Crime Analysis in Chennai

Dr. Jeffrey Strickland

8/28/2018

Using R for Crime Analysis

"Sherlock Holmes was a man, however, who, when he had an unsolved problem upon his mind, would go for days, and even for a week, without rest, turning it over, rearranging his facts, looking at it from every point of view until he had either fathomed it or convinced himself that his data were insufficient."

—Dr. John Watson, The Man with the Twisted Lip

Introduction

Sherlock's data may have been insufficient, but at present we cannot make the same complaint. If we have any burden whatsoever, then it is possessing too much data. In this chapter, will have two objectives: (1) visualizing spatial and temporal trends of criminal activity based on data, (2) analyzing factors that may affect unlawful behavior based upon the data. Taken together, we are really using R programming to cluster and describe data that the police departments and/or city halls have already collected and turned over to the skillful hands of the analyst.

Install R libraries

```
if(!require(readr)) install.packages("readr")

if(!require(dplyr)) install.packages("dplyr")

if(!require(DT)) install.packages("DT")

if(!require(ggrepel)) install.packages("ggrepel")

if(!require(leaflet)) install.packages("leaflet")
```

the Crime Data

The crime data was simulated using distributions derived fro an analysis of major crimes in Atlanta, Georgia USA, San Fancisco, California USA, and Boston, Massatuchetts USA. For instance Larceny/theft was approximately uniformly distributed Uniform~[120000, 160000] with mean 140000 and standard deviation of approximately 11547.

The distributions were then applied to Chennai India using the inverse transform method. To simulate larceny/theft in Chennai, we used the reverse inverse method, which for the Uniform distribution is X = a + (b - a)U, where U is the Uniform[0,1] random number. In Excel, this is the RAND() function, so the inverse transform would be X = a + (b - a) *RAND(). The crimes are then geographically distributed in zones in a similar fashion.

The data was simulated due to the constraint that crime data was only available at the state-level in Inida. The data include the following factors:

- IncidntNum (N) Incident number
- Category (C) Crime category, i.e., larceny/theft
- Descript (C)
- DayOfWeek (C)
- Date (D) Date: DD/MM/YYYY
- Time (T) Time: 24-hour system
- PdDistrict (C) Police district where incident occured
- Resolution (C) Resolution of the crime
- X (N) Longitude
- Y (N) Latitude
- Location (C) Lat/long
- PdId (N) Police Department ID

(N = Numeric, T = Time, D = Date, C = Class)

Read the data

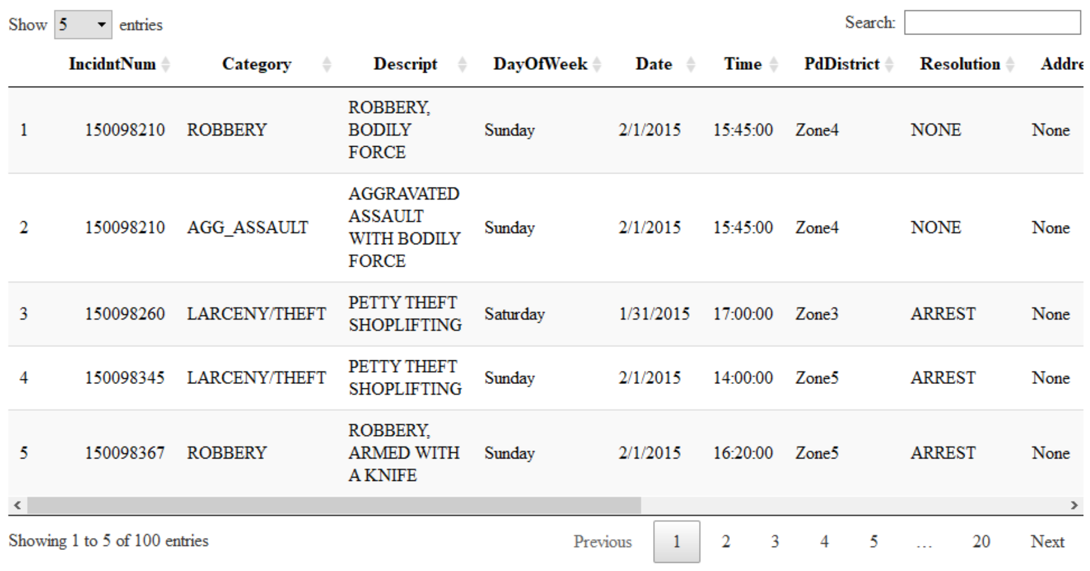
First, we load the data using readr and read_csv().

```
library(readr)
path <-
“https://github.com/stricje1/VIT_University/blob/master/Data_Analytics_2018/d
ata/chennai_crimes.zip
df <- read_csv(path)

## Parsed with column specification:
## cols(
##       IncidntNum = col_integer(),
##       Category = col_character(),
##       Descript = col_character(),
##       DayOfWeek = col_character(),
##       Date = col_character(),
##       Time = col_time(format = ""),
##       PdDistrict = col_character(),
##       Resolution = col_character(),
##       Address = col_character(),
##       X = col_double(),
##       Y = col_double(),
##       Location = col_character(),
##       PdId = col_double()
## )
```

Display Data

Now, we display the data using DT and datatable(). The table below shows a partial view of the Chennai crime data

```r
library(DT)
df_sub <- df[1:100,]  # display the first 100 rows
df_sub$Time <- as.character(df_sub$Time)
datatable(df_sub, options = list(pageLength = 5,scrollX='400px'))
```



Show 5 entries Search:

	IncidntNum	Category	Descript	DayOfWeek	Date	Time	PdDistrict	Resolution	Addre
1	150098210	ROBBERY	ROBBERY, BODILY FORCE	Sunday	2/1/2015	15:45:00	Zone4	NONE	None
2	150098210	AGG_ASSAULT	AGGRAVATED ASSAULT WITH BODILY FORCE	Sunday	2/1/2015	15:45:00	Zone4	NONE	None
3	150098260	LARCENY/THEFT	PETTY THEFT SHOPLIFTING	Saturday	1/31/2015	17:00:00	Zone3	ARREST	None
4	150098345	LARCENY/THEFT	PETTY THEFT SHOPLIFTING	Sunday	2/1/2015	14:00:00	Zone5	ARREST	None
5	150098367	ROBBERY	ROBBERY, ARMED WITH A KNIFE	Sunday	2/1/2015	16:20:00	Zone5	ARREST	None

Showing 1 to 5 of 100 entries

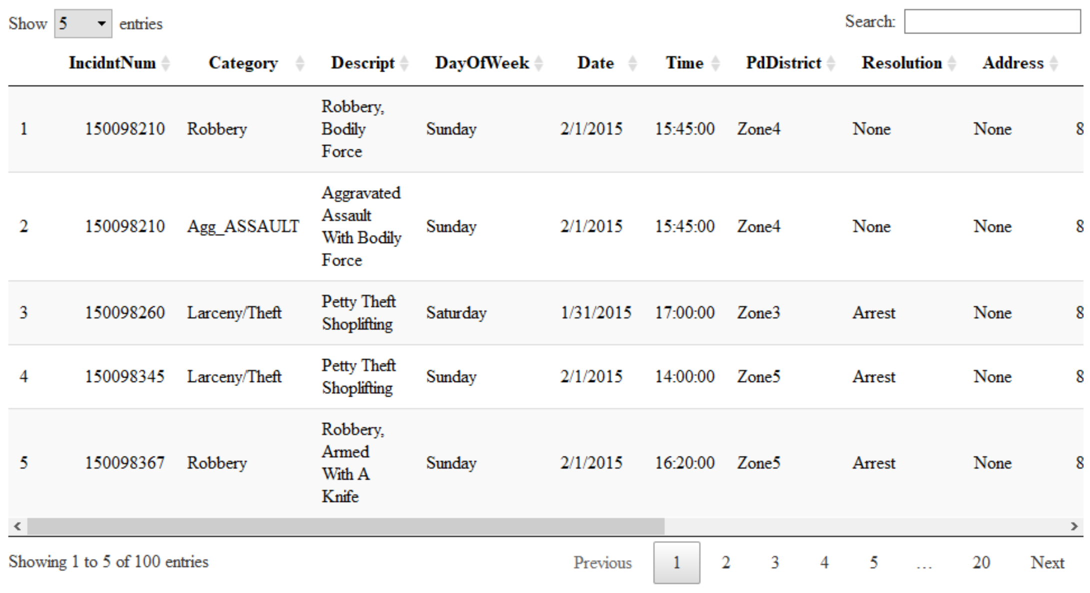
Previous 1 2 3 4 5 … 20 Next

Preprocess the Data

The All-Caps text is difficult to read. Let's force the text in the appropriate columns into proper case.

```r
proper_case <- function(x) {
  return (gsub("\\b([A-Z])([A-Z]+)", "\\U\\1\\L\\2" , x, perl=TRUE))
}

library(dplyr)
df <- df %>% mutate(Category = proper_case(Category),
                    Descript = proper_case(Descript),
                    PdDistrict = proper_case(PdDistrict),
                    Resolution = proper_case(Resolution),
                    Time = as.character(Time))
```

```r
df_sub <- df[1:100,]  # display the first 100 rows
datatable(df_sub, options = list(pageLength = 5,scrollX='400px'))
```



Show 5 entries Search:

	IncidntNum	Category	Descript	DayOfWeek	Date	Time	PdDistrict	Resolution	Address	
1	150098210	Robbery	Robbery, Bodily Force	Sunday	2/1/2015	15:45:00	Zone4	None	None	8
2	150098210	Agg_ASSAULT	Aggravated Assault With Bodily Force	Sunday	2/1/2015	15:45:00	Zone4	None	None	8
3	150098260	Larceny/Theft	Petty Theft Shoplifting	Saturday	1/31/2015	17:00:00	Zone3	Arrest	None	8
4	150098345	Larceny/Theft	Petty Theft Shoplifting	Sunday	2/1/2015	14:00:00	Zone5	Arrest	None	8
5	150098367	Robbery	Robbery, Armed With A Knife	Sunday	2/1/2015	16:20:00	Zone5	Arrest	None	8

