Advancing crime analysis with R and Shiny

Henry Partridge

github.com/cat-lord

Data Analyst, Transport for London

February 2016

Structure

- R
- Crime analysis in R
- Shiny
- Shiny app development at TfL
- Building a Shiny app

About me

- Data Analyst at Transport for London
- Background in philosophy and crime science
- Programming in R for two years

R

What is R?

- R is a programming language for statistical analysis and data visualisation
- Created by Ross Ihaka and Robert Gentleman (University of Auckland)
- Released in 1995
- Implements the S programming language created at Bell Labs
- Companies like Google, Facebook and the New York Times use it

Why use R?

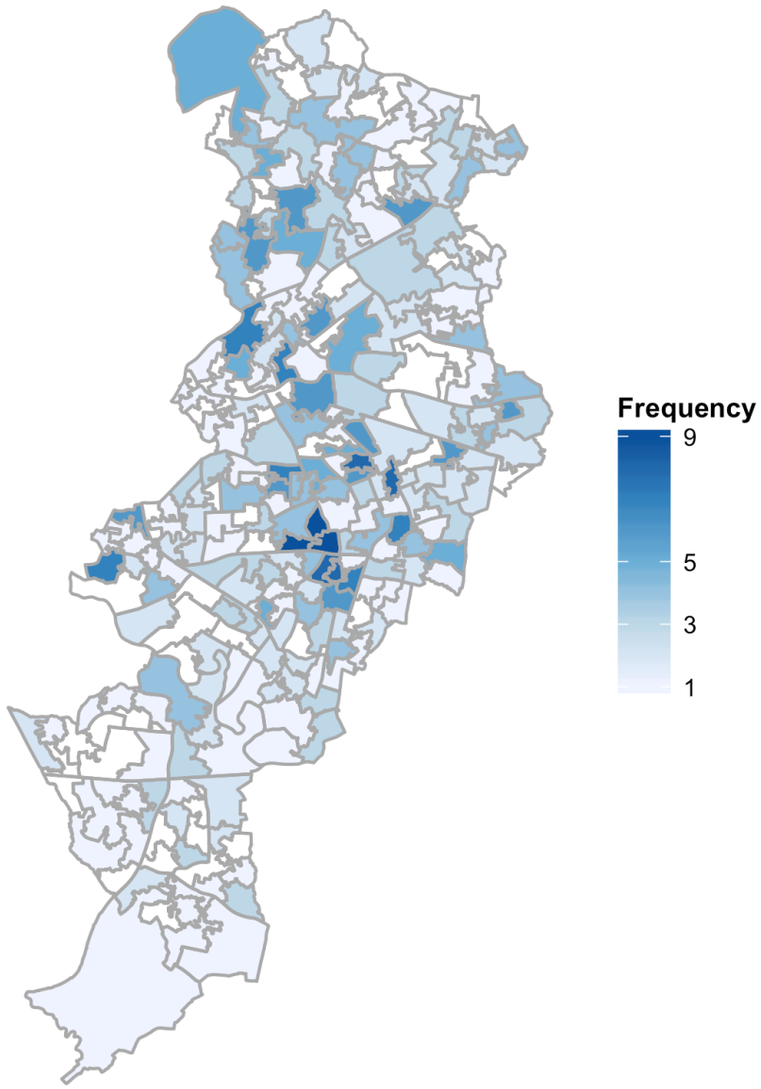
- Leading tool for statistical analysis, forecasting and machine learning
- Powerful graphics and data visualisations
- Open source
- Reproducibility
- Transparency
- Automation
- Support network

