CHAPTER 13

Crime Analyses Using R

Anindya Sengupta*, Madhav Kumar*, Shreyes Upadhyay[†]

Fractal Analytics, India, [†]Diamond Management and Technology Consultants, India

13.1 Introduction

The past couple of years have witnessed an overall declining trend in crime rate in the United States. This, in part, is attributable to the improvement in law enforcement strategies especially the inclusion of computer-aided technology for effective and efficient deployment of police resources. These advancements have been complemented by the availability of a vast amount of data and the capability to handle them. Many police departments have turned to data science to translate these bytes into actionable insights. Ranging from trend reports and correlation analyses to aggregated behavioral modeling, the developments in the field of crime analysis have paved the way for predictive policing and strategic foresight (Figure 13.1).

Predictive policing is an upcoming and growing area of research where statistical techniques are used to identify criminal hot-spots dynamically in order to facilitate anticipatory and precautionary deployment of police resources. However, the efforts that go into designing an effective predictive policing strategy involve a series of challenges. The most pertinent of these, especially concerning statisticians and analysts, is the one relating to data. How does one gather, process, and harness the copious real-time data? How does one develop a predictive engine that is simple enough to be understood and at the same time accurate enough to be useful? How does one implement a solution that gives stable results and yet updates dynamically? These are some of the concerns that we address in this chapter. We look at how crime data are stored, how they need to be handled, the different dimensions involved, the kind of techniques that would be applicable, and the limitations of such analysis. We look at all this, within the central theme of the powerful statistical programming language R.

The chapter is organized into eight sections including Section 13.1 After defining the problem in Section 13.2, we dive straight into the data through R with data extraction in Section 13.3. Data exploration and preprocessing in Section 13.4 provide details on how to deal with crime data in R and how to clean and process them for modeling. We visualize the data in Section 13.5 to get a better understanding of what we are dealing with and how we can reorganize it to get optimal results through modeling. In Sections 13.6 and 13.7, we build a

Data Mining Applications with R

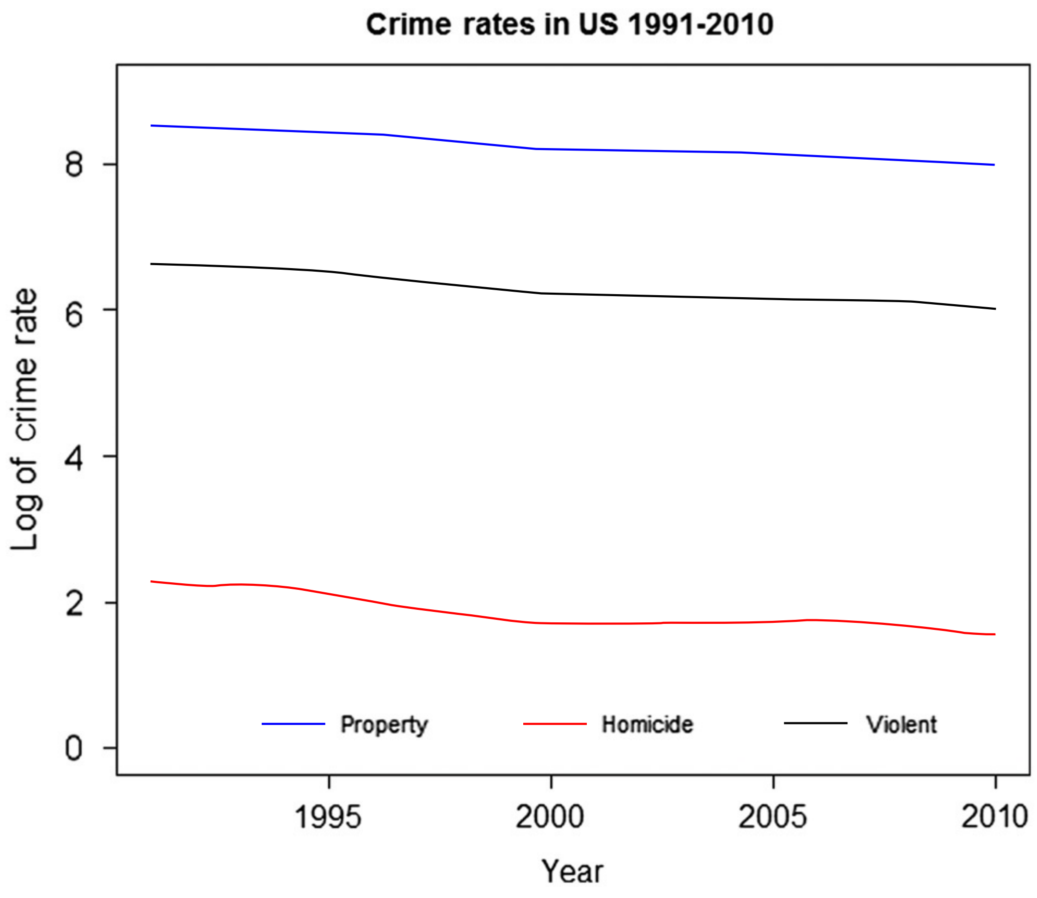

Figure 13.1
U.S. crime rates over time.

multivariate regression model to predict crime counts using historical crime data for the city of Chicago and evaluate its performance. Section 13.8 closes the chapter with discussions on the limitations of the model and the scope for improvement.

13.2 Problem Definition

Predictive policing is a multidimensional optimization problem where law enforcement agencies try to efficiently utilize a scarce resource to minimize instances of crime over time and across geographies. But how do we optimize? This is precisely what we answer in this chapter by using real and publicly available criminal data of Chicago. Using statistics and computer-aided technologies, we try to devise a solution for this optimization problem. In the attempt to address a real world problem, our primary focus here is on the fundamentals of data rather than on acrobatics with techniques.

Crime analysis includes looking at the data from two different dimensions—spatial and temporal. Spatial dimension involves observing the characteristics of a particular region along with its neighbors. Temporal dimension involves observing the characteristics of a particular region over time. The question then is how far away, from the epicenter, do we look for similar patterns and how far back in time, from the date of event, do we go to capture the trend. Ideally we would like to have as much data as possible. But we don't. And that makes data science a creative process. A process bound by mathematical logic and centered around statistical validity.

Crime data are not easy to deal with. With both spatial and temporal attributes, processing them can be a challenging task. The challenge is not limited to handling spatial and temporal data but also deriving information from them at these levels. Any predictive model for crime will have these two dimensions attached to it. And to make an effort toward effective predictive policing strategies, this inherent structure of the data needs to be leveraged.

13.3 Data Extraction

For this exercise, we use the crime data for the city of Chicago which are available from 2001 onwards on the city's open data portal. To make analysis manageable, we utilize the past one year of data from the current date.

R has the capability of reading files and data tables directly from the Web. We can do this by specifying the connection string instead of the file name in the read.csv() function.[1]

```
> url.data <- "https://data.cityofchicago.org/api/views/x2n5-
8w5q/rows.csv?accessType=DOWNLOAD"

> crime.data <- read.csv(url.data, na.strings= '')
```

13.4 Data Exploration and Preprocessing

Before we start playing with data, it is important to understand how the data are organized, what fields are present in the table, and how they are stored. str() is a useful command for this which displays the internal structure of the data neatly.

```
> str(crime.data)
'data.frame': 337793 obs. of 17 variables:
$ CASE.                  : Factor w/ 337775 levels "","223432","C431426",..:
64 71 5316 109 66 95 1526 133 65 2534 ...
$ DATE..OF.OCCURRENCE    : Factor w/ 121027 levels "1/1/2012 0:00",..: 71649
71650 71650 71651 71652 71652 71652 71653 71653 71654 ...
$ BLOCK                  : Factor w/ 28472 levels "0000X E 100TH PL",..: 3951
12626 16621 5479 889 16317 7765 11248 5132 7765 ...
$ IUCR                   : Factor w/ 323 levels "031A","031B",..: 301 307 23
173 291 192 25 226 45 23 ...
```

[1] The url can also be directly fed as input to read.csv() as `crime.data <- read.csv(https://data.cityofchicago.org/api/views/x2n5-8w5q/rows.csv?accessType=DOWNLOAD)`.

```
$ PRIMARY.DESCRIPTION    : Factor w/ 31 levels "ARSON","ASSAULT",..: 30 30 8 5
2 26 8 22 6 8 ...
$ SECONDARY.DESCRIPTION  : Factor w/ 303 levels "$500 AND UNDER",..: 1 240 96
230 252 180 145 299 277 96 ...
$ LOCATION.DESCRIPTION   : Factor w/ 117 levels "","ABANDONED BUILDING",..:
108 106 19 69 88 105 22 94 113 24 ...
$ ARREST                 : Factor w/ 2 levels "N","Y": 2 2 1 1 2 1 1 1 1 1 ...
$ DOMESTIC               : Factor w/ 2 levels "N","Y": 1 1 1 1 1 1 1 2 2 1 ...
$ BEAT                   : int 1322 1924 1922 1433 1832 1131 1922 2523 1011
1922 ...
$ WARD                   : int 32 44 47 1 42 24 47 31 24 47 ...
$ FBI.CD                 : Factor w/ 26 levels "01A","01B","04A",..: 24 24 8
17 5 20 8 21 11 8 ...
$ X.COORDINATE           : int 1163854 1169774 1165297 1164854 1174827
1147597 1163099 1145928 1151256 1163099 ...
$ Y.COORDINATE           : int 1906393 1921705 1930328 1909126 1905650
1897631 1931867 1918688 1893861 1931867 ...
$ LATITUDE               : num 41.9 41.9 42 41.9 41.9 ...
$ LONGITUDE              : num -87.7 -87.7 -87.7 -87.7 -87.6 ...
$ LOCATION               : Factor w/ 182456 levels "","(41.64471369097245, -
87.6128856478878)",..: 125420 155666 167007 132060 123200 104650 169311
150477 98599 169311 ...
```

