7C37.86,-122.49%7C37.59,-122.27%7C37.64,-122.32%7C37.53,-122.24%7C37.57,-122.58%7C37.61,-122.54%7C37.65,-122.57%7C37.9,-122.58%7C37.9,-122.35%7C37.61,-122.56%7C37.71,-122.44%7C37.86,-122.26%7C37.83,-122.26%7C37.79,-122.58%7C37.74,-122.22%7C37.78,-122.58%7C37.76,-122.52%7C37.83,-122.43%7C37.65,-122.26%7C37.7,-122.4%7C37.66,-122.25%7C37.9,-122.36%7C37.57,-122.36%7C37.53,-122.41%7C37.64,-122.5%7C37.55,-122.5&sensor=false

## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=San%20franscisco&sensor=false

ggmap(map, extent = 'normal')

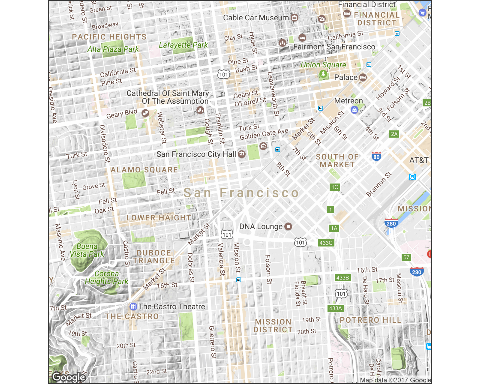


# find a reasonable spatial extent  
  
theme\_set(theme\_bw(16))  
  
qmap('San Francisco', zoom = 14)

## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=San+Francisco&zoom=14&size=640x640&scale=2&maptype=terrain&language=en-EN&sensor=false

## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=San%20Francisco&sensor=false

## Warning: `panel.margin` is deprecated. Please use `panel.spacing` property  
## instead



SanfranciscoMap <- qmap("San Francisco", zoom = 14, color = "bw",  
 extent = "device", legend = "topleft")

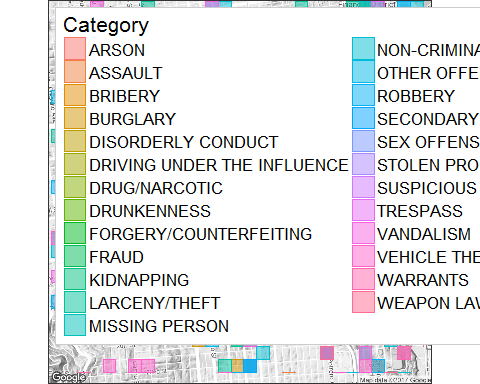
## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=San+Francisco&zoom=14&size=640x640&scale=2&maptype=terrain&language=en-EN&sensor=false  
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=San%20Francisco&sensor=false

## Warning: `panel.margin` is deprecated. Please use `panel.spacing` property  
## instead

SanfranciscoMap +  
 stat\_bin2d(  
 aes(x = X, y = Y, colour = Category, fill = Category),  
 size = .5, bins = 30, alpha = 1/2,  
 data = crime\_map  
 )

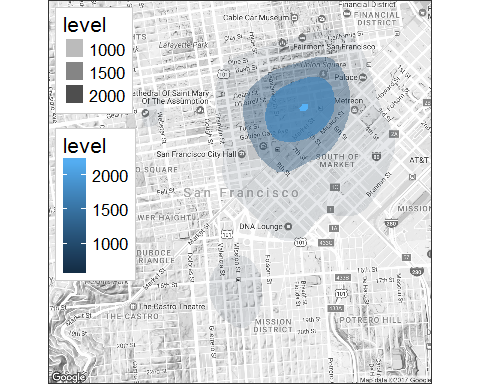
## Warning: Removed 933 rows containing non-finite values (stat\_bin2d).

## Warning: Removed 6 rows containing missing values (geom\_tile).



SanfranciscoMap +  
 stat\_density2d(  
 aes(x = X, y = Y, fill = ..level.., alpha = ..level..),  
 size = 2, bins = 4, data = crime\_map,  
 geom = "polygon"  
 )

## Warning: Removed 933 rows containing non-finite values (stat\_density2d).



SanfranciscoMap +  
 stat\_density2d(aes(x = X, y = Y, fill = ..level.., alpha = ..level..),  
 bins = 5, geom = "polygon",  
 data = crime\_map) +  
 scale\_fill\_gradient(low = "black", high = "red") +  
 facet\_wrap(~ DayOfWeek)

## Warning: Removed 933 rows containing non-finite values (stat\_density2d).