Crime analysis in R

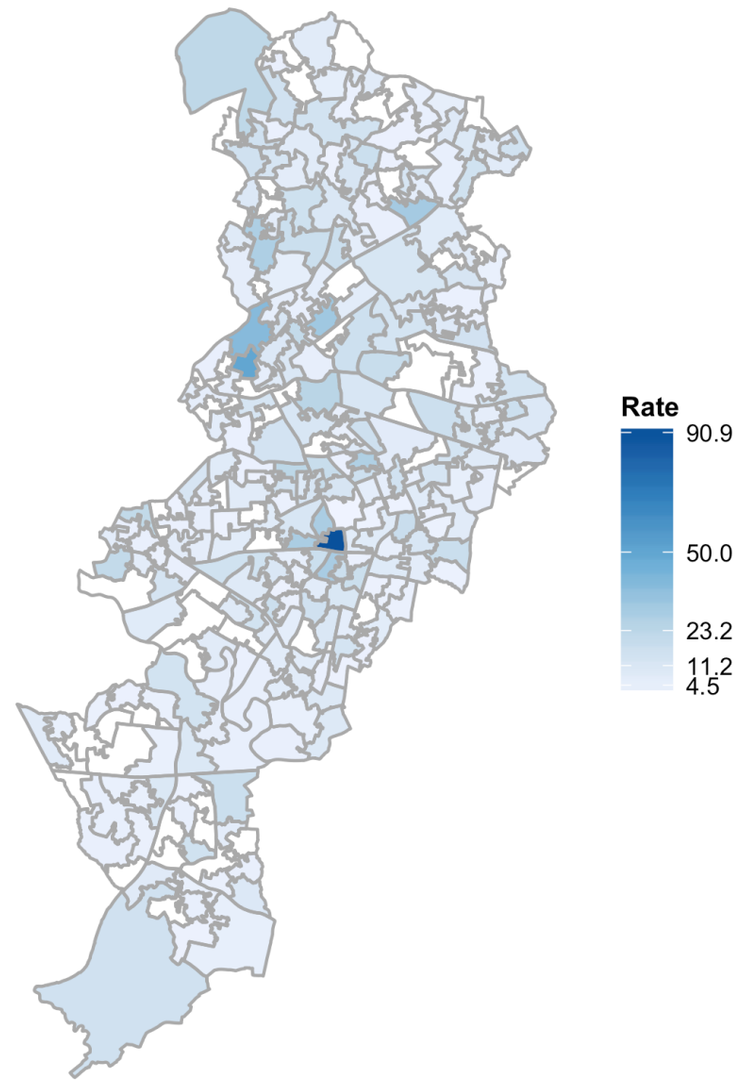
Choropleth maps

ggplot2

Frequency of crime



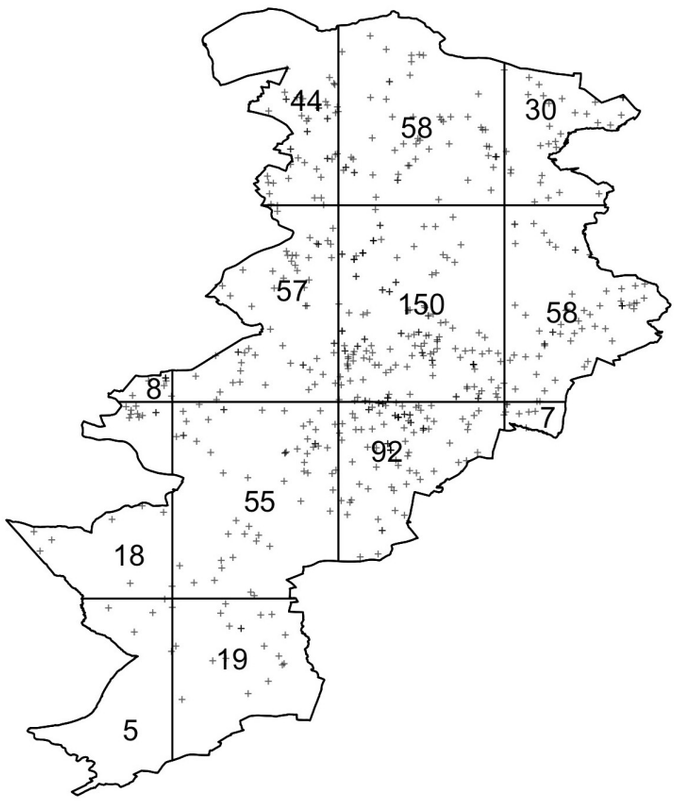
Rate per 1,000 population



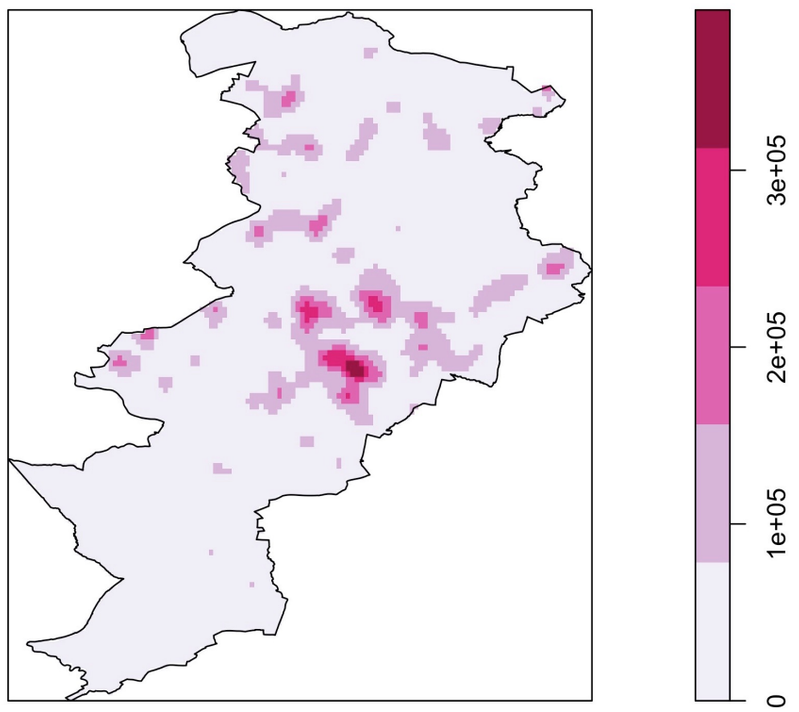
Point pattern analysis

spatstat

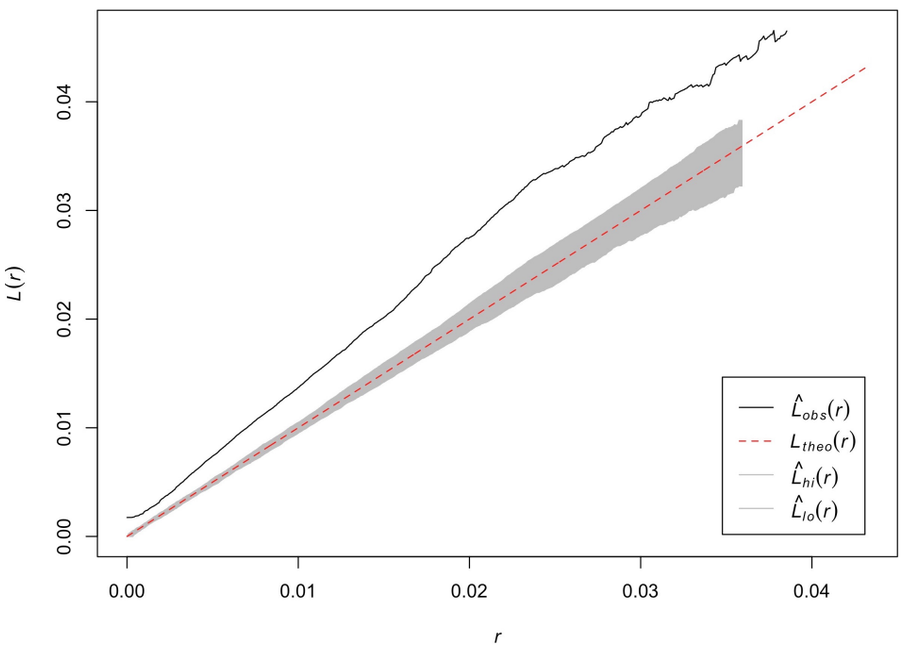
Quadrat counts



Kernel density estimation

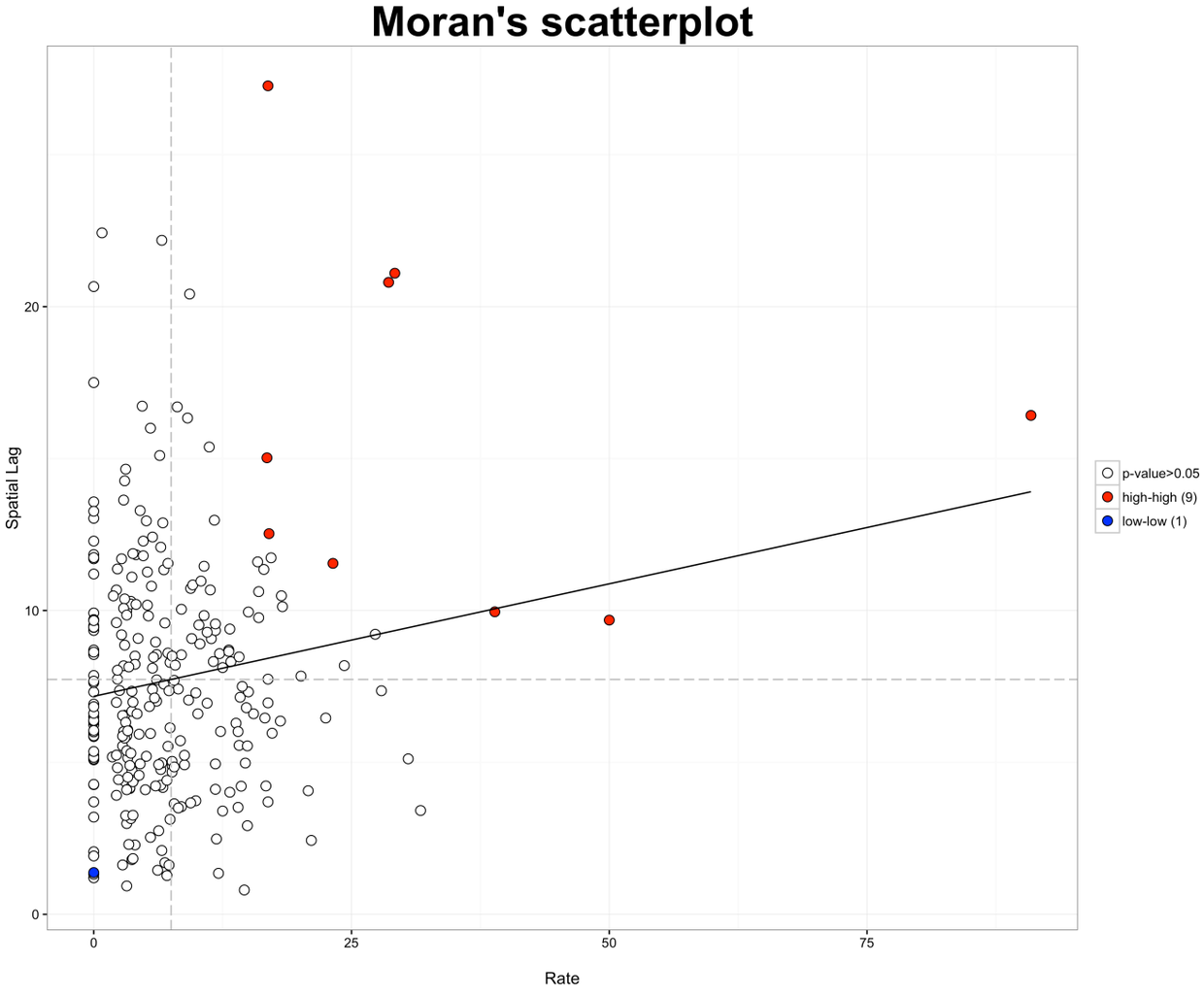


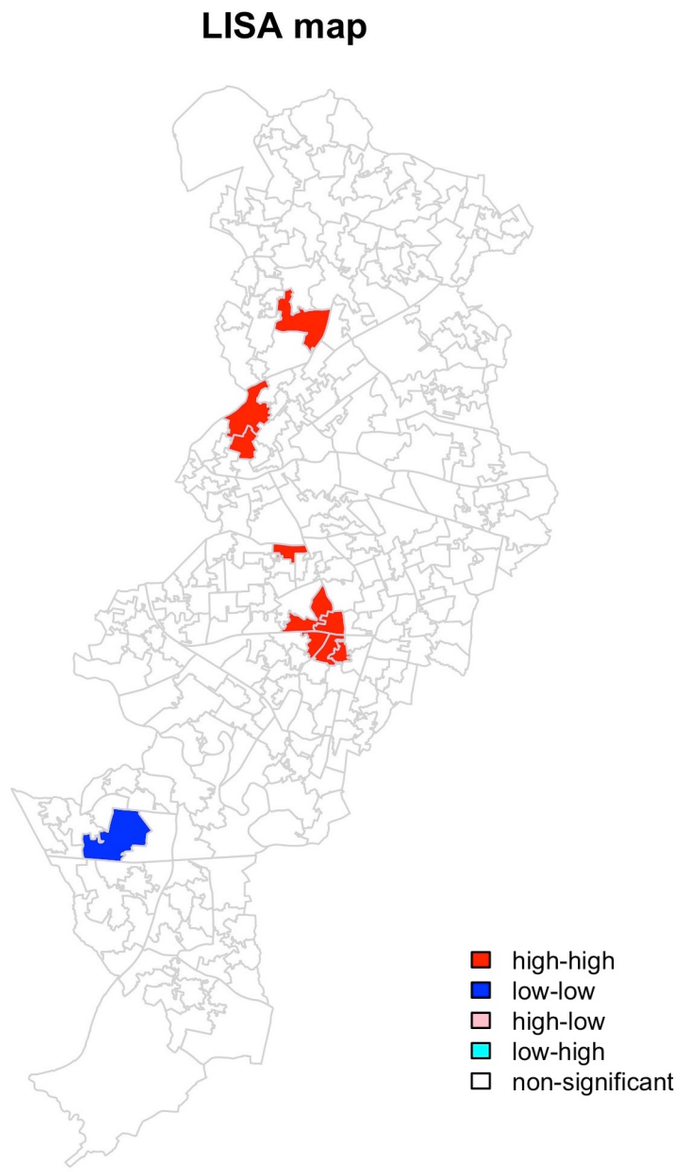
L-Function



Spatial autocorrelation

spdep

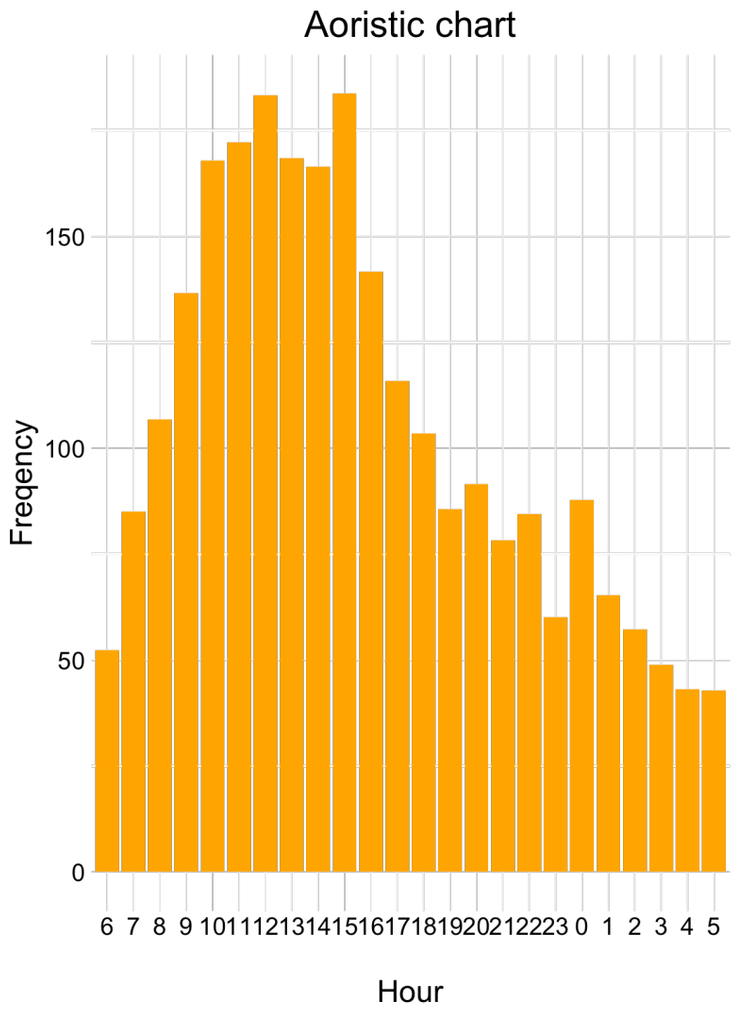




Aoristic analysis

aoristic

```
Observations: 2,484
Variables: 27
$ id       (dbl) 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, ...
$ HourFrom (int) 7, 2, 9, 6, 5, 4, 12, 23, 3, 15, 9, 13, 0, 18, 14, 6, 11, 7, 22, 7, 10, 22, 22, 4, ...