Showing 1 to 5 of 100 entries Previous 1 2 3 4 5 … 20 Next

Explore the Data

Crime across space

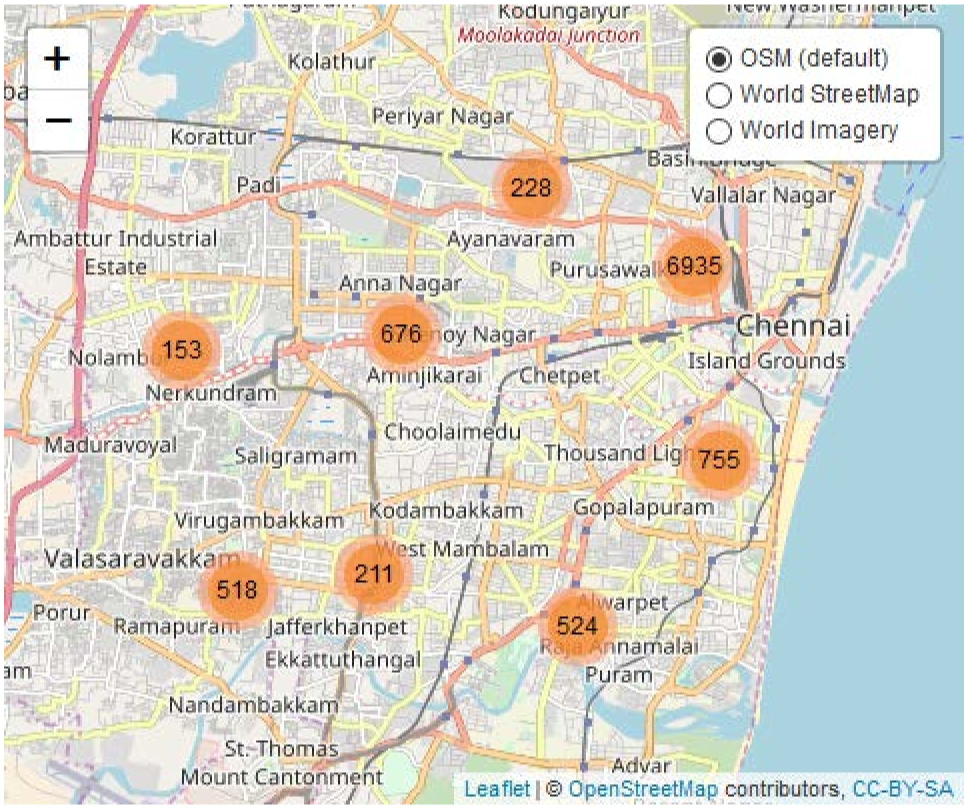
In this section, we use the leaflet function. It creates a Leaflet map widget using htmlwidgets. The widget can be rendered on HTML pages generated from R Markdown. In addition to matrices and data frames, leaflet supports spatial objects from the sp package and spatial data frames from the sf package. We create a Leaflet map with these basic steps: First, create a map widget by calling leaflet(). Next, we add layers (i.e., features) to the map by using layer functions (e.g. addTiles, addMarkers, addPolygons) to modify the map widget. Then you keep adding layers or stop when satisfied with the result. We will add a tile layer from a known map provider, using the leaflet function addProviderTiles. A list of providers can be found at http://leaflet-extras.github.io/leaflet-providers/preview/. We will also add graphics elements and layers to the map widget with addCondtroll (addTiles). We use markers to call out points on the map. Marker locations are expressed in latitude/longitude coordinates, and can either appear as icons or as circles. When there are many markers on a map as in our case with crimes, we can cluster them together. library(leaflet)

The following code defines the popups used on the Leaflet map we will create. The popups will appear on the interactive Leaflet map. When you click on a popup, it will contain the information defined below:

```
data <- df[1:10000,] # display the first 10,000 rows
data$popup <- paste("<b>Incident #: </b>", data$IncidntNum,
                    "<br>", "<b>Category: </b>", data$Category,
                    "<br>", "<b>Description: </b>", data$Descript,
                    "<br>", "<b>Day of week: </b>", data$DayOfWeek,
                    "<br>", "<b>Date: </b>", data$Date,
                    "<br>", "<b>Time: </b>", data$Time,
                    "<br>", "<b>PD district: </b>", data$PdDistrict,
                    "<br>", "<b>Resolution: </b>", data$Resolution,
                    "<br>", "<b>Address: </b>", data$Address,
                    "<br>", "<b>Longitude: </b>", data$X,
                    "<br>", "<b>Latitude: </b>", data$Y)
```

Now, we define crime incident locations on the map using leaflet, which we use for our popups:

```
library(leaflet)
leaflet(data, width = "100%") %>% addTiles() %>%
  addTiles(group = "OSM (default)") %>%
  addProviderTiles(provider = "Esri.WorldStreetMap",group = "World
StreetMap") %>%
  addProviderTiles(provider = "Esri.WorldImagery",group = "World Imagery")
%>%
  addMarkers(lng = ~X, lat = ~Y, popup = data$popup, clusterOptions =
markerClusterOptions()) %>%
  addLayersControl(
    baseGroups = c("OSM (default)","World StreetMap", "World Imagery"),
    options = layersControlOptions(collapsed = FALSE)
  )
```



In this manner, we can click icons on the map to show incident details. We need to set up some generate some parameters that we concatenate or "paste" together to form these incident descriptions. For example, the concatenated strings pdata$popup, provides the content of the second incident as shown here:

```
data$popup[1]

## [1] "<b>Incident #: </b> 150098210 <br> <b>Category: </b> Robbery <br>
<b>Description: </b> Robbery, Bodily Force <br> <b>Day of week: </b> Sunday
<br> <b>Date: </b> 2/1/2015 <br> <b>Time: </b> 15:45:00 <br> <b>PD district:
</b> Zone4 <br> <b>Resolution: </b> None <br> <b>Address: </b> None <br>
<b>Longitude: </b> 80.26669397 <br> <b>Latitude: </b> 13.09199072"
```

You may notice the "%>%" or forward-pipe operator in the leaflet arguments. The operators pipe their left-hand side values forward into expressions that appear on the right-hand side, rather than from the inside and out. For example:

```
leaflet(data, width = "100%") %>% addTiles() %>%
  addTiles(group = "OSM (default)") %>% ...
```

Crime Over Time

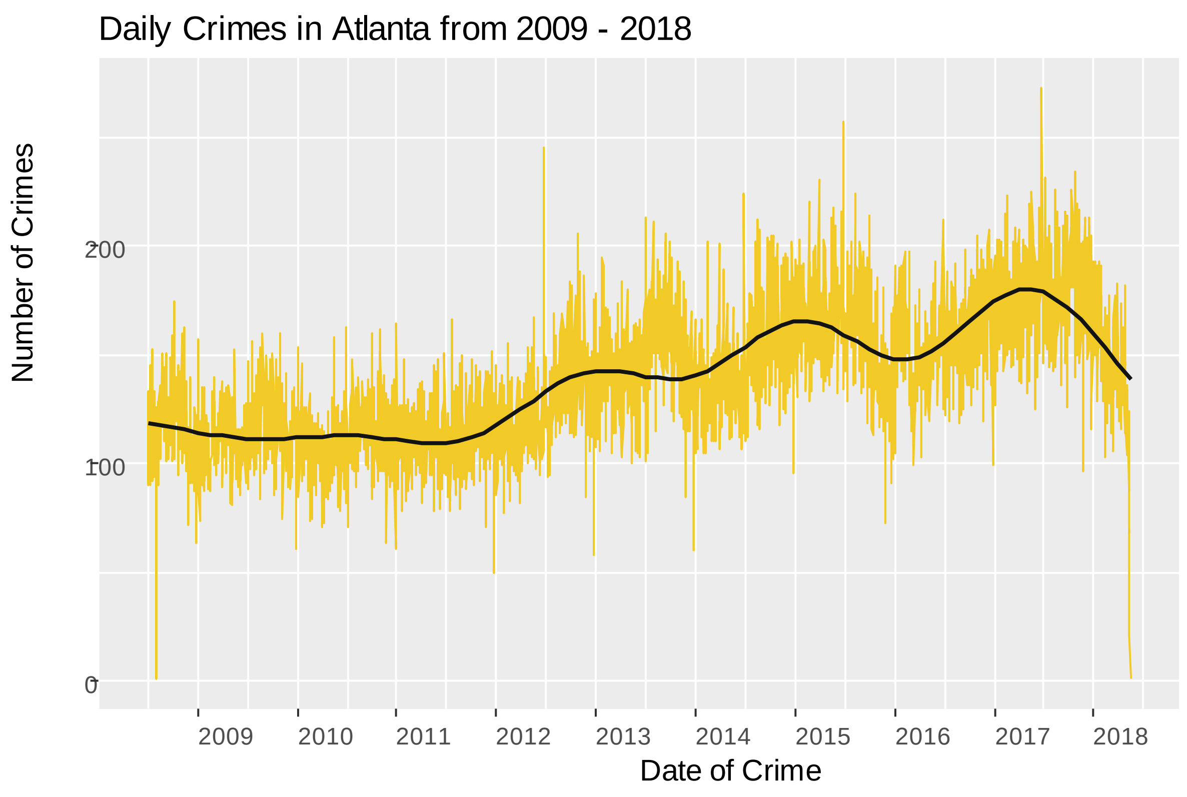
In this section, we will manipulate the data using the dplyr::mutate function. mutate adds new variables while preserving extisting variables. Below, we used "shades of bue" in the code for our plot, with a dark blue line that smooths the data.

```
library(dplyr)
df_crime_daily <- df %>%
      mutate(Date = as.Date(Date, "%m/%d/%Y")) %>%
      group_by(Date) %>%
      summarize(count = n()) %>%
      arrange(Date)
```

Crimes Series Plot

A crimes series is a time series where the events are crimes. These have the usual components of a time series like seasonality, trend, and noise. However, the seasonality may follow a pattern from day to night, where crimes may increase at night and fall off during the day. Another scenario might involve increased crime rate during certain events, like parades, rodeos, fairs, and so on. The 1996 Summer Olympics (not in the dataset) brought an increase in crime to Atlanta, including the Centennial Olympic Park bombing on July 27, attributed to domestic terrorism.

```
library(ggplot2)
library(scales)

##
## Attaching package: 'scales'

## The following object is masked from 'package:readr':
##
##     col_factor

plot <- ggplot(df_crime_daily, aes(x = Date, y = count)) +
      geom_line(color = "#F2CA27", size = 0.1) +
      geom_smooth(color = "#1A1A1A") +
      scale_x_date(breaks = date_breaks("1 year"), labels =
date_format("%Y")) +
      labs(x = "Date of Crime", y = "Number of Crimes",
           title = "Daily Crimes in Chennai from 2009 - 2018")
plot

## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

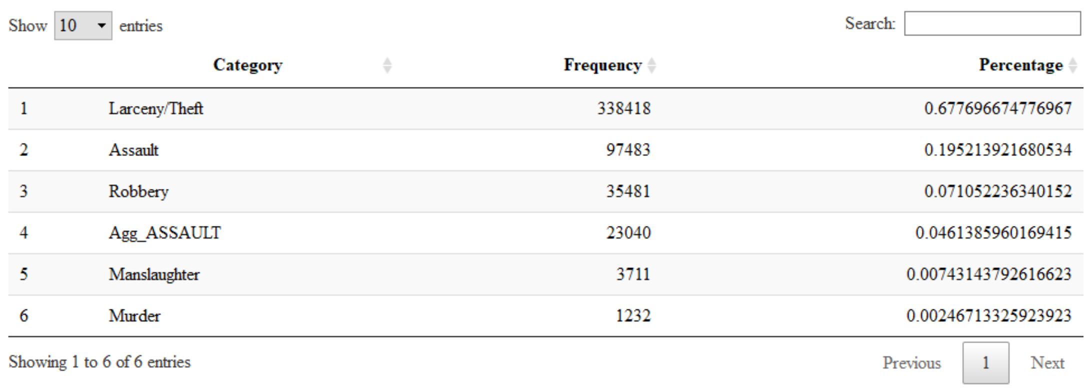


The trend shown by Figure 4-3 shows a slight decrease in reported crimes up to the sharp rise beginning in 2017. Late summer of 2018 shows a possible decrease. The crime series also shows consistent seasonality, at least up to the sharp increase where it is difficult to observe.

Aggregated Data

No, we can aggregate the data and create a table that summarizes the data by incident category. (In Rstudio, this table shows in the Viewer pane.) We used the descending order of "decreasing" for sorting the incident category. DT::datatable or the datatable function generates the HTML table widget.