str() is a compact version of summary(), which provides a detailed summary of the data.

```
> summary(crime.data)
     CASE.            DATE..OF.OCCURRENCE              BLOCK
HT572234:      3   1/1/2012 0:01 :    68    008XX  N  MICHIGAN AVE:      701
HT576362:      3   9/1/2011 9:00 :    62    001XX  N  STATE ST    :      582
HV217424:      3   12/1/2011 9:00:    50    008XX  N  STATE ST    :      519
HT341462:      2   3/1/2012 9:00 :    50    0000X  W  TERMINAL ST :      484
HT387816:      2   10/1/2011 9:00:    47    076XX  S  CICERO AVE  :      470
HT421725:      2   8/1/2011 9:00 :    47    0000X  N  STATE ST    :      444
(Other)  :337778   (Other)       :337469    (Other)               :334593
   IUCR           PRIMARY.DESCRIPTION
486     :  29717   THEFT            :72233
820     :  28160   BATTERY          :59892
460     :  20321   NARCOTICS        :37856
1811    :  20151   CRIMINAL DAMAGE  :36023
1320    :  16505   BURGLARY         :24827
610     :  16462   ASSAULT          :19492
(Other):206477     (Other)          :87470
LOCATION.DESCRIPTION                         ARREST          DOMESTIC
STREET                          : 79864      N:245198        N:289939
RESIDENCE                       : 53826      Y: 92595        Y: 47854
SIDEWALK                        : 41390
APARTMENT                       : 40530
OTHER                           : 11249
PARKING LOT/GARAGE(NON.RESID.) : 10370
(Other)                         :100564
     BEAT            WARD         FBI.CD         X.COORDINATE
Min.   : 111     Min.    :  1.00   6     :72233    Min.    :1094469
1st Qu.: 622     1st Qu. :10.00    08B   :51842    1st Qu.:1152885
```

```
Median :1032    Median :22.00   14     :36023    Median :1165987
Mean   :1188    Mean   :22.64   18     :35958    Mean   :1164633
3rd Qu.:1724    3rd Qu.:34.00   26     :30328    3rd Qu.:1176390
Max.   :2535    Max.   :50.00   5      :24827    Max.   :1205078
                 NA's :13      (Other):86582    NA's   :331
  Y.COORDINATE        LATITUDE      LONGITUDE
Min.   :1813949  Min.   :41.64    Min.   :-87.93
1st Qu.:1858248  1st Qu.:41.77    1st Qu.:-87.71
Median :1889370  Median :41.85    Median :-87.67
Mean   :1884885  Mean   :41.84    Mean   :-87.67
3rd Qu.:1908824  3rd Qu.:41.91    3rd Qu.:-87.63
Max.   :1951532  Max.   :42.02    Max.   :-87.52
NA's   :331      NA's   :331      NA's   :331
                                   LOCATION
(41.896757773203376, -87.62835155256259): 490
(41.754641620170695, -87.74138515773002): 464
(41.883559699067106, -87.62773649602941): 357
                                        : 331
(41.978893800537726, -87.90646758438514): 269
(41.909763298422995, -87.74331203591734): 266
(Other)                                 :335616
```

The data are stored at a crime incident level, that is, there is one observation for each crime incident in the data table. Each incident has a unique identifier associated with it which is stored in the CASE. variable. By definition then, CASE. should have all unique values. However, we see that some instances are duplicated, i.e., there are two or more rows which have the same case value. For example, there are three rows in the data that have a case value equal to HT572234. These duplicated rows need to be removed. We can do this using a combination of the subset() and the duplicated()[2] function.[3]

```
> crime.data <- subset(crime.data, !duplicated(crime.data$CASE.))

> summary(crime.data)
```

Most raw data sets will have issues like duplicated rows, missing values, incorrectly imputed values, and outliers. This is the case with our data as well with some of the variables having missing values.

Missing value imputation is a critical ingredient to any modeling recipe. Depending on the meaning and type of the variable, the missing values need to be substituted logically. There are certain cases, however, where missing value imputation does not apply. For example, the longitude and latitude variables in our data represent the coordinates of the location where the crime incident occurred. We cannot substitute these values using simple mathematical logic. In such a scenario, depending also on the percentage of rows with missing values, we can ignore these observations. A neat function to deal with rows which have missing values is the is.na function.

[2] A negation of a command can be used by prefixing it with an exclamation mark.

[3] Output has been suppressed due to space constraints.

```
> crime.data <- subset(crime.data, !is.na(crime.data$LATITUDE))
```

```
> crime.data <- subset(crime.data, !is.na(crime.data$WARD))
```

Another way to remove observations with missing rows from the data is to explicitly find the ones that have missing values and then remove those from the data frame. We can use the which() function to determine the rows with the missing values.

```
> which(is.na(crime.data$LOCATION))
```

```
> crime.data <- crime.data[-which(is.na(crime.data$LOCATION)), ]
```

In our data, we also have illogical values in certain rows. For example, one of the values in the CASE. variable is "CASE#." This seems to be some sort of a data storage or data import issue and we can safely ignore these rows.

```
> crime.data <- crime.data[crime.data$CASE. != 'CASE#',]
```

The date of occurrence gives an approximate date and time stamp as to when the crime incident might have happened. To see how this variable is stored, we can use the head() function which shows the first few observations of the data/column.

```
> head(crime.data$DATE..OF.OCCURRENCE)
[1] 5/22/2011 18:03 5/22/2011 18:05 5/22/2011 18:05 5/22/2011 18:06 5/22/2011 18:10 5/22/2011
18:10
121027 Levels: 1/1/2012 0:00 1/1/2012 0:01 1/1/2012 0:02 1/1/2012 0:03 1/1/2012 0:05 1/1/2012
0:06 ... 9/9/2011 9:58
```

Currently, date is stored as a factor variable. To make R recognize that it is in fact a date, we need to present it to R as a date object. One way to do this is using the as.POSIXlt() function.[4]

```
> crime.data$date <- as.POSIXlt(crime.data$DATE..OF.OCCURRENCE,
            format= "%m/%d/%Y %H:%M")
```

```
> head(crime.data$date)
[1] "2011-05-22 18:03:00" "2011-05-22 18:05:00" "2011-05-22 18:05:00" "2011-05-22 18:06:00"
"2011-05-22 18:10:00"
[6] "2011-05-22 18:10:00"
```

R can now understand that the data stored in the column are date and time stamps. Processing the data a bit further, we can separate the time stamps from the date part using the times() function in the chron (James and Hornik, 2011) library. The chron library provides a bunch of useful functions to handle dates and time in R.

```
> library(chron)
```

```
> crime.data$time <- times(format(crime.data$date, "%H:%M:%S"))
```

```
> head(crime.data$time)
```

```
[1] 18:03:00 18:05:00 18:05:00 18:06:00 18:10:00 18:10:00
```

[4] During our exercise, we downloaded the data several times and noticed inconsistency in the format in which time was stored. In case, time appears as 05:30 PM instead of 17:30, please use

```
> crime.data$date <- as.POSIXlt(crime.data$DATE..OF.OCCURRENCE,
            format= "%m/%d/%Y %I:%M %p").
```

The frequency of crimes need not be consistent throughout the day. There could be certain time intervals of the day where criminal activity is more prevalent as compared to other intervals. To check this, we can bucket the timestamps into a few categories and then see the distribution across the buckets. As an example, we can create four six-hour time windows beginning at midnight to bucket the time stamps. The four time intervals we then get are—midnight to 6AM, 6AM to noon, noon to 6PM, and 6PM to midnight.[5]

For bucketing, we first create variable bins using the four time intervals mentioned above. Once the bins are created, the next step is to map each timestamp in the data to one of these time intervals. This can be done using the cut() function.

```
> time.tag <- chron(times= c("00:00:00", "06:00:00", "12:00:00", "18:00:00",
           "23:59:00"))

> time.tag
[1] 00:00:00 06:00:00 12:00:00 18:00:00 23:59:00

> crime.data$time.tag <- cut(crime.data$time, breaks= time.tag,
            labels= c("00-06","06-12", "12-18", "18-00"), include.lowest=
            TRUE)

> table(crime.data$time.tag)
00-06   06-12    12-18    18-00
56161   76224   103336   101723
```

The distribution of crime incidents across the day suggests that crimes are more frequent during the latter half of the day.

With our timestamps in order and stored in a separate variable, we can recode the date variable to contain just the date part by stripping the timestamps.

```
> crime.data$date <- as.POSIXlt(strptime(crime.data$date,
            format= "%Y-%m-%d"))

> head(crime.data$date)
[1] "2011-05-22" "2011-05-22" "2011-05-22" "2011-05-22" "2011-05-22" "2011-05-22"
```

One of the core aspects of data mining is deriving increasingly more information from the limited data we have. We will see a few examples of what we mean by this as we go along. Let's start with something simple and intuitive.

We can use the date of incidence to determine which day of the week and which month of the year the crime occurred. It is possible that there is a pattern in the way crimes occur (or are committed) depending on the day of the week and month.

```
> crime.data$day <- weekdays(crime.data$date, abbreviate= TRUE)

> crime.data$month <- months(crime.data$date, abbreviate= TRUE)
```

[5] There is no theoretical reasoning behind choosing four 6-h buckets. This is just indicative of how to look for patterns in the data.

There are two fields in the data which provide the description of the crime incident. The first, primary description provides a broad category of the crime type and the second provides more detailed information about the first. We use the primary description to categorize different crime types.[6]

```
> table(crime.data$PRIMARY.DESCRIPTION)

                 ARSON                        ASSAULT                              BATTERY
                   465                          19484                                59855
              BURGLARY            CRIM SEXUAL ASSAULT                      CRIMINAL DAMAGE
                 24807                           1295                                35995
     CRIMINAL TRESPASS             DECEPTIVE PRACTICE                             GAMBLING
                  8338                          11772                                  759
              HOMICIDE   INTERFERE WITH PUBLIC OFFICER   INTERFERENCE WITH PUBLIC OFFICER
                   474                            168                                 1022
          INTIMIDATION                     KIDNAPPING                  LIQUOR LAW VIOLATION
                   161                            245                                  618
   MOTOR VEHICLE THEFT                      NARCOTICS                         NON-CRIMINAL
                 17704                          37828                                    2
             OBSCENITY    OFFENSE INVOLVING CHILDREN              OTHER NARCOTIC VIOLATION
                    29                           2095                                    5
         OTHER OFFENSE                  OTHER OFFENSE                         PROSTITUTION
                 18323                              2                                 2449
      PUBLIC INDECENCY           PUBLIC PEACE VIOLATION                              ROBBERY
                    16                           2957                                13389
           SEX OFFENSE                       STALKING                                THEFT
                   994                            186                                72115
     WEAPONS VIOLATION
                  3892

> length(unique(crime.data$PRIMARY.DESCRIPTION))
[1] 31
```

Discuss: how check that it's OK to group certain cases?
One Answer: average characteristics

The data contain about 31 crime types, not all of which are mutually exclusive. We can combine two or more similar categories into one to reduce this number and make the analysis a bit easier.[7]

```
> crime.data$crime <- as.character(crime.data$PRIMARY.DESCRIPTION)

> crime.data$crime <- ifelse(crime.data$crime %in% c("CRIM SEXUAL ASSAULT",
            "PROSTITUTION", "SEX OFFENSE"), 'SEX', crime.data$crime)

> crime.data$crime <- ifelse(crime.data$crime %in% c("MOTOR VEHICLE THEFT"),
            "MVT", crime.data$crime)

> crime.data$crime <- ifelse(crime.data$crime %in% c("GAMBLING", "INTERFERE
```

[6] At the time the data were downloaded, there was an issue with one row of data set which had primary description set to " Primary." We can remove the row using `> crime.data <- crime.data[crime.data$PRIMARY.DESCRIPTION != " PRIMARY",]`