$ duration (dbl) 1, 1, 14, 8, 18, 2, 1, 1, 1, 1, 5, 2, 2, 24, 24, 12, 4, 7, 2, 24, 14, 1, 10, 10, 11...
$ time0    (dbl) 0.00000000, 0.00000000, 0.00000000, 0.00000000, 0.00000000, 0.00000000, 0.00000000,...
$ time1    (dbl) 0.00000000, 0.00000000, 0.00000000, 0.00000000, 0.00000000, 0.00000000, 0.00000000,...
$ time2    (dbl) 0.00000000, 1.00000000, 0.00000000, 0.00000000, 0.00000000, 0.00000000, 0.00000000,...
$ time3    (dbl) 0.00000000, 0.00000000, 0.00000000, 0.00000000, 0.00000000, 0.00000000, 0.00000000,...
$ time4    (dbl) 0.00000000, 0.00000000, 0.00000000, 0.00000000, 0.00000000, 0.50000000, 0.00000000,...
$ time5    (dbl) 0.00000000, 0.00000000, 0.00000000, 0.00000000, 0.05555556, 0.50000000, 0.00000000,...
$ time6    (dbl) 0.00000000, 0.00000000, 0.00000000, 0.12500000, 0.05555556, 0.00000000, 0.00000000,...