```r
df_category <- sort(table(df$Category),decreasing = TRUE)
df_category <- data.frame(df_category[df_category > 1000])
colnames(df_category) <- c("Category", "Frequency")
df_category$Percentage <- df_category$Frequency / sum(df_category$Frequency)
datatable(df_category, options = list(scrollX='400px'))
```



Show 10 entries Search:

	Category	Frequency	Percentage
1	Larceny/Theft	338418	0.677696674776967
2	Assault	97483	0.195213921680534
3	Robbery	35481	0.071052236340152
4	Agg_ASSAULT	23040	0.0461385960169415
5	Manslaughter	3711	0.00743143792616623
6	Murder	1232	0.00246713325923923

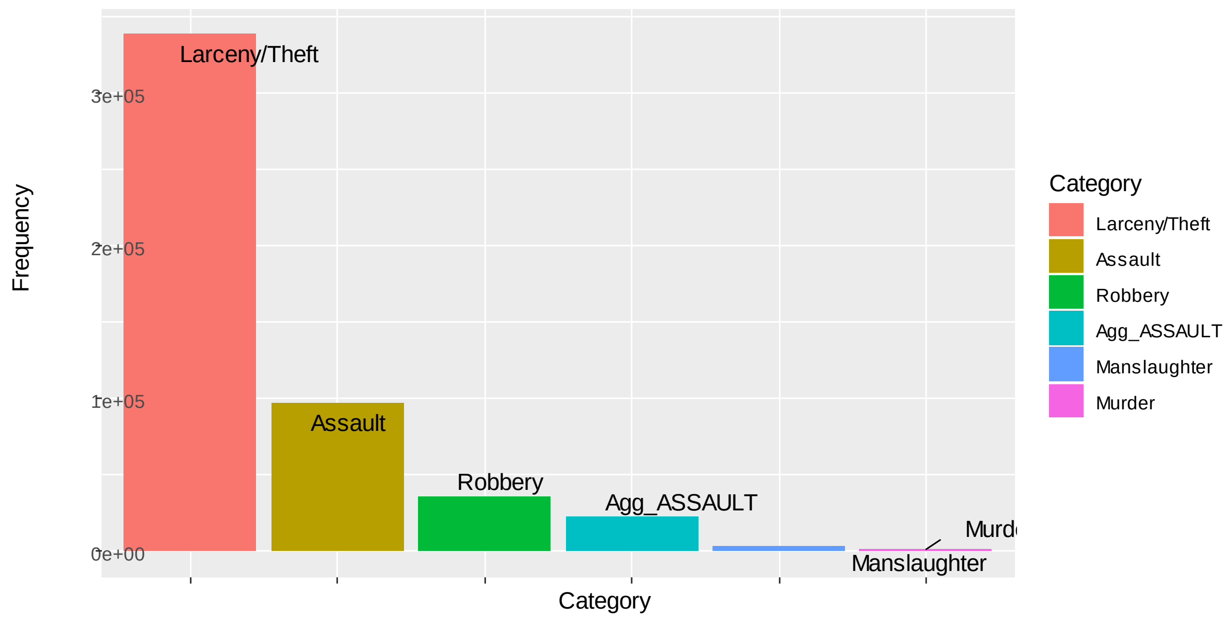
Showing 1 to 6 of 6 entries Previous 1 Next

The table shows the frequencies (counts) of the reported crimes with 10,000 or more occurrences, indicating that the most frequent crime is Larceny/Theft (non-vehicular and from the vehicle). With all the hype regarding the dangers of Chennai, there were only 886 murders in the 10 year period, or about 88 per year, and only 121 reported rapes for about 12 per year or 1 per month.

Create a Bar Chart

Now that we can aggregate the data, we will show the data with a bar graph, depicted in Figure 4-4. The bar graph (or histogram) shows the frequencies of the crimes recorded in Table 4-3. It makes it easy to see the vast difference between Larceny/Theft and the other reported crimes.

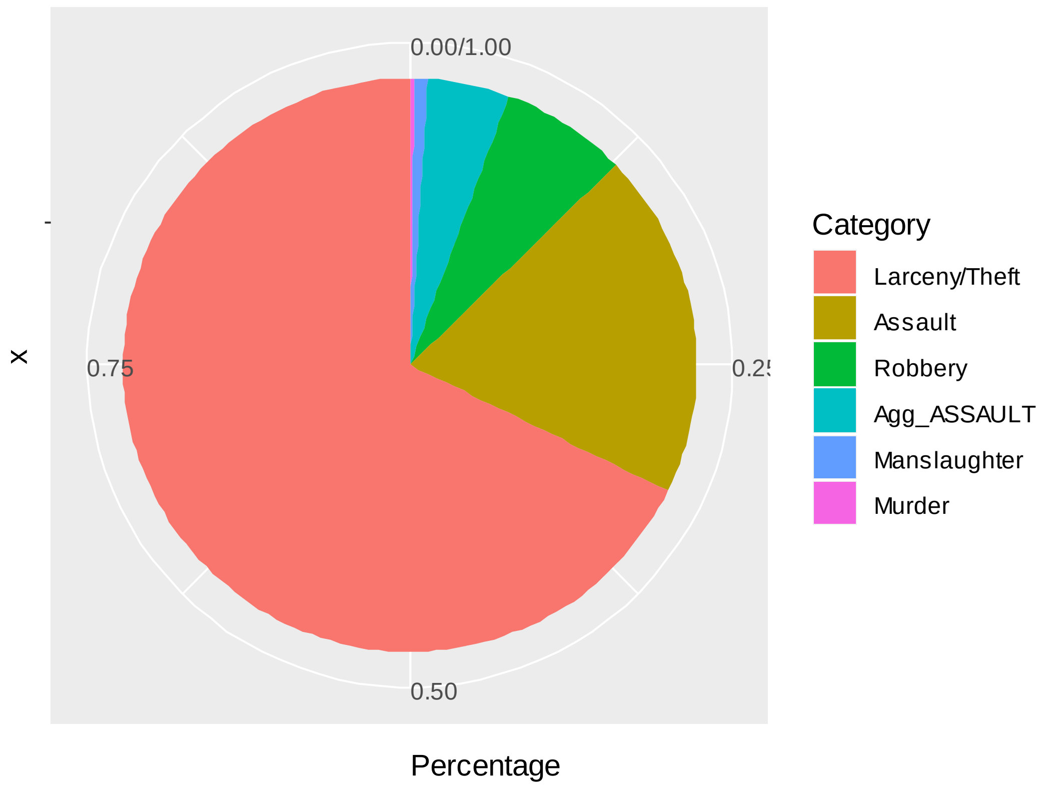
```
library(ggplot2)
library(ggrepel)
bp <- ggplot(df_category, aes(x=Category, y=Frequency, fill=Category)) +
  geom_bar(stat="identity") +
  theme(axis.text.x=element_blank()) +
  geom_text_repel(data=df_category, aes(label=Category))
bp
```



Create a pie chart

To further illustrate the crime incident data, a subsequent pie chart is plotted in Figure 4-5. The chart illustrates the same data as does the bar chart, but it may be more understandable in this instance. It shows that Larceny/Theft occurs more than twice as much as all other reported crimes taken together. The chart is also ascetically pleasing.

```
bp<-ggplot(df_category, aes(x="", y=Percentage, fill=Category)) +
  geom_bar(stat="identity")
pie <- bp + coord_polar("y")
pie
```

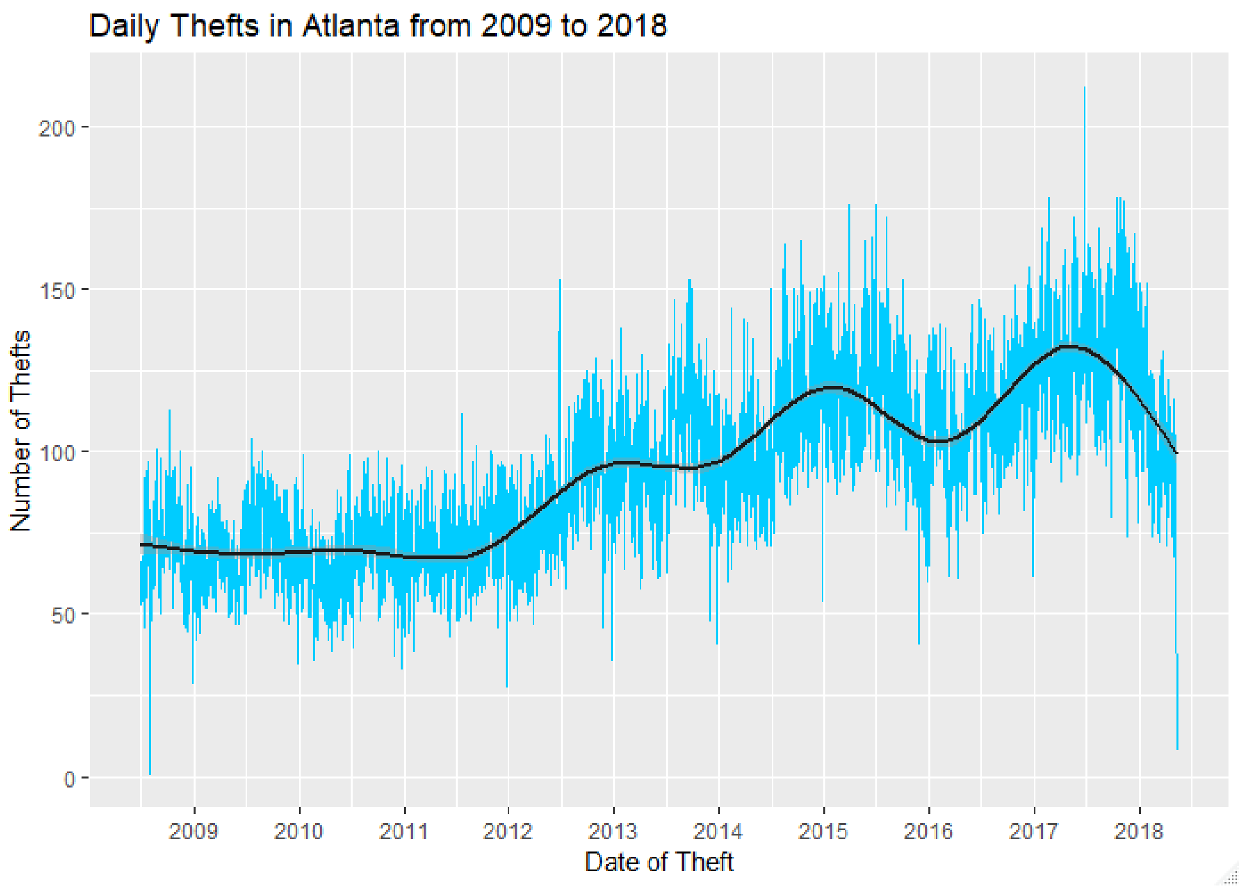


Temporal Trends

Theft Over Time

In this section, we create a chart of crimes (Larceny/Theft) over time. And for aesthetic effect as well as clarity, we make use color in depicting the series shown in Figure 4-6. This will provide us with a crime series for thefts, which we will smooth as we did before.

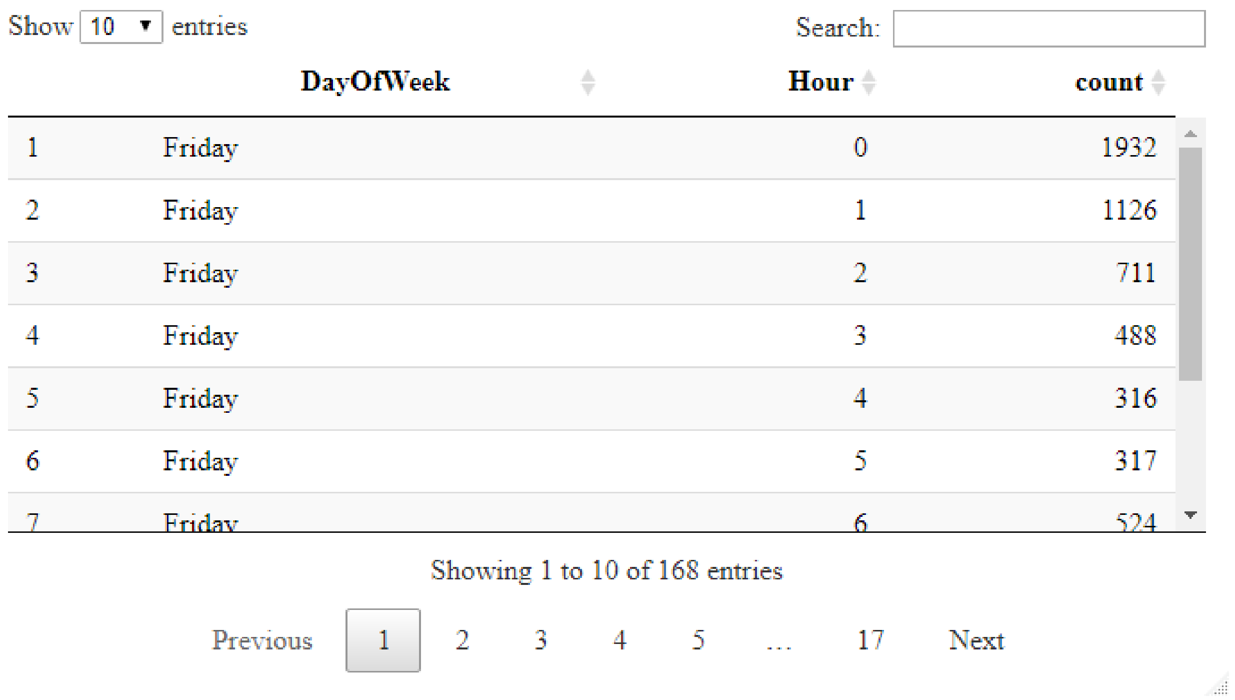
```
path <- "c:\\Users\\jeff\\Documents\\Crime
Analysis\\India_data\\chennai_crimes.csv"
df <- read_csv(path)