[7] The statements below can be written as one ifelse() statement. They have been broken out to improve readability.

```
            WITH PUBLIC OFFICER", "INTERFERENCE WITH PUBLIC OFFICER", "INTIMIDATION",
            "LIQUOR LAW VIOLATION", "OBSCENITY", "NON-CRIMINAL", "PUBLIC PEACE VIOLATION",
            "PUBLIC INDECENCY", "STALKING", "NON-CRIMINAL (SUBJECT SPECIFIED)"),
            "NONVIO", crime.data$crime)

> crime.data$crime <- ifelse(crime.data$crime == "CRIMINAL DAMAGE", "DAMAGE",
            crime.data$crime)

> crime.data$crime <- ifelse(crime.data$crime == "CRIMINAL TRESPASS",
            "TRESPASS", crime.data$crime)

> crime.data$crime <- ifelse(crime.data$crime %in% c("NARCOTICS", "OTHER
            NARCOTIC VIOLATION", "OTHER NARCOTIC VIOLATION"), "DRUG", crime.data$crime)

> crime.data$crime <- ifelse(crime.data$crime == "DECEPTIVE PRACTICE",
            "FRAUD", crime.data$crime)

> crime.data$crime <- ifelse(crime.data$crime %in% c("OTHER OFFENSE", "OTHER
            OFFENSE"), "OTHER", crime.data$crime)

> crime.data$crime <- ifelse(crime.data$crime %in% c("KIDNAPPING", "WEAPONS
            VIOLATION", "OFFENSE INVOLVING CHILDREN"), "VIO", crime.data$crime)

> table(crime.data$crime)
ARSON   ASSAULT   BATTERY   BURGLARY   DAMAGE    DRUG      FRAUD   HOMICIDE
  465     19484     59855      24807    35995   37833      11772        474
  MVT    NONVIO     OTHER    ROBBERY      SEX   THEFT   TRESPASS        VIO
17704      5918     18325      13389     4738   72115       8338       6232
```

In addition, we also have data on whether the crime incident led to an arrest or not. This is currently stored with Yes/No inputs. For now, let us convert this to a binary numeric variable. We will come back to it later and see how we can use it.

```
> crime.data$ARREST <- ifelse(as.character(crime.data$ARREST) == "Y", 1, 0)
```

With a couple of basic variables in place, we can start with a few visualizations to see how, when, and where are the crime incidents occurring.

13.5 Visualizations

Visualizing data is a powerful way to derive high-level insights about the underlying patterns in the data. Visualizations provide helpful clues as to where we need to dig down deeper for the gold. To see a few examples, we start with some simple plots of variables we processed in the previous section using the powerful ggplot2 (Wickham, 2009) library.

```
> library(ggplot2)

> qplot(crime.data$crime, xlab = "Crime", main ="Crimes in Chicago) +
            scale_y_continuous("Number of crimes")
```

qplot() is a quick plotting function from the ggplot2 library which works similar to the plot() function in the base package (Figure 13.2).

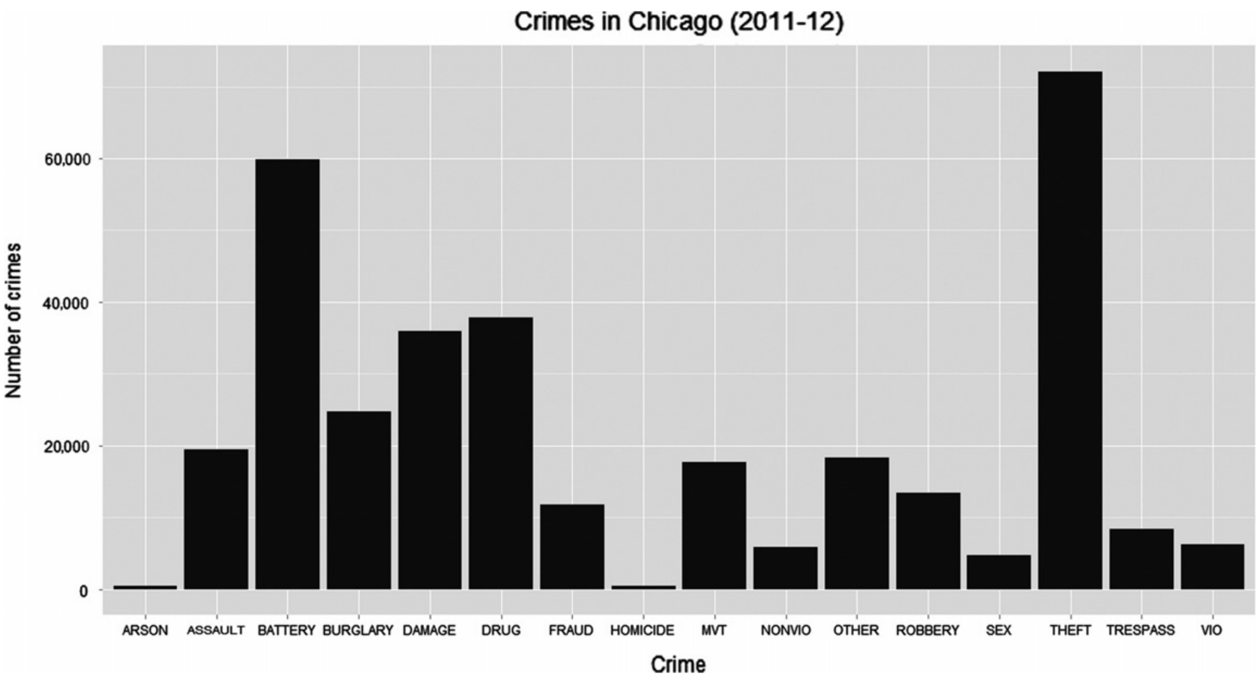

Figure 13.2
Frequency of different crimes in Chicago (2011-2012).

Prevalence of different crimes seems to be unevenly distributed in Chicago with theft and battery being much more frequent. It would be interesting to look at how crimes are distributed with respect to time of day, day of week, and month (Figures 13.3–13.5).

```
> qplot(crime.data$time.tag, xlab="Time of day", main = "Crimes by time of
          day") + scale_y_continuous("Number of crimes")

> crime.data$day <- factor(crime.data$day, levels= c("Mon", "Tue", "Wed",
          "Thu", "Fri", "Sat", "Sun"))
> qplot(crime.data$day, xlab= "Day of week", main= "Crimes by day of week") +
          scale_y_continuous("Number of crimes")

> crime.data$month <- factor(crime.data$month, levels= c("Jan", "Feb", "Mar",
          "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"))

> qplot(crime.data$month, xlab= "Month", main= "Crimes by month") +
          scale_y_continuous("Number of crimes")
```

There does seem to a pattern in the occurrence of crime with respect to the dimension of time. The latter part of the day, Fridays, and summer months witness more crime incidents, on average, with respect to other corresponding time periods.

These plots show the combined distribution of all the crime with respect to different intervals of time. We can demonstrate the same plots with additional information by splitting out the different crime types. For example, we can see how different crimes vary by different times of the day. To get the number of different crimes by time of day, we need to roll-up the data

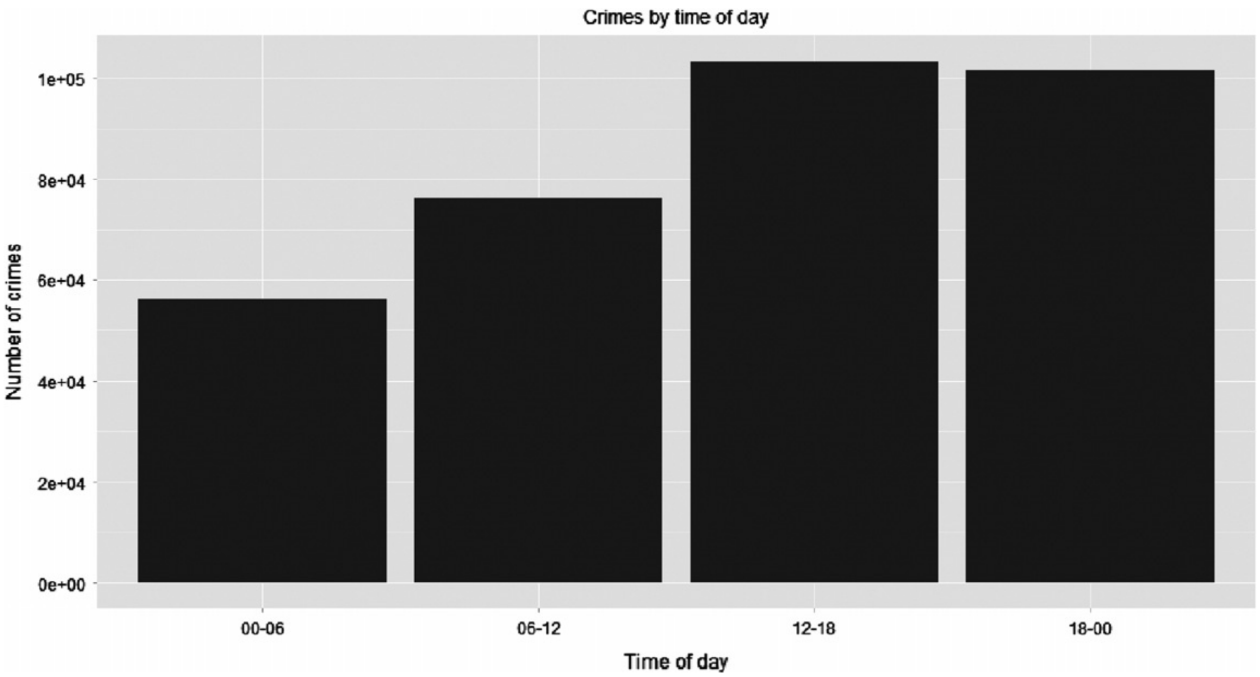

Figure 13.3
Distribution of crime by time of day.

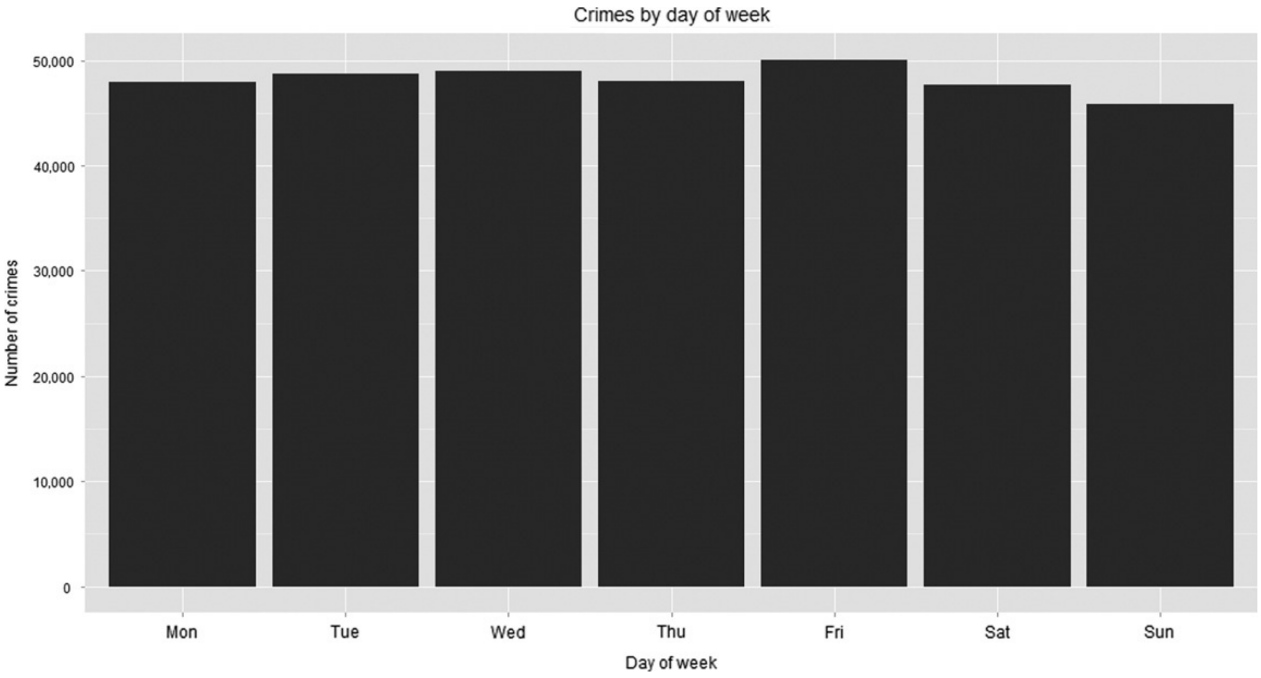

Figure 13.4
Distribution of crimes by day of week.