$ time7    (dbl) 1.00000000, 0.00000000, 0.00000000, 0.12500000, 0.05555556, 0.00000000, 0.00000000,...
$ time8    (dbl) 0.00000000, 0.00000000, 0.00000000, 0.12500000, 0.05555556, 0.00000000, 0.00000000,...
$ time9    (dbl) 0.00000000, 0.00000000, 0.07142857, 0.12500000, 0.05555556, 0.00000000, 0.00000000,...
$ time10   (dbl) 0.00000000, 0.00000000, 0.07142857, 0.12500000, 0.05555556, 0.00000000, 0.00000000,...
$ time11   (dbl) 0.00000000, 0.00000000, 0.07142857, 0.12500000, 0.05555556, 0.00000000, 0.00000000,...
$ time12   (dbl) 0.00000000, 0.00000000, 0.07142857, 0.12500000, 0.05555556, 0.00000000, 1.00000000,...
$ time13   (dbl) 0.00000000, 0.00000000, 0.07142857, 0.12500000, 0.05555556, 0.00000000, 0.00000000,...
$ time14   (dbl) 0.00000000, 0.00000000, 0.07142857, 0.00000000, 0.05555556, 0.00000000, 0.00000000,...
$ time15   (dbl) 0.00000000, 0.00000000, 0.07142857, 0.00000000, 0.05555556, 0.00000000, 0.00000000,...
$ time16   (dbl) 0.00000000, 0.00000000, 0.07142857, 0.00000000, 0.05555556, 0.00000000, 0.00000000,...
$ time17   (dbl) 0.00000000, 0.00000000, 0.07142857, 0.00000000, 0.05555556, 0.00000000, 0.00000000,...
$ time18   (dbl) 0.00000000, 0.00000000, 0.07142857, 0.00000000, 0.05555556, 0.00000000, 0.00000000,...
$ time19   (dbl) 0.00000000, 0.00000000, 0.07142857, 0.00000000, 0.05555556, 0.00000000, 0.00000000,...
$ time20   (dbl) 0.00000000, 0.00000000, 0.07142857, 0.00000000, 0.05555556, 0.00000000, 0.00000000,...
$ time21   (dbl) 0.00000000, 0.00000000, 0.07142857, 0.00000000, 0.05555556, 0.00000000, 0.00000000,...
$ time22   (dbl) 0.00000000, 0.00000000, 0.07142857, 0.00000000, 0.05555556, 0.00000000, 0.00000000,...
$ time23   (dbl) 0.00000000, 0.00000000, 0.00000000, 0.00000000, 0.00000000, 0.00000000, 0.00000000,...
```



Other types of analysis

- Geographic profiling - **Rgeoprofile**
- Crime series identification - **crimelinkage**
- Text mining - **tm**
- Network analysis - **igraph**

Getting started

- Download R from CRAN and RStudio
- Try out some online tutorials
- If you get stuck search on stackoverflow.com using “[R]” to limit your results
- Keep up to date with r-bloggers.com and search Twitter for #rstats
- Attend a local meetup group like ManchesterR, EdinbR or LondonR

Shiny

What is Shiny?