## Parsed with column specification:
## cols(
##   IncidntNum = col_integer(),
##   Category = col_character(),
##   Descript = col_character(),
##   DayOfWeek = col_character(),
##   Date = col_character(),
##   Time = col_time(format = ""),
##   PdDistrict = col_character(),
##   Resolution = col_character(),
##   Address = col_character(),
##   X = col_double(),
##   Y = col_double(),
##   Location = col_character(),
##   PdId = col_double()
## )

head(df)

## # A tibble: 6 x 13
##   IncidntNum Category Descript DayOfWeek Date  Time  PdDistrict Resolution
##        <int> <chr>    <chr>    <chr>     <chr> <tim> <chr>      <chr>
## 1  150098210 ROBBERY  ROBBERY~ Sunday    2/1/~ 15:45 Zone4      NONE
## 2  150098210 AGG_ASS~ AGGRAVA~ Sunday    2/1/~ 15:45 Zone4      NONE
## 3  150098260 LARCENY~ PETTY T~ Saturday  1/31~ 17:00 Zone3      ARREST
## 4  150098345 LARCENY~ PETTY T~ Sunday    2/1/~ 14:00 Zone5      ARREST
## 5  150098367 ROBBERY  ROBBERY~ Sunday    2/1/~ 16:20 Zone5      ARREST
## 6  150098395 LARCENY~ PETTY T~ Sunday    2/1/~ 14:30 Zone7      ARREST
## # ... with 5 more variables: Address <chr>, X <dbl>, Y <dbl>,
## #   Location <chr>, PdId <dbl>

df_theft <- df %>% filter(grepl("Larceny/Theft", Category))

df_theft_daily <- df_theft %>%
      mutate(Date = as.Date(Date, "%m/%d/%Y")) %>%
      group_by(Date) %>%
      summarize(count = n()) %>%
      arrange(Date)

library(ggplot2)
library(scales)
plot <- ggplot(df_theft_daily, aes(x = Date, y = count)) +
      geom_line(color = "#00ccff", size = 0.1) +
      geom_smooth(color = "#1A1A1A") +
      # fte_theme() +
      scale_x_date(breaks = date_breaks("1 year"), labels =
date_format("%Y")) +
      labs(x = "Date of Theft", y = "Number of Thefts", title = "Daily Thefts
in Chennai from 2009 to 2018")
plot
```



Theft Time Heatmap

Now, we aggregate counts of thefts by Day-of-Week and Time to create heat map. Fortunately, the Day-Of-Week part is pre-derived, but Hour is slightly harder. We need a function that gets the hour from the time string in atlanta_crime_10yr.csv, so that we can use an approximate arrest time with day of the week. But R does not have one, or one I can find. So, we build the function below, using the colon delimiter to separate hours from minutes. Then we build a table that allows us to check that the code is doing what we expected.

```
get_hour <- function(x) {
      return (as.numeric(strsplit(x,":")[[1]][1]))
}

df_theft_time <- df_theft %>%
      mutate(Hour = sapply(Time, get_hour)) %>%
      group_by(DayOfWeek, Hour) %>%
      summarize(count = n())
    # df_theft_time %>% head(10)
datatable(df_theft_time, options = list(scrollX='400px'))
```



Show 10 entries Search:

	DayOfWeek	Hour	count
1	Friday	0	1932
2	Friday	1	1126
3	Friday	2	711
4	Friday	3	488
5	Friday	4	316
6	Friday	5	317
7	Friday	6	524

Showing 1 to 10 of 168 entries

Previous 1 2 3 4 5 … 17 Next

Reorder and format Factors

In this section, we demonstrate how to reorder and format factors using the aggregated data. For instance, the rev function reverses elements so that the days of the week are "Saturday", "Friday", "Thursday", "Wednesday", "Tuesday", "Monday" and "Sunday." We use the factor function to encode a vector of times as a factor (the terms 'category' and 'enumerated type' are also used for factors)., thereby using the hours 12AM through 11PM for Time, as shown in the Table.

```
dow_format <-
c("Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday")
hour_format <- c(paste(c(12,1:11),"AM"), paste(c(12,1:11),"PM"))
```

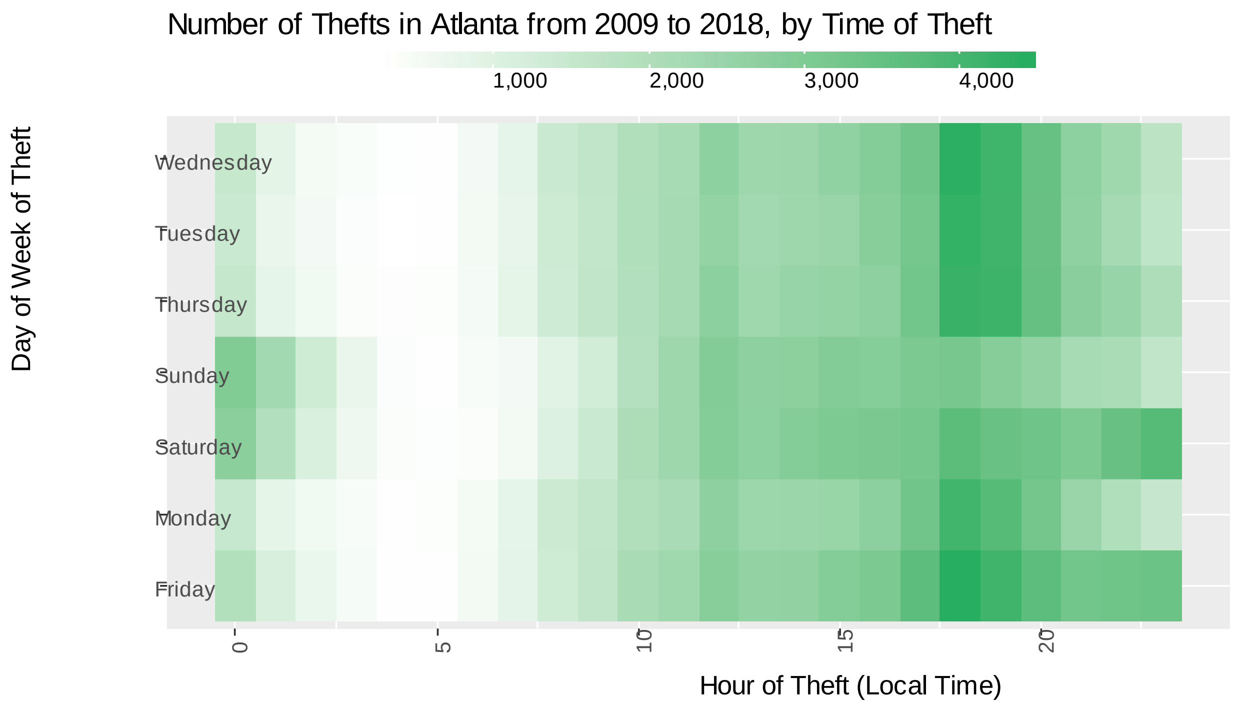
df_theft_time$DayOfWeek < -factor(df_theft_time$DayOfWeek, level = rev(dow_format)) df_theft_time$Hour < -factor(df_theft_time$Hour, level = 0:23, label = hour_format)

```
datatable(df_theft_time, options = list(scrollX='400px'))
```

Create Time Heatmap

Using our previous results, we build a "heatmap" and plot it with a red color scheme as seen in Figure 4-7. If you have not noticed, most of the colors I am using are in hexadecimal (hex) numbers, like "#000000" instead of black. A great interactive website to get colors with hex number is for any color is https://www.colorhexa.com/000000. The website also suggests "web safe colors" as alternatives.