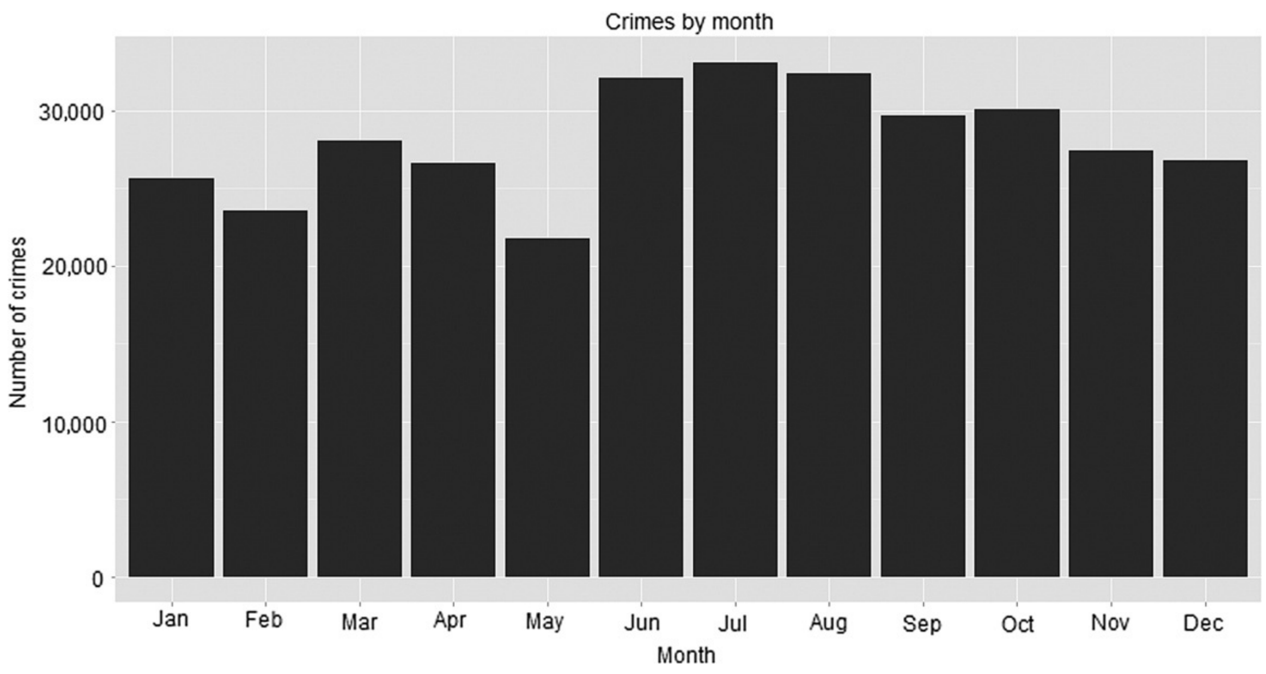

Figure 13.5
Distribution of crimes by month.

at a crime – time.tag level. That is, four rows for each crime type – one for each time interval of the day. An easy way to roll-up data is by using the aggregate() function (Figure 13.6).

```
> temp <- aggregate(crime.data$crime, by= list(crime.data$crime,
            crime.data$time.tag), FUN= length)
> names(temp) <- c("crime", "time.tag", "count")
```

To construct the plot, we use the ggplot() function from the ggplot2 (Wickham, 2009) library.

```
> ggplot(temp, aes(x= crime, y= factor(time.tag))) +
      geom_tile(aes(fill= count)) + scale_x_discrete("Crime", expand = c(0,0)) +
      scale_y_discrete("Time of day", expand = c(0,-2)) +
      scale_fill_gradient("Number of crimes", low = "white", high = "steelblue") +
      theme_bw() + ggtitle("Crimes by time of day") +
      theme(panel.grid.major = element_line(colour = NA), panel.grid.minor = element_line
      (colour = NA))
```

A quick look at the heat map above shows that most of the theft incidents occur in the afternoon whereas drug related crimes are more prevalent in the evening.

We can do a similar analysis by day of week and month as well and see the distribution of crimes with respect to these parameters. For this purpose, we show two other powerful aggregating functions in R. One is called ddply() from the plyr (Wickham, 2011) library and the other is summaryBy() from the doBy (Højsgaard and Ulrich, 2012) library. Note that all these three functions, in this case, serve the same purpose and can be used interchangeably (Figures 13.7 and 13.8).

this is a very useful library and method.

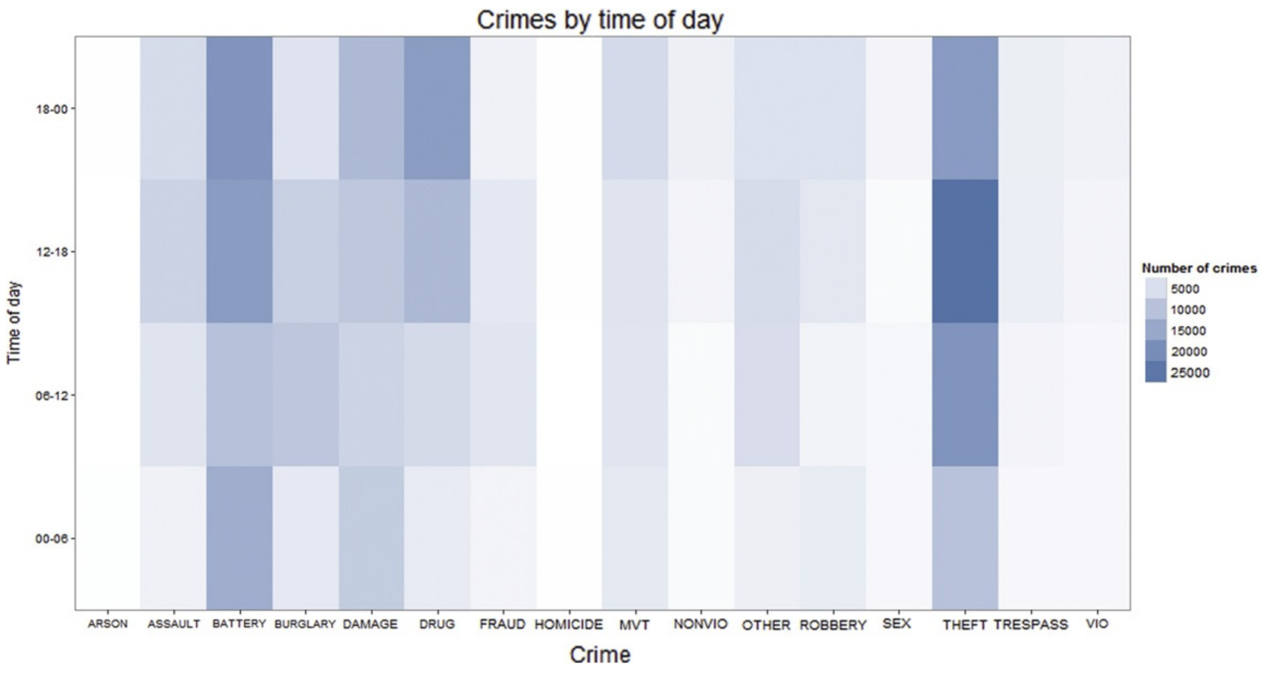

Figure 13.6
Heat map of different crimes by time of day.

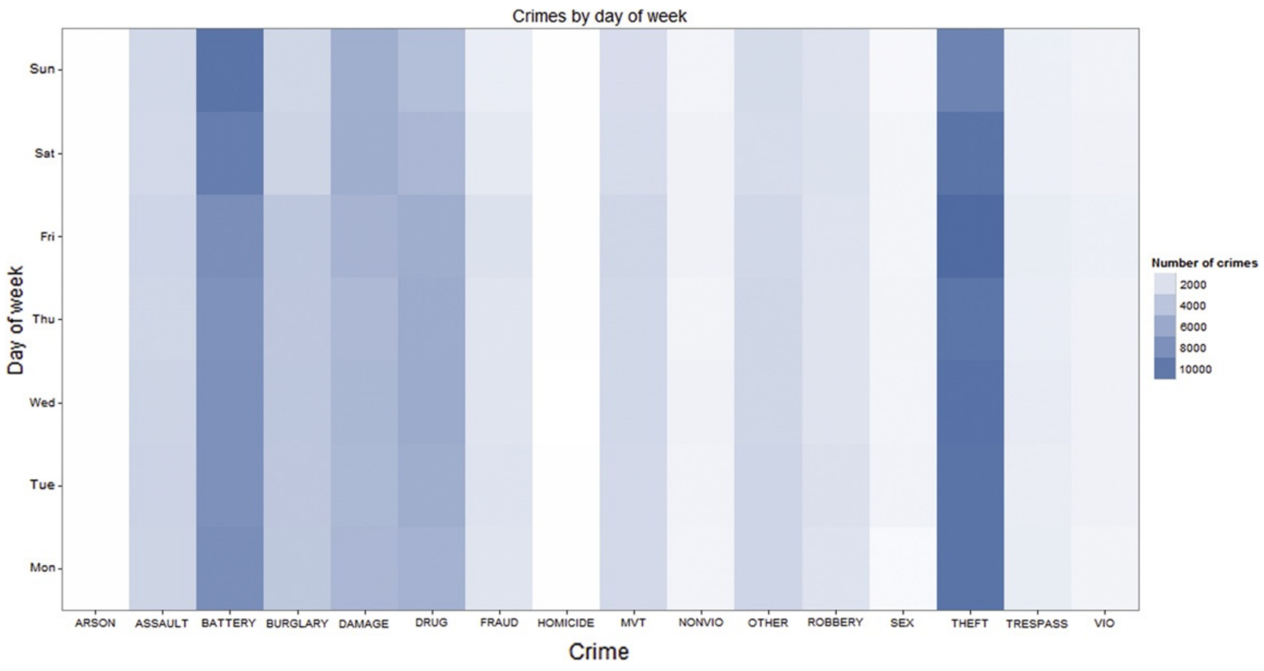

Figure 13.7
Heat map of different crimes by day of week.

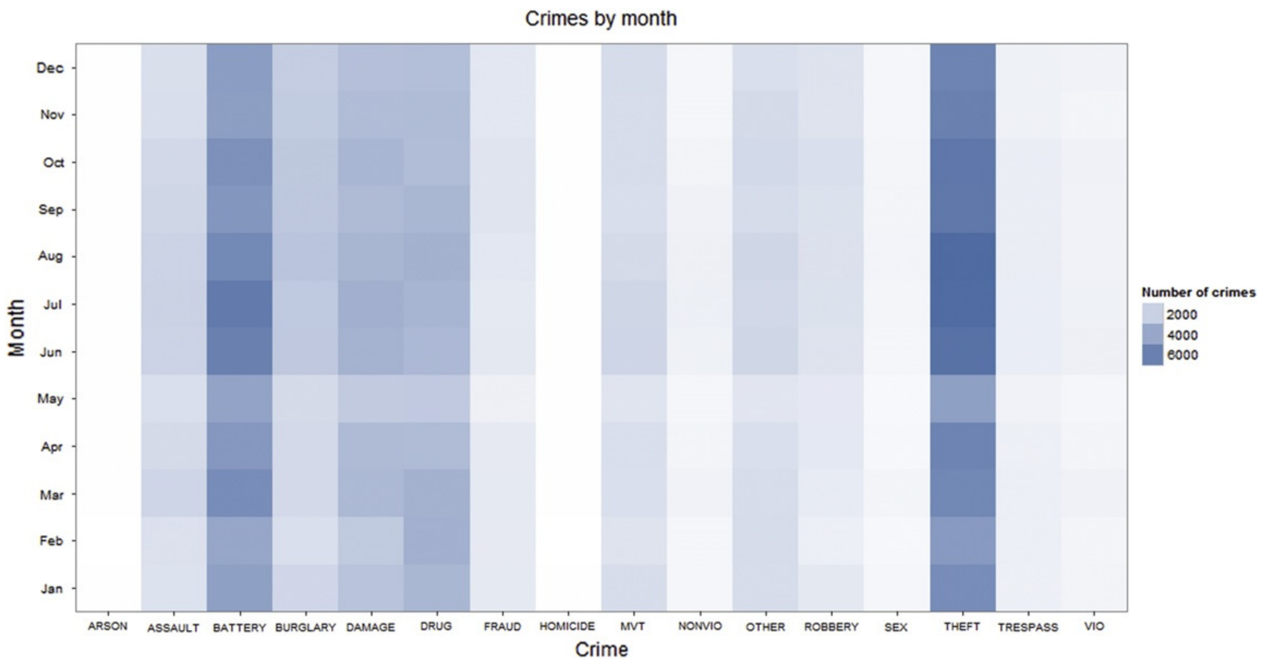

Figure 13.8
Heat map of different crimes by month.

```
> library(plyr)

> temp <- ddply(crime.data, .(crime, day), summarise, count = length(date))

> ggplot(temp, aes(x= crime, y= day, fill= count)) +
      geom_tile(aes(fill= count)) +
      scale_x_discrete("Crime", expand = c(0,0)) +
      scale_y_discrete("Day of week", expand = c(0,-2)) +
      scale_fill_gradient("Number of crimes", low = "white", high =
      "steelblue") +
      theme_bw() + ggtitle("Crimes by day of week") +
      theme(panel.grid.major = element_line(colour = NA), panel.grid.minor =
      element_line(colour = NA))

> library(doBy)

> temp <- summaryBy(CASE. ~ crime + month, data= crime.data, FUN= length)
      names(temp)[3] <- 'count'

> ggplot(temp, aes(x= crime, y= month, fill= count)) +
      geom_tile(aes(fill= count)) +
      scale_x_discrete("Crime", expand = c(0,0)) +
      scale_y_discrete("Month", expand = c(0,-2)) +
      scale_fill_gradient("Number of crimes", low = "white", high = "steelblue") +
      theme_bw() + ggtitle("Crimes by Month") + theme(panel.grid.major = element_line
      (colour = NA), panel.grid.minor = element_line(colour = NA))
```

Till now we have only looked at the temporal distribution of crimes. But there is also a spatial element attached to them. Crimes vary considerably with respect to geographies. Typically, within an area like a zip code, city, or county, there will be pockets or zones which observe higher criminal activity as compared to the others. These zones are labeled as crime hot-stops and are often the focus areas for effective predictive policing. We have the location of each crime incident in our data that can be used to look for these spatial patterns in the city of Chicago. For this purpose, we will utilize the shape files for Chicago Police Department's beats[8] by processing them in R using the maptools (Lewin-Koh and Bivand, 2012) library.

```
> library(maptools)
> beat.shp <- readShapePoly("PoliceBeats/PoliceBeat.shp")
> plot(beat.shp)
```

If we plot the shape file, we get the city of Chicago cut up into beats. We can plot our crime incidences by mapping them to the coordinates of the shape files (Figure 13.9).

To add more value to our analysis, we can also plot the locations of the police stations in Chicago and see if there is any relation between hot-spots of crime and proximity of police stations.

```
> station.shp <- readShapePoly("Police_20Stations-2/police_stations.shp")
```
[9]

Unfortunately, we don't have exact locations of the police stations in a ready format. They are embedded in the shape files and need to be extracted out. We have written a function to do just that. It takes a shape file as an input and returns the location (lat and long) of each police station in Chicago as the input.

```
> getPoliceData <- function(shp.file){
            PoliceData <- data.frame()
            NumOfStations = nrow(shp.file @data)
                  for(ii in 1:NumOfStations){
                   PoliceData[ii, 1] <- shp.file@data$DESCRIPTIO[ii]
                   PoliceData[ii, 2] <- shp.file@polygons[[ii]]@labpt[1]
                   PoliceData[ii, 3] <- shp.file@polygons[[ii]]@labpt[2]
                  }
            names(PoliceData) <- c("description", "lat", "long")
            return(PoliceData)
}
> police.data <- getPoliceData(station.shp)
```

Before we can plot the data on the map, we need to process it a bit further and get crime counts for each location. ddply() from the plyr library is a convenient function for this. ddply() takes a portion of a data frame, applies a function to that portion, and returns the result back in the form of a data frame. It works similar to the aggregate() function in the base package but is more flexible, versatile, and powerful.

[8] The shape files for CPD beats can be downloaded from https://data.cityofchicago.org/Public-Safety/Boundaries-Police-Beats/kd6k-pxkv.

[9] Shape files for CPD police stations can be downloaded from https://data.cityofchicago.org/Public-Safety/Police-Stations-Shapefiles/tc9m-x6u6.

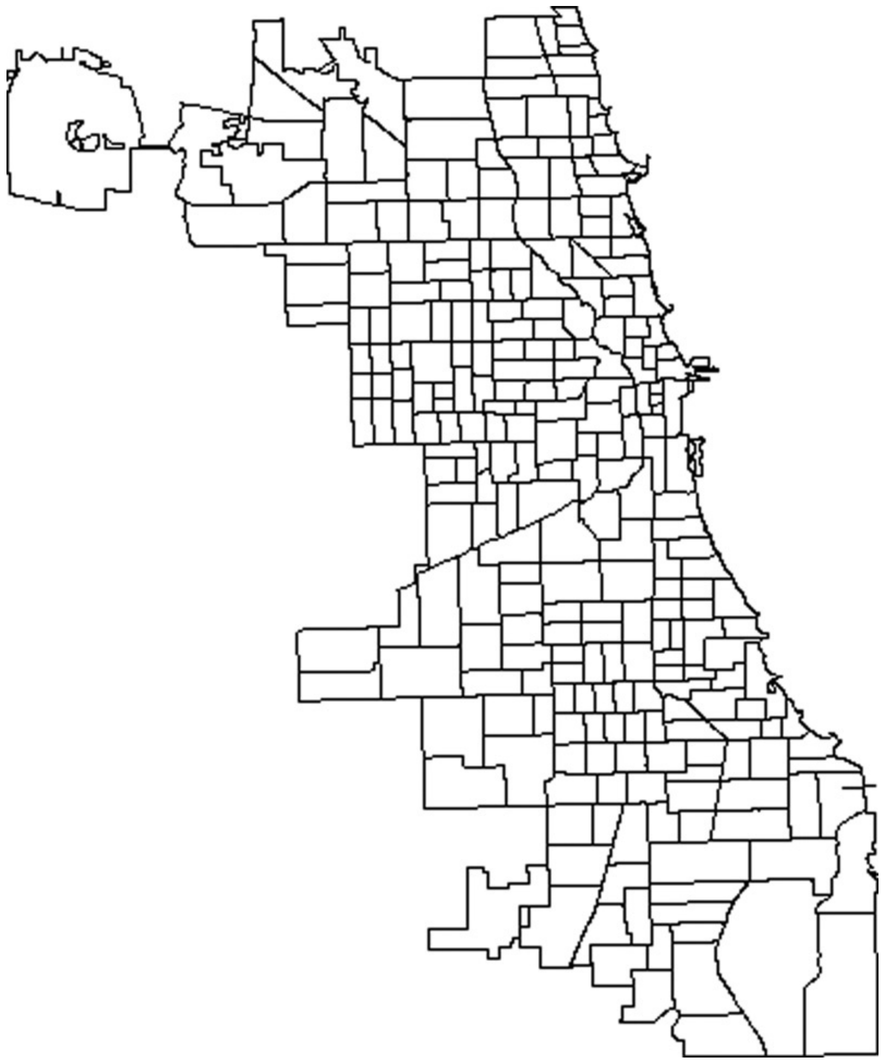

Figure 13.9
Chicago city police beats.

```
> crime.agg <- ddply(crime.data, .(crime, ARREST, BEAT, date, X.COORDINATE,
            Y.COORDINATE, time.tag, day, month), summarise, count = length(date),
            .progress= 'text')
```

We are going to utilize ggplot2's (Wickham, 2009) extensive plotting capabilities for this exercise as well. To plot the shape file using ggplot(), we need to convert the data in the shape file into a data frame using the fortify.SpatialPolygons() function.

```
> source('fortify.R')
```

```
> beat.shp.df <- fortify.SpatialPolygons(beat.shp)
```
[10]

[10] The function fortify.SpatialPolygons() has been written by Hadley Wickham and is available on his GitHub page https://github.com/hadley/ggplot2/blob/master/R/fortify-spatial.r.

```
> crime.plot <- ggplot(beat.shp.df, aes(x=long, y=lat)) +
           geom_path(aes(group = group)) + theme_bw() +
           geom_point(data= police.data, aes(x= lat, y= long))
```

To see how the crimes in Chicago are spread out spatially on a given day, we take the first date in our data set and the subset data for observations relating to this date only. In addition, to plot the crime incidents on the map, we convert our coordinates from factor to numeric (Figure 13.10).