```
plot <- ggplot(df_theft_time, aes(x = Hour, y = DayOfWeek, fill = count)) +
      geom_tile() +
      theme(axis.text.x = element_text(angle = 90, vjust = 0.6),
          legend.title = element_blank(),            legend.position="top",
          legend.direction="horizontal", legend.key.width=unit(2, "cm"),
          legend.key.height=unit(0.25, "cm"), legend.margin=unit(-0.5,"cm"),
          panel.margin=element_blank()) +
      labs(x = "Hour of Theft (Local Time)", y = "Day of Week of Theft",
          title = "Number of Thefts in Chennai from 2009 to 2018, by Time of
Theft") +
      scale_fill_gradient(low = "white", high = "#27AE60", labels = comma)
plot
```



Note that most of the code is for the legend and its formatting. The line of code that gives the heatmap its "hear" is the scale_fill_gradient() function with its low and high intensity fill colors.

The graph brings up a question: why is there a surge at 6-7 PM on weekdays? Note that Saturday and Sunday are in the middle running and the time axis is in 24-hour time. Law enforcement crime experts would probably be able to explain the "heat," but without seeing the information provided by the data, they may not realize that 6-7 PM on weekdays is an issue.

Arrest Over Time

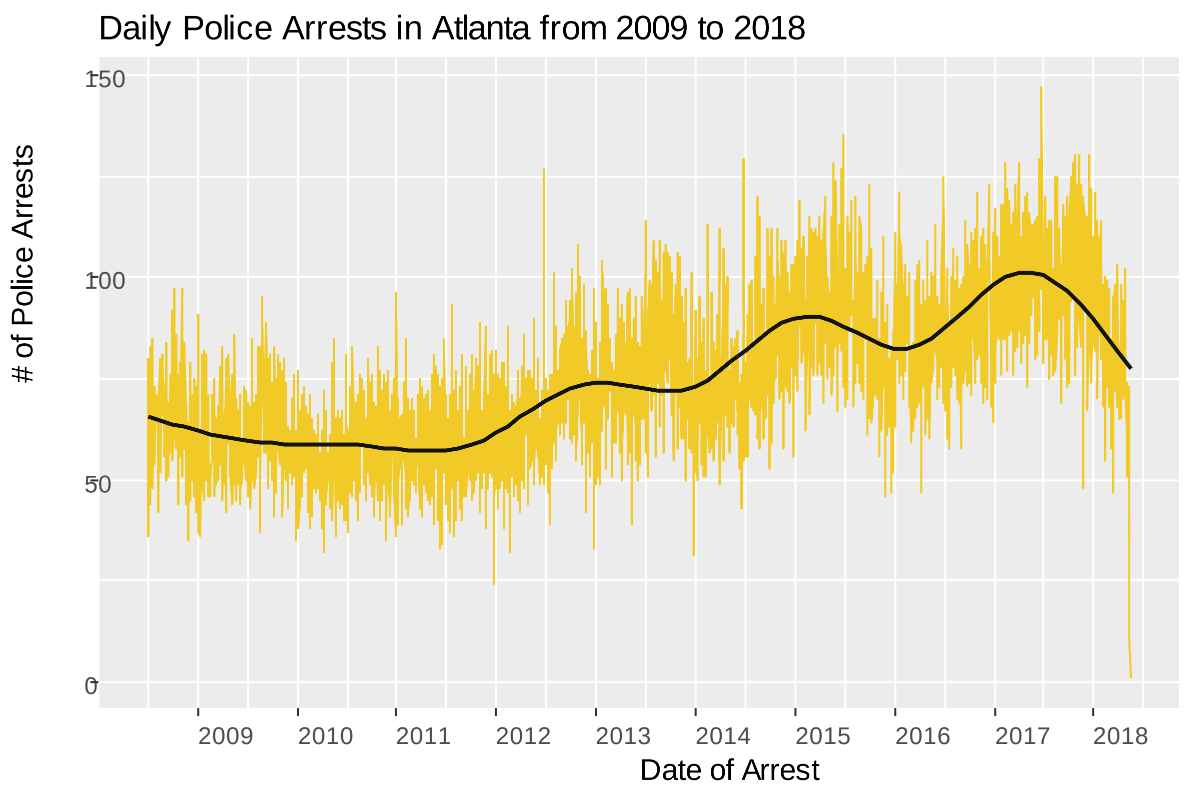
Now, we create a chart of arrests over time. First, we setup the data to get arrest counts by date. Then we plot the number of thefts given the date of the theft.

```
df_arrest <- df %>% filter(grepl("Arrest", Resolution))

df_arrest_daily <- df_arrest %>%
      mutate(Date = as.Date(Date, "%m/%d/%Y")) %>%
      group_by(Date) %>%
      summarize(count = n()) %>%
      arrange(Date)
```

Daily Arrests

Next, we build the plot shown in Figure 4-8 of daily theft arrests by police or theft arrests over time. This will provide us with another crime series and we will smooth it as usual, noticing a downward trend in arrests over time until a sharp increase starting around 2017.

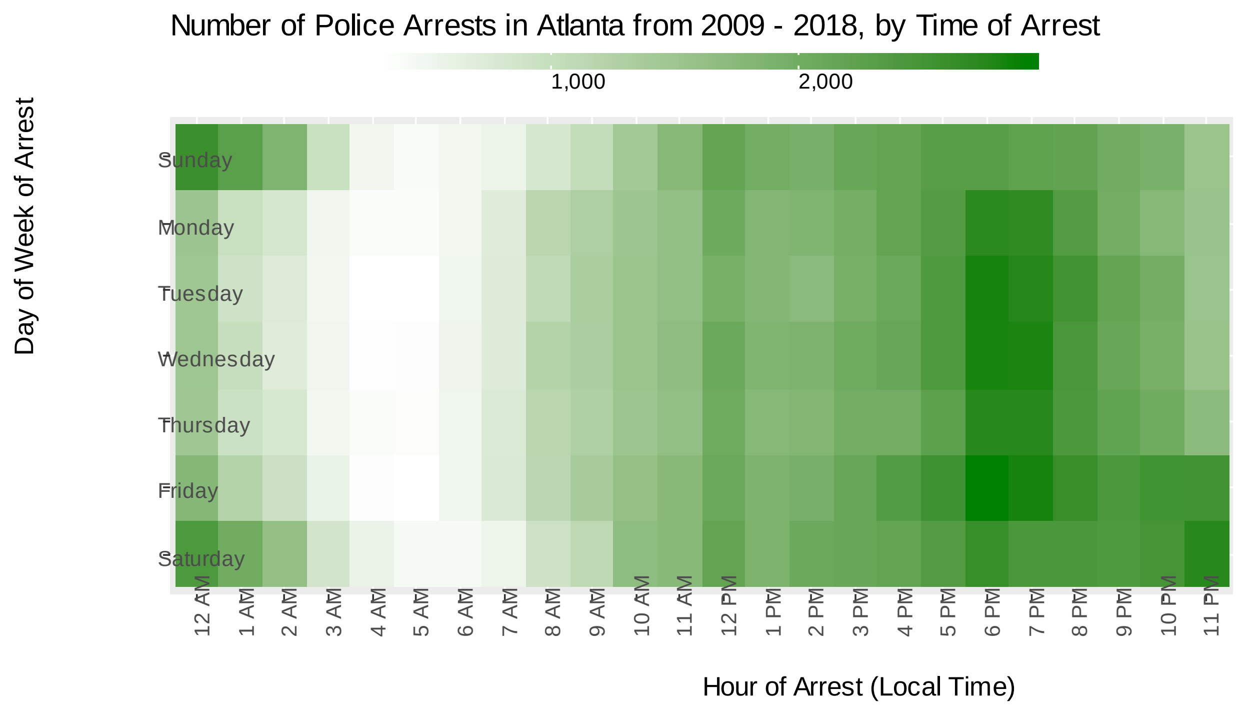
```
library(ggplot2)
library(scales)
plot <- ggplot(df_arrest_daily, aes(x = Date, y = count)) +
      geom_line(color = "#F2CA27", size = 0.1) +
      geom_smooth(color = "#1A1A1A") +
      # fte_theme() +
      scale_x_date(breaks = date_breaks("1 year"), labels =
date_format("%Y")) +
      labs(x = "Date of Arrest", y = "# of Police Arrests",
      title = "Daily Police Arrests in Chennai from 2009 to 2018")
plot
```



Number of Arrest by TIme of Arrest

Here, we again use the function we created that gets the hour from the time string in chennai_crime.csv, so that we can use an approximate arrest time with the day of the week. This allows us to bin the crimes by hour, etc.

```
get_hour <- function(x) {
      return (as.numeric(strsplit(x,":")[[1]][1]))
  }

df_arrest_time <- df_arrest %>%
      mutate(Hour = sapply(Time, get_hour)) %>%
      group_by(DayOfWeek, Hour) %>%
      summarize(count = n())

dow_format <-
c("Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday")
hour_format <- c(paste(c(12,1:11),"AM"), paste(c(12,1:11),"PM"))

df_arrest_time$DayOfWeek <- factor(df_arrest_time$DayOfWeek, level =
rev(dow_format))
df_arrest_time$Hour <- factor(df_arrest_time$Hour, level = 0:23, label =
hour_format)

plot <- ggplot(df_arrest_time, aes(x = Hour, y = DayOfWeek, fill = count)) +
      geom_tile() +
      # fte_theme() +
      theme(axis.text.x = element_text(angle = 90, vjust = 0.6),
            legend.title = element_blank(), legend.position="top",
            legend.direction="horizontal", legend.key.width=unit(2, "cm"),
            legend.key.height=unit(0.25, "cm"), legend.margin=unit(-
0.5,"cm"),
            panel.margin=element_blank()) +
      labs(x = "Hour of Arrest (Local Time)", y = "Day of Week of Arrest",
           title = "Number of Police Arrests in Chennai from 2009 - 2018, by
Time of Arrest") +
      scale_fill_gradient(low = "white", high = "#008000", labels = comma)
plot
```



From the figure, most arrests are made on Thursdays and Fridays during the midnight hour. The fact that arrests seem to occur around 12AM, 5, AM, 10AM, 3PM, and 8PM may simply be shift changes and perhaps the reporting times correspond with the shift changes.

Why is there a surge on Wednesday afternoon, and at 4-5PM on all days? Let's look at subgroups to verify there isn't a latent factor.

Correlation Analysis

Factor by Crime Category

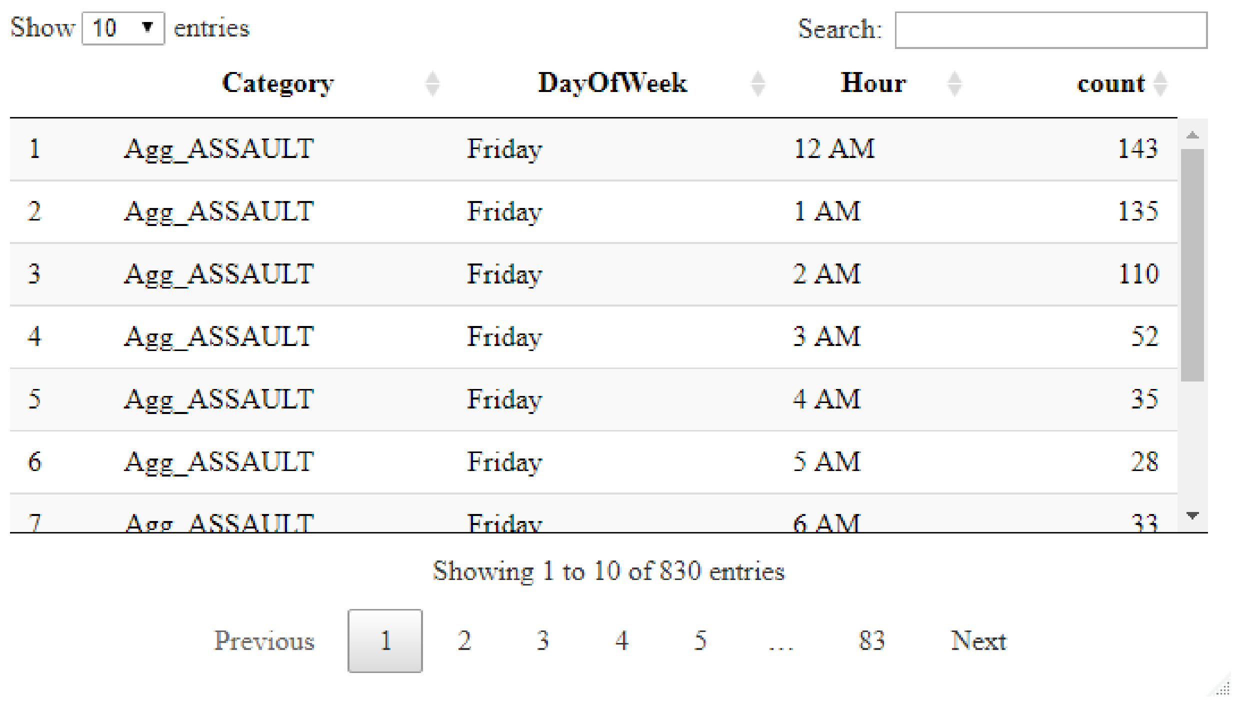
Certain types of crime may be more time dependent. (e.g., more traffic violations when people leave work). While we are interested in crime frequencies as shown in Table 4-6, we can gain more information from the data. For instance, in the table below, we look at frequency of the crime category per hour.