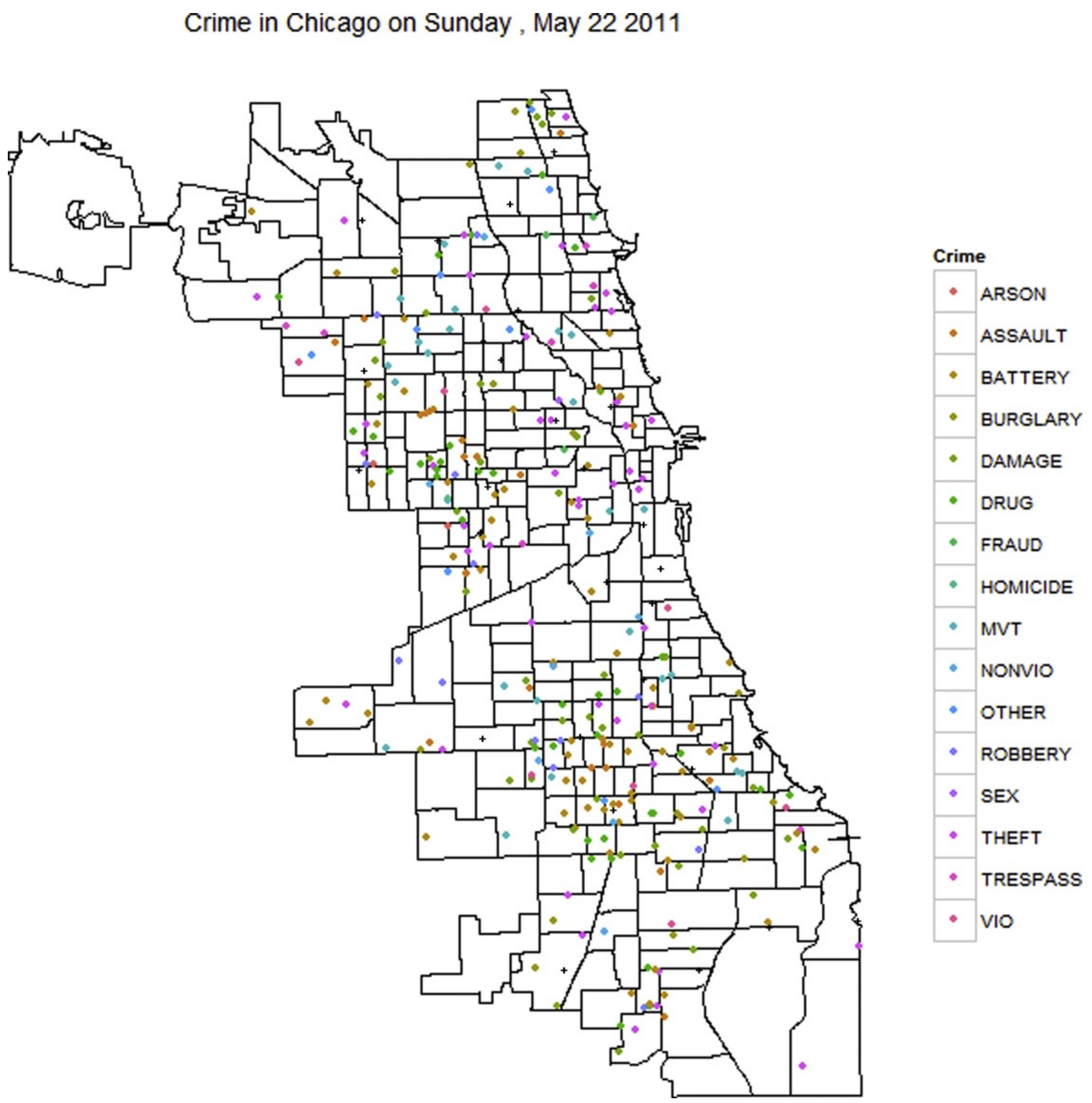

Figure 13.10
Crimes in Chicago on May 22, 2011.

```
> today <- crime.agg[1, 'date']

> crime.agg$X.COORDINATE <- as.numeric(crime.agg$X.COORDINATE)
> crime.agg$Y.COORDINATE <- as.numeric(crime.agg$Y.COORDINATE)

> crime.plot <- crime.plot + geom_point(data= subset(crime.agg,
            as.character(crime.agg$date) == today),
            aes(x=X.COORDINATE, y= Y.COORDINATE, color= crime))

> crime.plot <- crime.plot + theme_bw() +
            ggtitle(paste("Crime in Chicago on", weekdays(today), ",",
            months(today), substr(today, 9, 12), substr(today, 1, 4))) +
            theme(panel.grid.major= element_blank(), panel.grid.minor= element_blank(),
            panel.border= element_blank(),
            axis.title.x= element_blank(), axis.title.y= element_blank(),
            axis.text.x= element_blank(), axis.text.y= element_blank(), axis.ticks=
            element_blank())
> crime.plot
```

The "+" symbols on the plot are the police stations, whereas the other dots represent different crimes. A quick look at the plot indicates that quite a few police stations are located in and around high-crime areas except for the ones in the south-east corner where the incidents of crime appear much lesser. This plot, however, just shows the crime instances for one particular day. A better way to understand the pattern would be look at these crimes for a larger time period, for example, a month or a year. However, plotting the data for the entire year on one plot would not be visually conclusive since there will be too many data points over-crowding the limited space. An effective alternative is to plot the crimes for each day and see how the pattern is changing. We, however, don't want to run the code above for 365 days and then look at 365 different files. This is where R's powerful plotting capabilities come out in full glory. We can use the animation (Xie, 2012) library to create a rolling window plot that updates for each date, plots crimes for the entire year, and provides only one file as the output.

We plot the crimes by taking the subset of the data for each date in the data set, plotting it, saving it, and repeating the process again. We then use the saveMovie()[11] function in the animation package to convert the temporarily saved png files into a gif image.[12]

```
> back.plot <- ggplot(beat.shp.df, aes(x=long, y=lat)) +
            geom_path(aes(group = group)) + theme_bw() +
            geom_point(data= police.data, aes(x= lat, y= long))

> crime.plot <- function(dates){
      for (today in dates){
            date.crime <- crime.agg[as.character(crime.agg$date) == today, ]
            today <- as.Date(today, origin = "1970-01-01")
```

[11] saveMovie() requires ImageMagick which can be downloaded from http://www.imagemagick.org/script/index.php.

[12] A visually more appealing plot on the same lines has been made by Drew Conway and can be found on his blog at http://www.drewconway.com/zia/?p=2741.

```
            date.plot <- back.plot + theme_bw() +
            ggtitle(paste("Crime in Chicago on", weekdays(today), ",", months(today),
            substr(today, 9, 12), substr(today, 1, 4))) +
            theme(panel.grid.major=element_blank(), panel.grid.minor=element_blank(),
            panel.border= element_blank(),
            axis.title.x= element_blank(), axis.title.y= element_blank(),
            axis.text.x= element_blank(), axis.text.y= element_blank(),
            axis.ticks=element_blank())

            date.plot <- date.plot + geom_point(data= date.crime,
            aes(x=X.COORDINATE, y= Y.COORDINATE, color= crime))
            + scale_size(guide= "none", range= c(1.5,2.5)) +
            scale_alpha(guide= "none", range=c(0.3,0.5)) +
            scale_colour_discrete(name="Type of Crime") +
            xlab("") + ylab("")

      plot(date.plot)
  }
}

> dates <- as.character(unique(crime.agg$date))

> saveMovie(crime.plot(dates), movie.name= "chicago_crime_animation.gif",
            img.name= "Rplot", convert= "convert", cmd.fun= system, clean= TRUE, outdir=
      getwd(), interval = 1, width = 580, height = 400)
```

We have seen some interesting visualizations that helped us gauge the data better and provide clues to potential predictor variables for the model. Visualizations, like these and many others, are a great resource for data explorers as they guide us towards information hidden deep down somewhere among these large piles of data.

13.6 Modeling

Before we get to our model, we need to process the data a bit more. Specifically, we need to determine at what level of data are we going to build the model and what kind of independent variables can we derive from the existing data set.

The data we have are at a crime incident level, that is, for each recorded each crime incident in the city of Chicago that occurred in the past 12 months, we have the location, date, type, beat, and ward. The location is available at a very granular level with the exact geographical coordinates present alongside the larger territorial identity of police beats. Such information is vital, but it needs to be understood that not all information can be directly transformed into an implementable solution. We cannot predict exactly when and where the next crime is going to happen. Data scientists don't make prophecies. They derive actionable insights in the form of statistical aggregates. Therefore, understanding the nature of the problem is critical for designing a workable solution within a given set of constraints.

Keeping the ultimate goal of predictive policing in mind, our modeling engine would need to be constructed at a reasonably sized predefined geographical area for reasonably long predefined time interval. For example, the geographical area can be a block, or a couple of blocks taken together, a zip code, a beat etc. Similarly, the time interval can be an hour or a couple hours or a day.

For our purpose, we will build a multivariate regression model with a negative binomial distribution for errors[13] at a beat-day level for all the crimes taken together. It is debatable whether building the model in this fashion is the right approach or not. We agree to the fact that in terms of accuracy, a better solution would be to deal with different crime types separately and generate predictions for smaller time periods. However, this is just an indicative exercise which has been broken down for understanding and at times, brevity. Following these lines, we try to explain how a solution can be designed through a well suited approach without worrying too much about the exact details and metrics.

Maybe, but not necess arily.

To create the modeling data set, we first create our base data set which has all possible unique combinations of beats and dates. The number of rows for the base data will be the product of the total number of unique beats and total number of unique dates.

Notice you are creating an entirely new data set out of the original data!

```
> length(unique(crime.agg$BEAT))
[1] 297

> length(unique(crime.agg$date))[1] 359

> beats <- sort(unique(crime.agg$BEAT))

> dates <- sort(as.character(unique(crime.agg$date)))
```

The function expand.grid() will help create the desired data frame with the unique combination of beats and dates.

```
> temp <- expand.grid(beats, dates)

> names(temp) <- c("BEAT", "date")
```

For convenience, we can sort/order the data frame by beats

```
> temp <- orderBy(~ BEAT, data= temp)
```

The total number of crimes in a beat on a given day can be found by aggregating the data by beats and dates.[14]

```
> model.data <- aggregate(crime.agg[, c('count', 'ARREST')], by=
              list(crime.agg$BEAT, as.character(crime.agg$date)), FUN= sum)

> names(model.data) <- c("BEAT", "date", "count", "ARREST")
```

We then overlap the aggregated crimes data with the temp data set we created to get our final modeling data set.

```
> model.data <- merge(temp, model.data, by= c('BEAT', 'date'), all.x= TRUE)

> View(model.data)
```

[13] There are many approaches that can be taken to develop crime prediction models. We chose a negative binomial model because of its simplicity, general acceptance and understanding, and ease of implementation.

[14] Note that we could have used the ddply() function here as well.

Our modeling data set has all the crime incidents that were recorded in the past 12 months. However, during this period, there were beats in which no crime occurred (or was not recorded) on certain days. For these rows, we see NAs in the count and arrest fields. We can replace the NAs with zero to indicate that there were no crimes, and therefore no arrests, in these beats on these days.

```
> model.data$count[is.na(model.data$count)] <- 0
```

```
> model.data$ARREST[is.na(model.data$ARREST)] <- 0
```

We saw in the visualization exercise that there was variation in the crime incidents with respect to days of the week and month, suggesting that they can be used as predictor variables in the model. These variables can be created using the weekdays() and months() functions, respectively.

```
> model.data$day <- weekdays(as.Date(model.data$date), abbreviate= TRUE)
```

```
> model.data$month <- months(as.Date(model.data$date), abbreviate= TRUE)
```

A potential important indicator of criminal activity in a particular area could be the history of criminal activities in the past. We can calculate the crime history of each beat for different time periods. To calculate the crime histories, we will use the group averages function ave() and supplement it with a customized input function that will help us look back in time for any number of days by supplying only one argument.