```
df_top_crimes <- df_arrest %>%
      group_by(Category) %>%
      summarize(count = n()) %>%
      arrange(desc(count))
#datatable(df_top_crimes, options = list(pageLength = 10,scrollX='400px'))
knitr::kable(df_top_crimes, caption = 'Top Crimes Table')
```

Table: Top Crimes Table

Category count ——— ——

```
df_arrest_time_crime <- df_arrest %>%
      filter(Category %in% df_top_crimes$Category[2:19]) %>%
      mutate(Hour = sapply(Time, get_hour)) %>%
      group_by(Category, DayOfWeek, Hour) %>%
      summarize(count = n())
df_arrest_time_crime$DayOfWeek <- factor(df_arrest_time_crime$DayOfWeek,
                                         level = rev(dow_format))
df_arrest_time_crime$Hour <- factor(df_arrest_time_crime$Hour,
                                    level = 0:23, label = hour_format)
#datatable(df_arrest_time_crime, options = list(pageLength =
10,scrollX='400px'))
knitr::kable(df_arrest_time_crime,
             caption = ' Arrest frequency table by day-of-week and hour
(time)')
```


Show 10 entries Search:

	Category	DayOfWeek	Hour	count
1	Agg_ASSAULT	Friday	12 AM	143
2	Agg_ASSAULT	Friday	1 AM	135
3	Agg_ASSAULT	Friday	2 AM	110
4	Agg_ASSAULT	Friday	3 AM	52
5	Agg_ASSAULT	Friday	4 AM	35
6	Agg_ASSAULT	Friday	5 AM	28
7	Agg_ASSAULT	Friday	6 AM	33

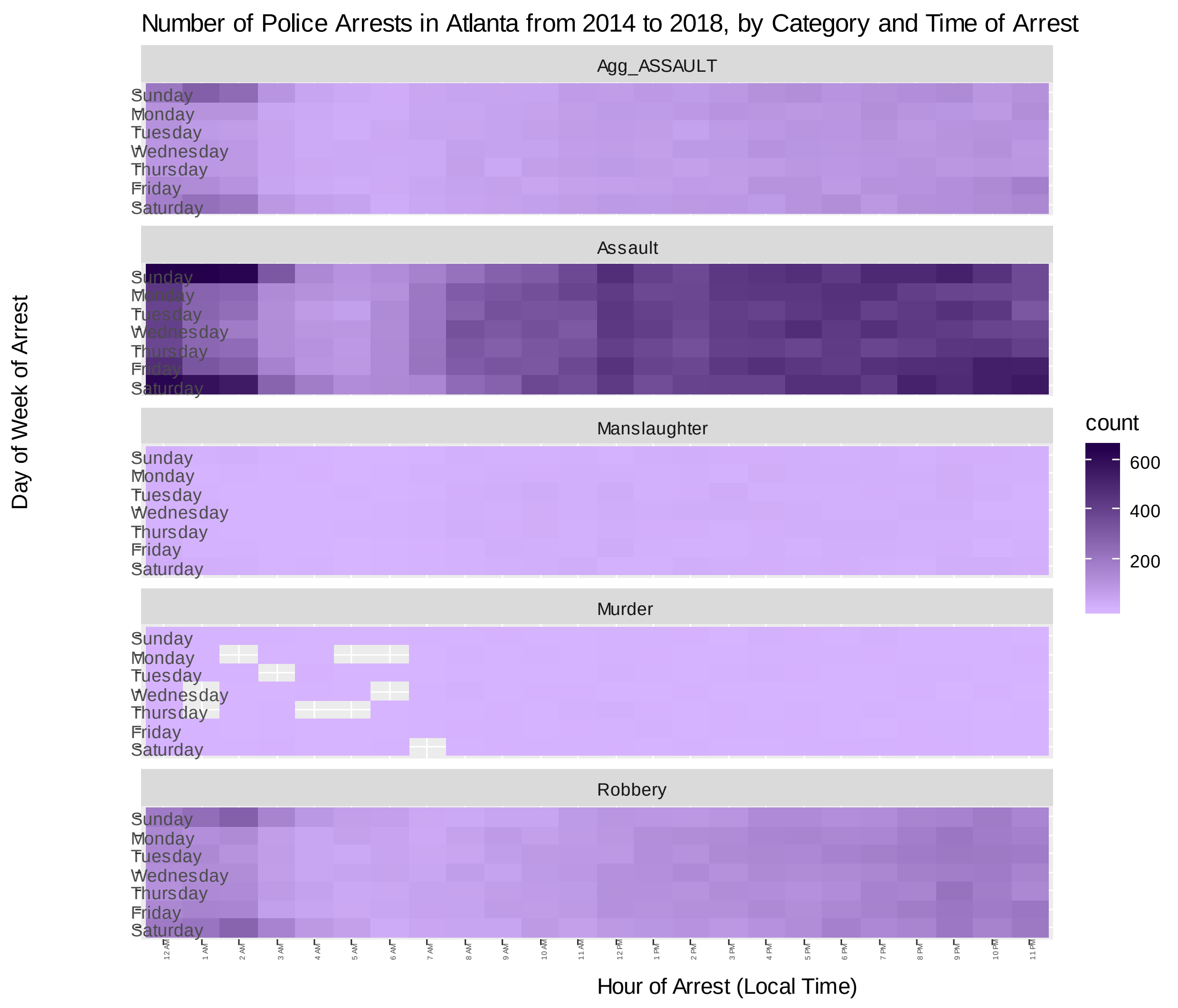
Showing 1 to 10 of 830 entries

Previous 1 2 3 4 5 … 83 Next

Number of Arrests by Category and time of Arrest

In this section, we plot the number of arrest by category and time of arrest. This leads us to use a chart type that you may not have seen very often. We take the heat map application from the previous section and plot all the crime heat maps in one aggregated chart as seen in the figure below. For each heat map in the chart, the horizontal axes are "hours of arrest (local time)," and the vertical axes are "days of the week."

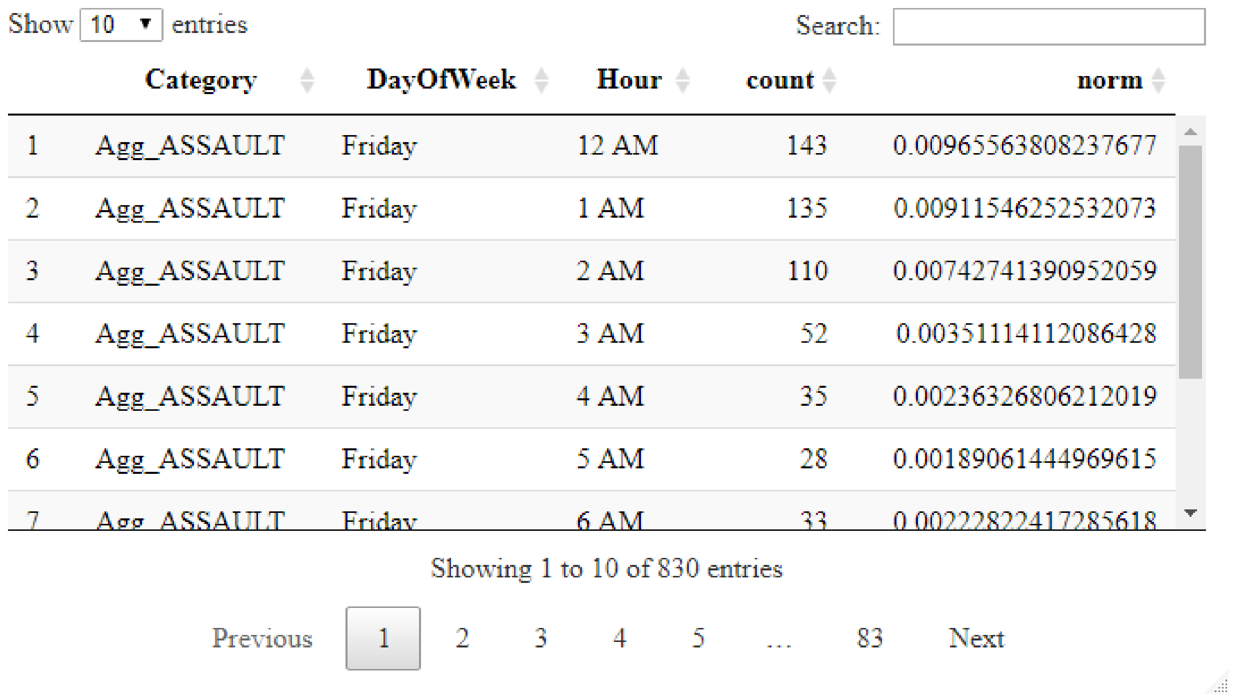
```
plot <- ggplot(df_arrest_time_crime, aes(x = Hour, y = DayOfWeek, fill =
count)) +
      geom_tile() +
      # fte_theme() +
      theme(axis.text.x = element_text(angle = 90, vjust = 0.6, size = 4)) +
      labs(x = "Hour of Arrest (Local Time)", y = "Day of Week of Arrest",
title = "Number of Police Arrests in Chennai from 2014 to 2018, by Category
and Time of Arrest") +
      scale_fill_gradient(low = "#d7b4ff", high = "#24004b") +
      facet_wrap(~ Category, nrow = 6)
plot
```



This graph looks good, but the gradients aren't helpful because they are not normalized. AS used here, normalization refers to adjustments in the measured scale where the intention is to bring the entire probability distributions of adjusted values into alignment. In this way, we can make one-to-one comparisons. We need to normalize the range on each facet as we do for Table 4-8 and the corresponding figure.

Normailzed Gradients

```
df_arrest_time_crime <- df_arrest_time_crime %>%
      group_by(Category) %>%
      mutate(norm = count/sum(count))