The pastDays() function below creates a vector beginning with a zero followed a number ones, with the ones indicating how many days in the past do we need to look back. For example, to look back at the crime history of the past week, we need to supply 7 as an input to the pastDays() function.

```
> pastDays <- function(x) {
            c(0, rep(1, x))
            }
```

The pastDays() function is supplied further to the ave() function to calculate the number of crimes in the past for each beat.

```
> model.data$past.crime.1 <- ave(model.data$count, model.data$BEAT,
            FUN= function(x) filter(x, pastDays(1), sides= 1))
```

The ave() function takes three arguments—the numeric variable to be manipulated, the group or level at which the aggregation is required, and the function to be applied to the numeric variable. The filter function applied to the numeric variable "count" in this case takes the sum of the number of rows going back equal to the argument supplied in the pastDays() function and returns the value for each beat date combination expect for the first date for which do not have any past observations.

We can calculate the crimes for the past 7 and 30 days as well.

```
> model.data$past.crime.7 <- ave(model.data$count, model.data$BEAT,
            FUN= function(x) filter(x, pastDays(7), sides= 1))
```

```
> model.data$past.crime.30 <- ave(model.data$count, model.data$BEAT,
            FUN= function(x) filter(x, pastDays(30), sides= 1))
```

If we look at the data, we can see that the past crime information will not be available for certain dates. For example, if the first observation in our data is from May 22, 2011. This implies that the first observation for the past.crime.30 variable can only be calculated from June 21, 2011 onwards. Since we have 297 beats in our data, we will have missing values for 8910 rows, which is about 8% of our total observations. Simply removing these rows can lead to loss of valuable information. To circumvent this barrier, we need to impute for these missing observations with logical estimates from the data. One way to approach this is by taking the mean for each of these variables and replace the missing cells with this value.

By default, R will give an NA if asked to calculate the mean of a variable which has missing values. It needs to be instructed to not consider the rows with missing values while doing the calculation. meanNA() below, is a simple function to do that.

```
> meanNA <- function(x){
      mean(x, na.rm= TRUE)
}

> model.data$past.crime.1 <- ifelse(is.na(model.data$past.crime.1),
            meanNA(model.data$past.crime.1), model.data$past.crime.1)

> model.data$past.crime.7 <- ifelse(is.na(model.data$past.crime.7),
            meanNA(model.data$past.crime.7), model.data$past.crime.7)

> model.data$past.crime.30 <- ifelse(is.na(model.data$past.crime.30),
            meanNA(model.data$past.crime.30), model.data$past.crime.30)
```

We are not done with past crimes as yet. There is more valuable information that we can extract from these and, typically, such will be the case in most data mining exercises we do. With a few base variables, we can tweak, twist, and turn them to look at the response from different perspectives.

We have information on the number of arrests made in each beat on each day. We calculate similar past variables for arrests and see the effects of police actions in the past on crimes in the future. For example, we can calculate the number of arrests in the past 30 days for each beat.

```
> model.data$past.arrest.30 <- ave(model.data$ARREST, model.data$BEAT,
            FUN= function(x) filter(x, pastDays(30), sides= 1))

> model.data$past.arrest.30 <- ifelse(is.na(model.data$past.arrest.30),
            meanNA(model.data$past.arrest.30), model.data$past.arrest.30)
```

Simply using the past arrests variable will not bring out the effect we are looking for since it will be highly correlated with the past crimes variable (the correlation between past.crime.30 and past.arrest.30 is ~83%)

```
> cor(model.data$past.crime.30, model.data$past.arrest.30)
[1] 0.8330903
```

To bring out the real effect of policing, we can normalize past arrests by past crimes and see if more arrests per crime in the past have deterred criminals from committing crimes in the future.

```
> model.data$policing <- ifelse(model.data$past.crime.30 == 0, 0,
            model.data$past.arrest.30/model.data$past.crime.30)
```

Another useful extension of past crimes variables can be the interaction between crimes in the past 7 days to crime in the past 30 days to see if crime has been increasing or decreasing in the recent past and what effect does this have on future crimes.

```
> model.data$crime.trend <- ifelse(model.data$past.crime.30 == 0, 0,
            model.data$past.crime.7/model.data$past.crime.30)
```

Our visualization exercise showed that there is considerable variation in crime incidents with respect to the time of the year. But we have only 1 year of data for both building the model and validating it. To use the month variable effectively, we can combine the similar months to form a new variable that reflects the changes in crime incidents with seasons.

```
> model.data$season <- as.factor(ifelse(model.data$month %in% c("Mar", "Apr",
            "May"), "spring",
            ifelse(model.data$month %in% c("Jun", "Jul",
            "Aug"), "summer",
            ifelse(model.data$month %in% c("Sep", "Oct",
            "Nov"), "fall", "winter"))))
```

We have created a number of predictor variables, including interactions. The motivation behind deriving these was to get a set of variables that explains a good amount of variation in the dependent variable. One possible way to check this is the correlation between the dependent variable and the independent variables. The psych (Revelle, 2012) library has a convenient function that will allow us to do just that (Figure 13.11).[15]

```
> library(psych)

> model.cor <- cor(model.data[, c('count', 'past.crime.1', 'past.crime.7',
            'past.crime.30','policing', 'crime.trend')])

> cor.plot(model.cor)
```

Though data mining is a creative process, it does have some rules. One of the most stringent ones being that the performance of any predictive model is to be tested on a separate out-of-sample dataset. Analysts and scientists who are ignorant of this are bound to face the wrath of over-fitting leading to poor results when the model is deployed.

[15] In case of an error message regarding the margins being too small for the plot, try adjusting the margins using

```
> op <- par(mar= rep(0,4))
> cor.plot(model.cor)
> par(op)
> dev.off()
```

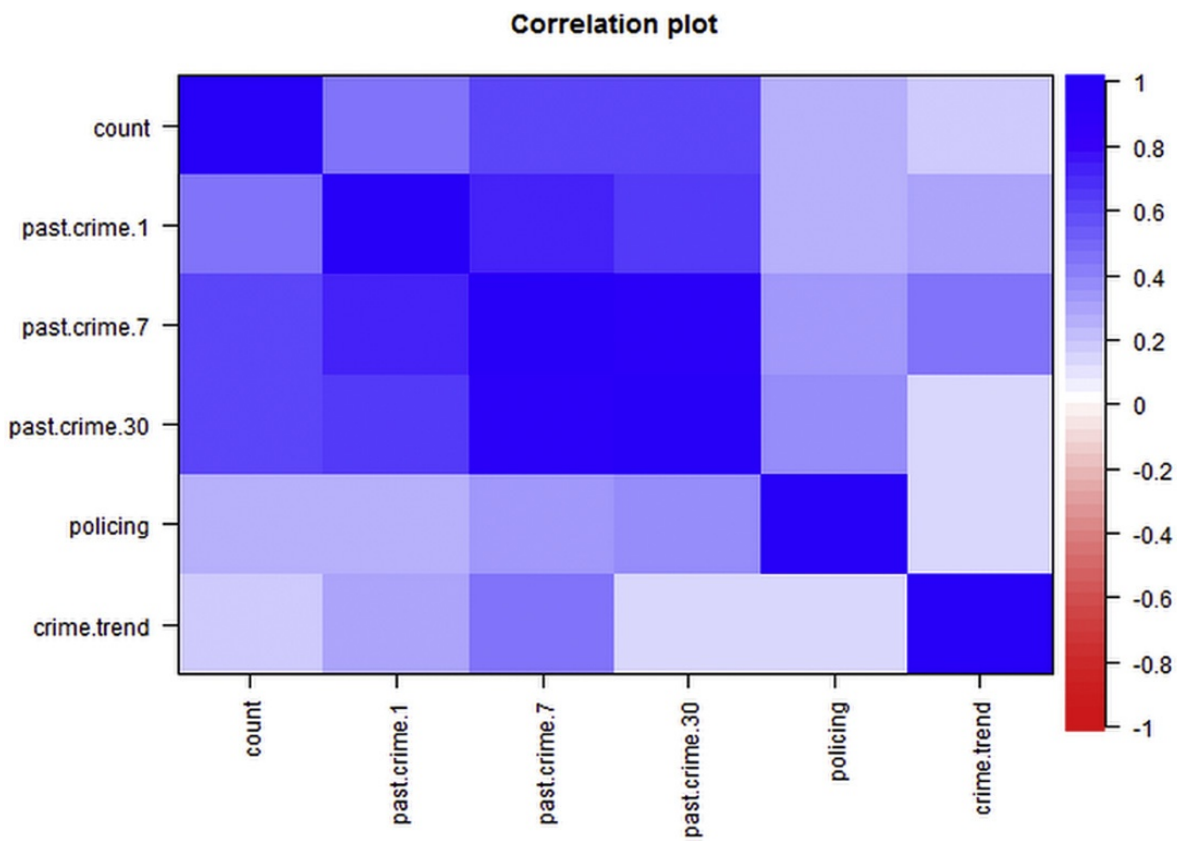

Figure 13.11
Correlation matrix plot.

In our case, to measure the performance of our model, we will use an out-of-time validation sample. We will divide our data into two portions—a 90% development sample where we can train or develop our model and a 10% validation sample where we can test the performance of our model. We can do this by ordering the data by date and then splitting them into train and test using proportional numbers for observations required in each.

```
> model.data <- orderBy(~ date, data= model.data)

> rows <- c(1:floor(nrow(model.data)*0.9))

> test.data <- model.data[-rows, ]

> model.data <- model.data[rows, ]
```

The dependent variable here is a count variable. It is a discrete variable that counts the number of crimes instances in a particular beat on a given day. Typically, for modeling counts, a poisson regression model is a preferred choice that seems to fit the data well. However, the poisson regression model comes with the strong assumption that the mean of the dependent variable is equal to its variance. In our case, this assumption does not hold.

```
> mean(model.data$count)
[1] 3.178985

> var(model.data$count)
[1] 6.028776
```

The variance is much greater than the mean indicating that the distribution is overdispersed. A suitable way to model for such overdispersion is using the negative binomial distribution. We will use the glm.nb() function in the MASS (Venables and Ripley, 2002) package to build the model.