datatable(df_arrest_time_crime, options = list(pageLength =
10,scrollX='400px'))
```



Show 10 entries Search:

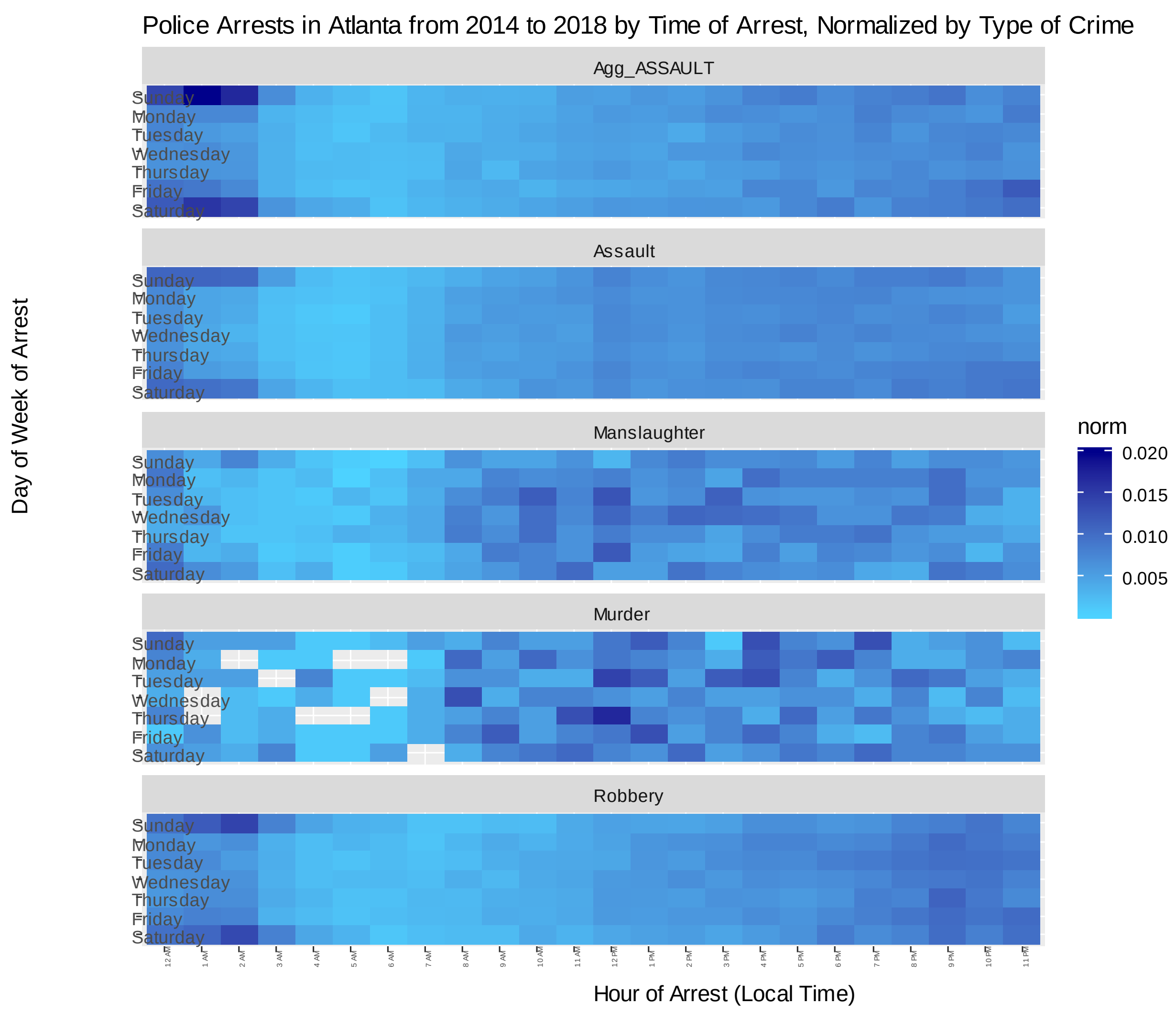
	Category	DayOfWeek	Hour	count	norm
1	Agg_ASSAULT	Friday	12 AM	143	0.00965563808237677
2	Agg_ASSAULT	Friday	1 AM	135	0.00911546252532073
3	Agg_ASSAULT	Friday	2 AM	110	0.00742741390952059
4	Agg_ASSAULT	Friday	3 AM	52	0.00351114112086428
5	Agg_ASSAULT	Friday	4 AM	35	0.00236326806212019
6	Agg_ASSAULT	Friday	5 AM	28	0.00189061444969615
7	Agg_ASSAULT	Friday	6 AM	33	0.00222822417285618

Showing 1 to 10 of 830 entries

Previous 1 2 3 4 5 … 83 Next

Normalized Number of Arrests by Category and Time of Arrest

```
plot <- ggplot(df_arrest_time_crime, aes(x = Hour, y = DayOfWeek, fill =
norm)) +
      geom_tile() +
      # fte_theme() +
      theme(axis.text.x = element_text(angle = 90, vjust = 0.6, size = 4)) +
      labs(x = "Hour of Arrest (Local Time)", y = "Day of Week of Arrest",
           title = "Police Arrests in Chennai from 2014 to 2018 by Time of
Arrest,
           Normalized by Type of Crime") +
      scale_fill_gradient(low = "#4dd2ff", high = "#00008b") +
      facet_wrap(~ Category, nrow = 6)
plot
```

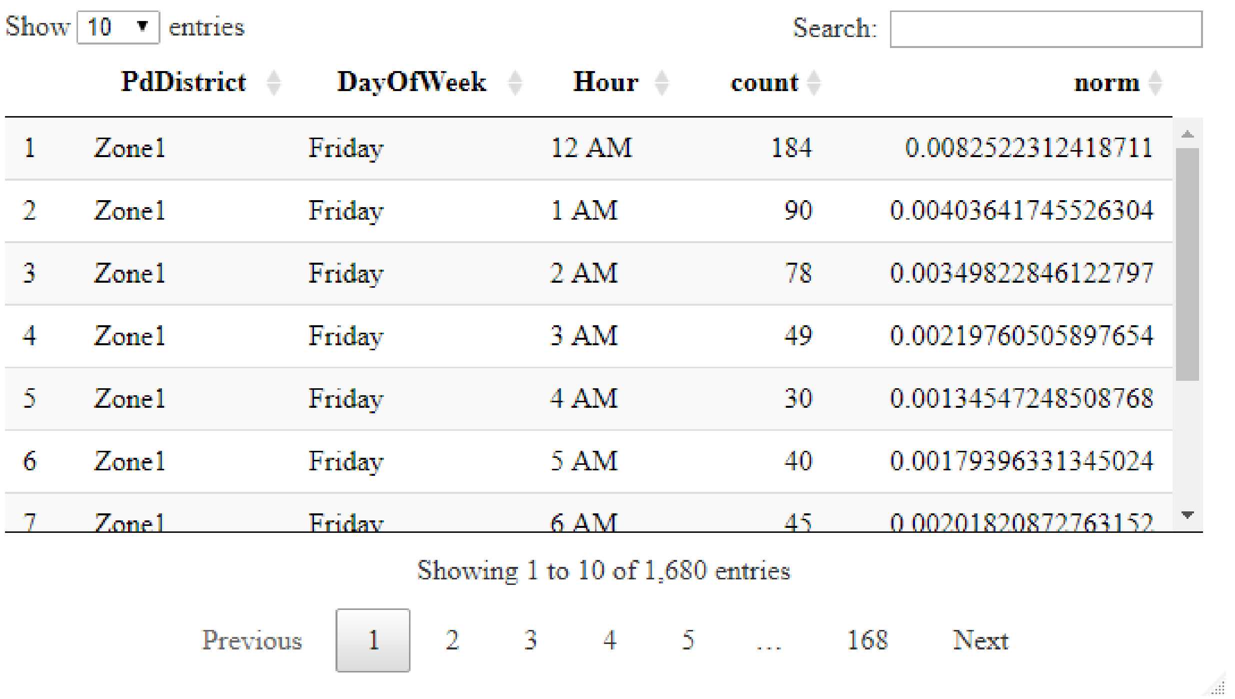


Factor by Police District

In this section, we plot like we did for Figure 4-10, but with a different scope. In Table 4-9 and its corresponding Figure 4-12, we want the normalized frequency of arrest from each PD district (or zone in this case) by day-of-week and hour.

```
df_arrest_time_district <- df_arrest %>%
      mutate(Hour = sapply(Time, get_hour)) %>%
      group_by(PdDistrict, DayOfWeek, Hour) %>%
      summarize(count = n()) %>%
      group_by(PdDistrict) %>%
      mutate(norm = count/sum(count))

df_arrest_time_district$DayOfWeek <-
factor(df_arrest_time_district$DayOfWeek, level = rev(dow_format))
df_arrest_time_district$Hour <- factor(df_arrest_time_district$Hour, level =
0:23, label = hour_format)
```

```
datatable(df_arrest_time_district, options = list(pageLength =
10,scrollX='400px'))
```



Show 10 entries   Search:

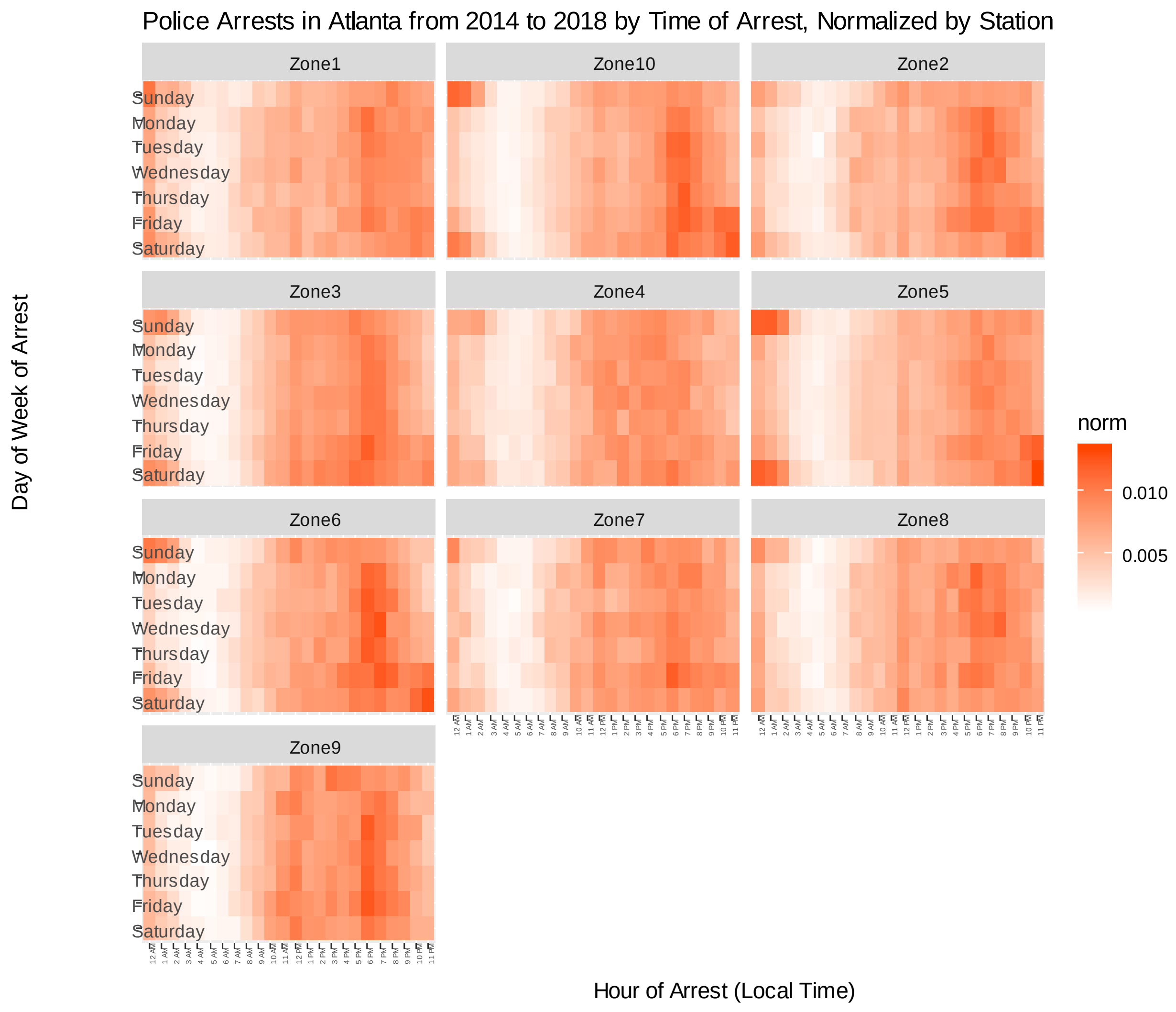
	PdDistrict	DayOfWeek	Hour	count	norm
1	Zone1	Friday	12 AM	184	0.0082522312418711
2	Zone1	Friday	1 AM	90	0.00403641745526304
3	Zone1	Friday	2 AM	78	0.00349822846122797
4	Zone1	Friday	3 AM	49	0.00219760505897654
5	Zone1	Friday	4 AM	30	0.00134547248508768
6	Zone1	Friday	5 AM	40	0.00179396331345024
7	Zone1	Friday	6 AM	45	0.00201820872763152

Showing 1 to 10 of 1,680 entries

Previous 1 2 3 4 5 … 168 Next

Factor by Police District

```
plot <- ggplot(df_arrest_time_district, aes(x = Hour, y = DayOfWeek, fill =
norm)) +
      geom_tile() +
      theme(axis.text.x = element_text(angle = 90, vjust = 0.6, size = 4)) +
      labs(x = "Hour of Arrest (Local Time)", y = "Day of Week of Arrest",
title = "Police Arrests in Chennai from 2014 to 2018 by Time of Arrest,
Normalized by Station") +
      scale_fill_gradient(low = "white", high = "#ff4500") +
      facet_wrap(~ PdDistrict, nrow = 4)
plot
```



Factor by Month

We now look at factor by month. If crime is tied to activities, the period at which activities end may impact.

```
df_arrest_time_month <- df_arrest %>%
      mutate(Month = format(as.Date(Date, "%m/%d/%Y"), "%B"), Hour =
sapply(Time, get_hour)) %>%
      group_by(Month, DayOfWeek, Hour) %>%
      summarize(count = n()) %>%
      group_by(Month) %>%
      mutate(norm = count/sum(count))
```

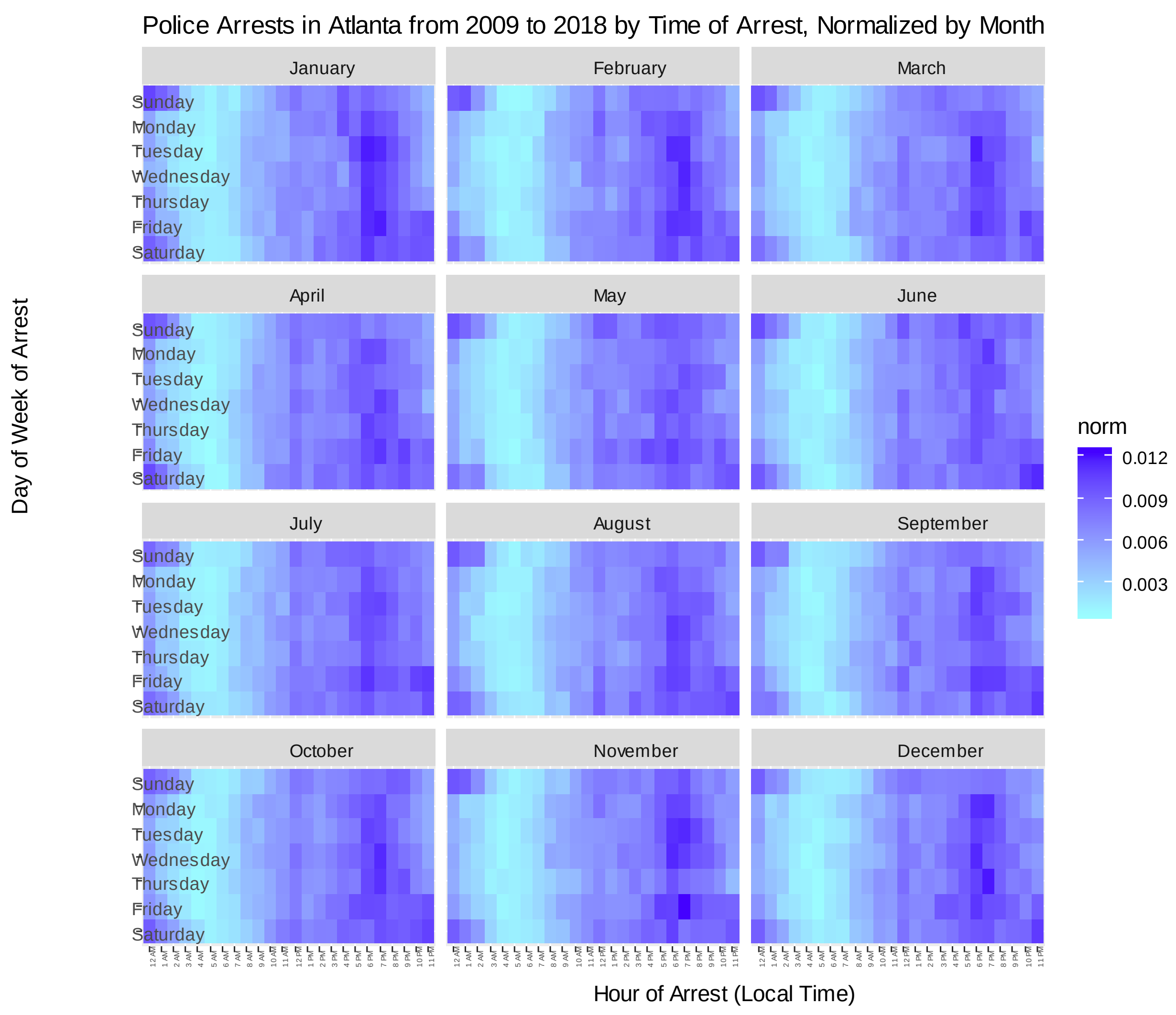
Here, we set order of month facets by chronological order instead of alphabetical.

```
df_arrest_time_month$DayOfWeek <- factor(df_arrest_time_month$DayOfWeek,
level = rev(dow_format))
df_arrest_time_month$Hour <- factor(df_arrest_time_month$Hour, level = 0:23,
label = hour_format)
```

```
df_arrest_time_month$Month <- factor(df_arrest_time_month$Month,
                                         level =
c("January","February","March","April",

"May","June","July","August",

"September","October","November","December"))
```

Now we plot the data as shown below and seen in the figure below.

```
plot <- ggplot(df_arrest_time_month, aes(x = Hour, y = DayOfWeek, fill =
norm)) +
      geom_tile() +
      theme(axis.text.x = element_text(angle = 90, vjust = 0.6, size = 4)) +
      labs(x = "Hour of Arrest (Local Time)", y = "Day of Week of Arrest",
           title = "Police Arrests in Chennai from 2009 to 2018 by Time of
Arrest,
           Normalized by Month") +
      scale_fill_gradient(low = "#9bfdff", high = "#4401ff") +
      facet_wrap(~ Month, nrow = 4)
plot
```



Factor By Year

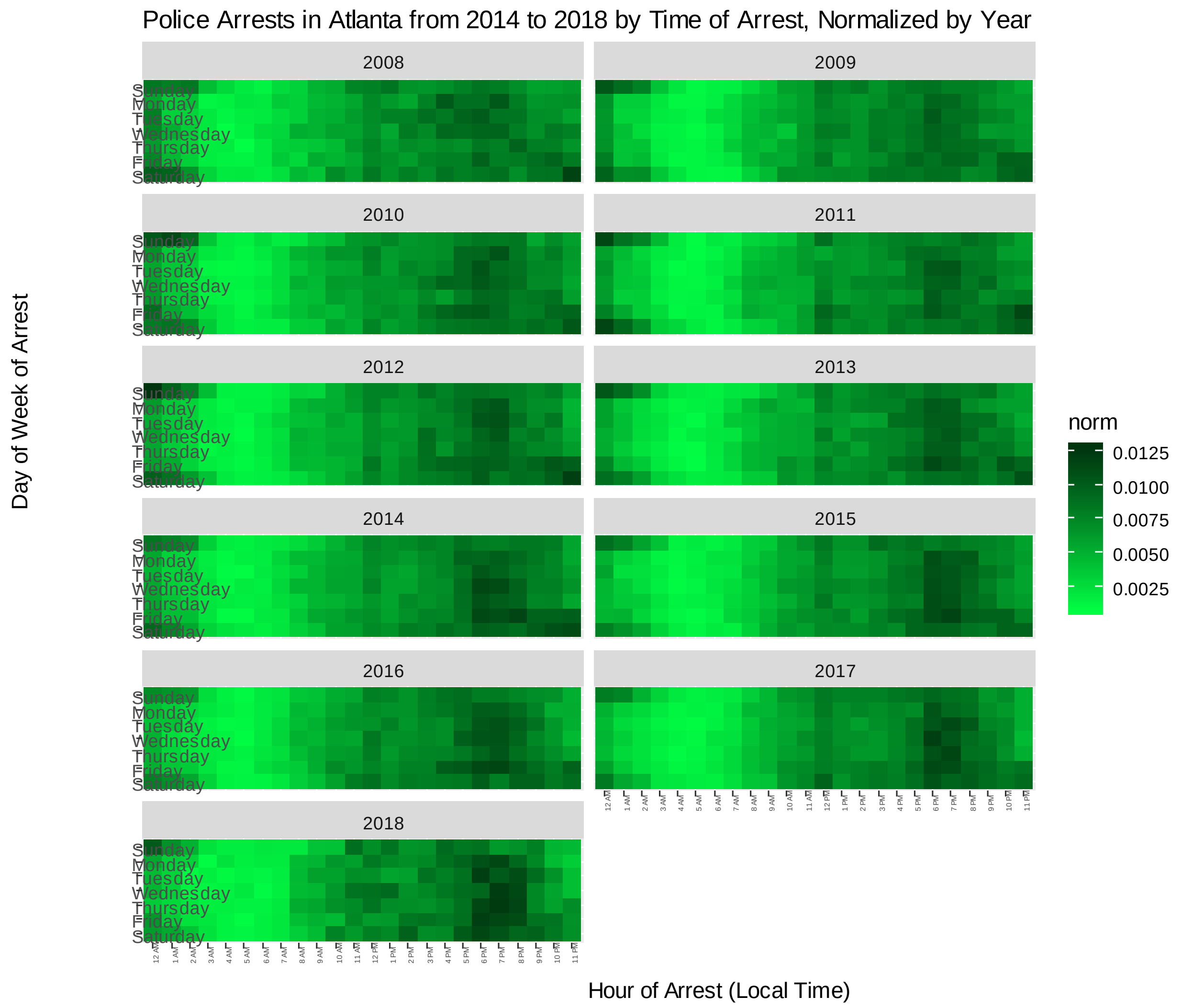
what if things changed overtime?

```
df_arrest_time_year <- df_arrest %>%
      mutate(Year = format(as.Date(Date, "%m/%d/%Y"), "%Y"), Hour =
sapply(Time, get_hour)) %>%
      group_by(Year, DayOfWeek, Hour) %>%
      summarize(count = n()) %>%
      group_by(Year) %>%
      mutate(norm = count/sum(count))

df_arrest_time_year$DayOfWeek <- factor(df_arrest_time_year$DayOfWeek, level
= rev(dow_format))
df_arrest_time_year$Hour <- factor(df_arrest_time_year$Hour, level = 0:23,
label = hour_format)
```

Police Arrest Normalized by YEar

In similar fashion, we can look at the arrests by year aggregated over day-of-week and time-of-day, as seen in the figure.

```
plot <- ggplot(df_arrest_time_year, aes(x = Hour, y = DayOfWeek, fill =
norm)) +
    geom_tile() +
    theme(axis.text.x = element_text(angle = 90, vjust = 0.6, size = 4)) +
    labs(x = "Hour of Arrest (Local Time)", y = "Day of Week of Arrest",
    title = "Police Arrests in Chennai from 2014 to 2018 by Time of Arrest,
      Normalized by Year") +
      scale_fill_gradient(low = "#01ff44", high = "#00340e") +
      facet_wrap(~ Year, nrow = 6)
plot
```



Works Cited