As a basic model, we will include all linear variables and interactive effects we created in the previous section.

```
> crime.model <- glm.nb(count ~ past.crime.1 + past.crime.7 + past.crime.30 +
             + policing + crime.trend + factor(day) + season, data= model.data)

> summary(crime.model)

Call:
glm.nb(formula = count ~ past.crime.1 + past.crime.7 + past.crime.30 +
    policing + crime.trend + day + season, data = model.data,
    init.theta = 11.62188729, link = log)

Deviance Residuals:
    Min 1Q Median 3Q Max
-3.8253 -0.8946 -0.1503 0.5434 5.4160

Coefficients:
                 Estimate   Std. Error  z  value   Pr(>|z|)
(Intercept)     -0.1747152   0.0121101  -14.427    < 2e-16  ***
past.crime.1     0.0121566   0.0010234   11.879    < 2e-16  ***
past.crime.7     0.0009976   0.0005072    1.967   0.049199  *
past.crime.30    0.0086720   0.0001221   71.042     < 2e-16 ***
policing         0.2290357   0.0195652   11.706     < 2e-16 ***
crime.trend      1.3358112   0.0374330   35.685     < 2e-16 ***
dayMon          -0.0568357   0.0073488   -7.734   1.04e-14  ***
daySat          -0.0500302   0.0073600   -6.798   1.06e-11  ***
daySun          -0.1028851   0.0074047  -13.894     < 2e-16 ***
dayThu          -0.0441040   0.0073510   -6.000   1.98e-09  ***
dayTue          -0.0244230   0.0073278   -3.333   0.000859  ***
dayWed          -0.0227957   0.0073220   -3.113   0.001850  **
seasonspring     0.0313964   0.0057203    5.489   4.05e-08  ***
seasonsummer     0.0428420   0.0054100    7.919   2.39e-15  ***
seasonwinter    -0.0059497   0.0057152   -1.041   0.297866
–
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for Negative Binomial(11.6219) family taken to be 1)

    Null deviance: 174906 on 106622 degrees of freedom
Residual deviance: 119530 on 106608 degrees of freedom
AIC: 422674
```

```
Number of Fisher Scoring iterations: 1

              Theta:  11.622
          Std. Err.:  0.235

2 x log-likelihood: -422642.430
```

Most of the variables are significant at the 0.1% level except crimes in the past 7 days given by past.crime.7 and the winter season. This shows that the visualization exercise was fruitful and our insights based on those did represent the structure present in the data. The significance of variables, however, is not a deciding factor for the model being good or bad. For that, we need to go a step further.

13.7 Model Evaluation

One widely used industry practice to decide whether a model is good or bad is to check its performance on an out-of-sample or out-of-time data set. Before starting the modeling exercise, we kept a portion of data for this purpose. We can check the performance on the out-of-sample data set by scoring it using the predict() function and comparing it to the actual values by using the root mean squared error (RMSE).

```
> crime.model.pred <- predict(crime.model, test.data, type= "response")

> sqrt(mean((test.data$count - crime.model.pred)^2))
[1] 1.950504
```

The RMSE is a popular industry metric that gives us a value of how far we are on average from the actual values. We establish a benchmark with the first iteration of the model. This benchmark then initiates an iterative process of tweaking the model constituents, adding or deleting variables, transforming them till we get some improvement. As an example, consider the scenario where we believe that crimes in the future increase as crimes in the past increase, but at a decreasing rate, that is, the relationship between then is concave. In such a situation including another variable which is a higher-order term for past.crime.30 might help reduce the RMSE.[16,17]

```
> crime.model <- glm.nb(count ~ past.crime.1 + past.crime.7 + past.crime.30 +
             + crime.trend + policing + factor(day) + season + I(past.crime.30^2),
             data= model.data)

> summary(crime.model)

> crime.model.pred <- predict(crime.model, test.data, type= "response")

> sqrt(mean((test.data$count - crime.model.pred)^2))
[1] 1.916148
```

[16] In R, to include a functional form in a glm(), we do not have to create a separate variable in the data set that is the transformation of the original variable. We can simply indicate it directly in the model statement as shown in the example.

[17] The output of the model has been suppressed due to space constraints.

Inclusion of an additional important variable dropped the RMSE from 1.95 to 1.92. Though not a very big jump, it still is an indication that we are headed in the right direction.

Typically, with limited amount of data and time one can only improve so much on a given model. The key is then to decide the optimal point where the trade-off between accuracy and effort is minimized.

RMSE is just one of the many metrics that can be employed to determine the predictive power of a model. Another common method used is the actual versus predicted plot that gives a visual representation of how far away our predictions are from the actual values.

Directly comparing the actual and predicted values may not yield any insights considering there are about ~10,000 data points within a small range. To make the comparison visually convenient and conclusive, we can bin the predictions and the actuals into groups and compare the average values for corresponding groups. For example, we can create 10 groups of the predicted values and take the average of the predicted and actual values in each group. The idea is that if the predictions were reasonably accurate, we would get reasonably overlapping lines. This gives us a comprehensive and condensed view of the differences between our predictions and the actual crimes.

For the plot, we first bring the actual and the predicted values into one data frame and then use the cut() function to create 10 groups of the predicted values and calculate the average values of the predicted and actual crimes by applying the mean() function to each group (Figure 13.12).

```
> validate <- data.frame(test.data$count, crime.model.pred)

> names(validate) <- c("actual", "predicted")

> validate$bucket <- with(validate, cut(predicted, breaks=
            quantile(predicted, probs= seq(0, 1, 0.1)),
            include.lowest= TRUE, labels= c(1:10)))

> validate <- aggregate(validate[, c('actual', 'predicted')], by=
            list(validate$bucket), FUN = mean)
```

For comparison, we plot the actual and predicted means on the same graph.

```
> plot(validate$predicted, col= "red", type= "l", lwd= 1.5, ylab= "No. of Crashes", xlab=
"Predicted Crimes Decile", main= "Actual vs. Predicted")

> lines(validate$actual, col= "blue", lwd= 1.5)

> legend("topright", c("Actual", "Predicted"), col= c("blue", "red"), lwd= c(1.5, 1.5),
bty= "n")
```

The graph shows that the actual and predicted values come quite close in the middle and diverge in the extremes. We are, on average, under predicting in the first few deciles (except the first decile) and over predicting in the last few ones.

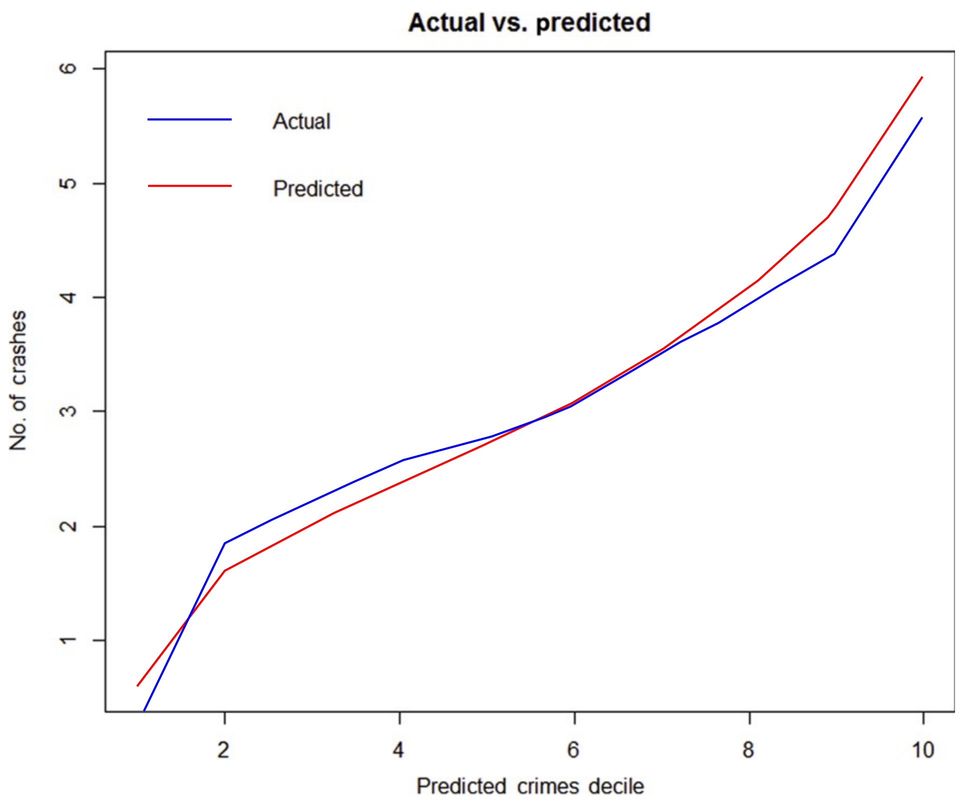

Figure 13.12
Actual versus predicted.

13.8 Discussions and Improvements

It has been a lengthy and informative exercise till now. We learnt how to—pull data directly from web, handle, clean, and process crime data, utilize geo-coded information through shape files, retrieve hidden information through visualizations, create new variables from limited data, build a predictive engine for crime, and test how good it is. There are still some issues that we need to address related to deployment, limitation, and improvements.

The model we built predicts the expected number of crimes in each beat in the city of Chicago for each day. The purpose behind building the model was to build a resource that would help law enforcement agencies deploy their resources proactively and efficiently. In purview of this, one way the model could be deployed is by using color-coded maps identifying the crime hot-spots in the city for a particular day. The crime hot-spots can then be administered effectively by using patrolling teams.

These strategies, however, do have their limitations. First, the 24-h prediction window might appear too large and static for effective patrolling. For example, we saw that crimes follow a pattern during a particular day. Their frequency tends to be much higher during the latter half of the day. A spatial dimension can be added to this pattern, that is, there might be certain beats which expect more crimes during a particular time interval within a day. A possible workaround could be to build the model for smaller time intervals and allow the crime hot-spots to change during the day. Second, our crime model predicts the expected number of crimes without being able to differentiate among them and by treating them equally. In reality, certain crimes types have totally different characteristics from the others. For example, crimes like murder, aggravated assault, and rape, along with other violent crimes, may need special attention from modelers and the police alike. These nuances could be better addressed if there were separate models for violent crimes, nonviolent and organized crimes. Third, zoning down to one beat as a crime hotspot ignores the impact of crimes in neighboring areas. Since crimes have a spatial distribution attached to them, there might be high correlation between criminal activities in adjoining beats. A simple way to control for this is to include crime history of the adjoining beats in the given model and check their impact on the predictions.

The limitations discussed earlier indicate that there is a lot of scope for further research and work in the area of crime prediction. And the scope is not limited to the data itself. There are different predictive techniques that can be employed to see if we can get incremental lift from these. The suggestions, however, do not undermine the work presented in the chapter which is expected to provide the reader with a detailed understanding of processes involved in mining crime data with R.

References

Højsgaard, S., Ulrich, H., 2012. doBy: doBy—groupwise summary statistics, general linear contrasts, population means (least-squares-means), and other utilities.

James, D., Hornik, K., 2011. chron: chronological objects which can handle dates and times.

Lewin-Koh, N.J., Bivand, R., 2012. maptools: tools for reading and handling spatial objects.

Revelle, W., 2012. Procedures for psychological, psychometric, and personality research.

Venables, W.N., Ripley, B.D., 2002. Modern Applied Statistics with S. Springer, New York.

Wickham, H., 2009. ggplot2: elegant graphics for data analysis. Springer, New York.

Wickham, H., 2011. The split-apply-combine strategy for data analysis. Journal of Statistical Software. 40, 1–29.

Xie, Y., 2012. animation: a gallery of animations in statistics and utilities to create.