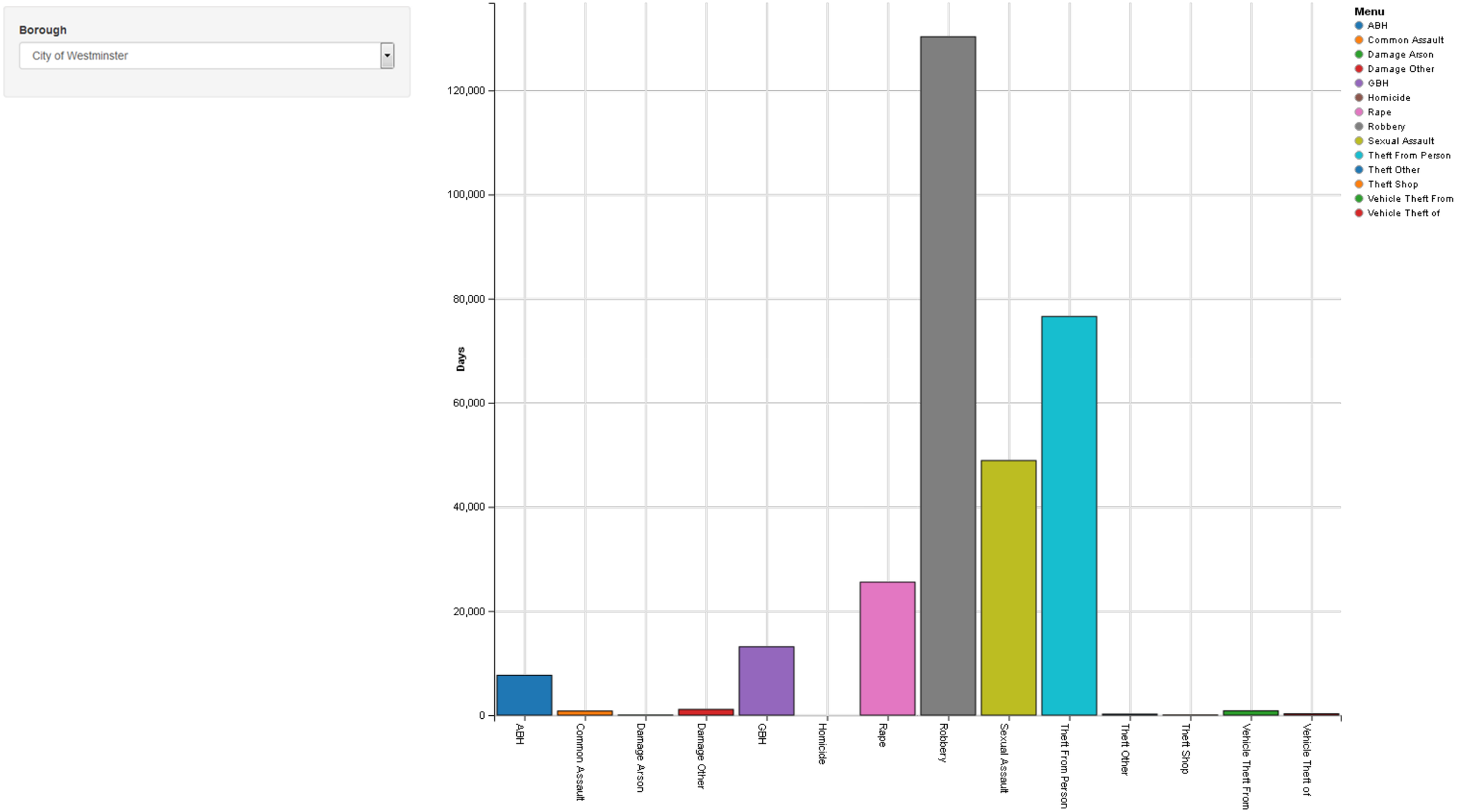
- An R package developed by RStudio that allows data analysts to analyse, visualise and share their results with non-R users
- Interactive web applications connected to an R session
- No knowledge of HTML, CSS, and JavaScript is required but web apps are customisable and extendible
- Integrates with JavaScript libraries
- Uses a reactive programming framework. An input is sent to an R process which generates a plot in a web browser.

Advantages of Shiny

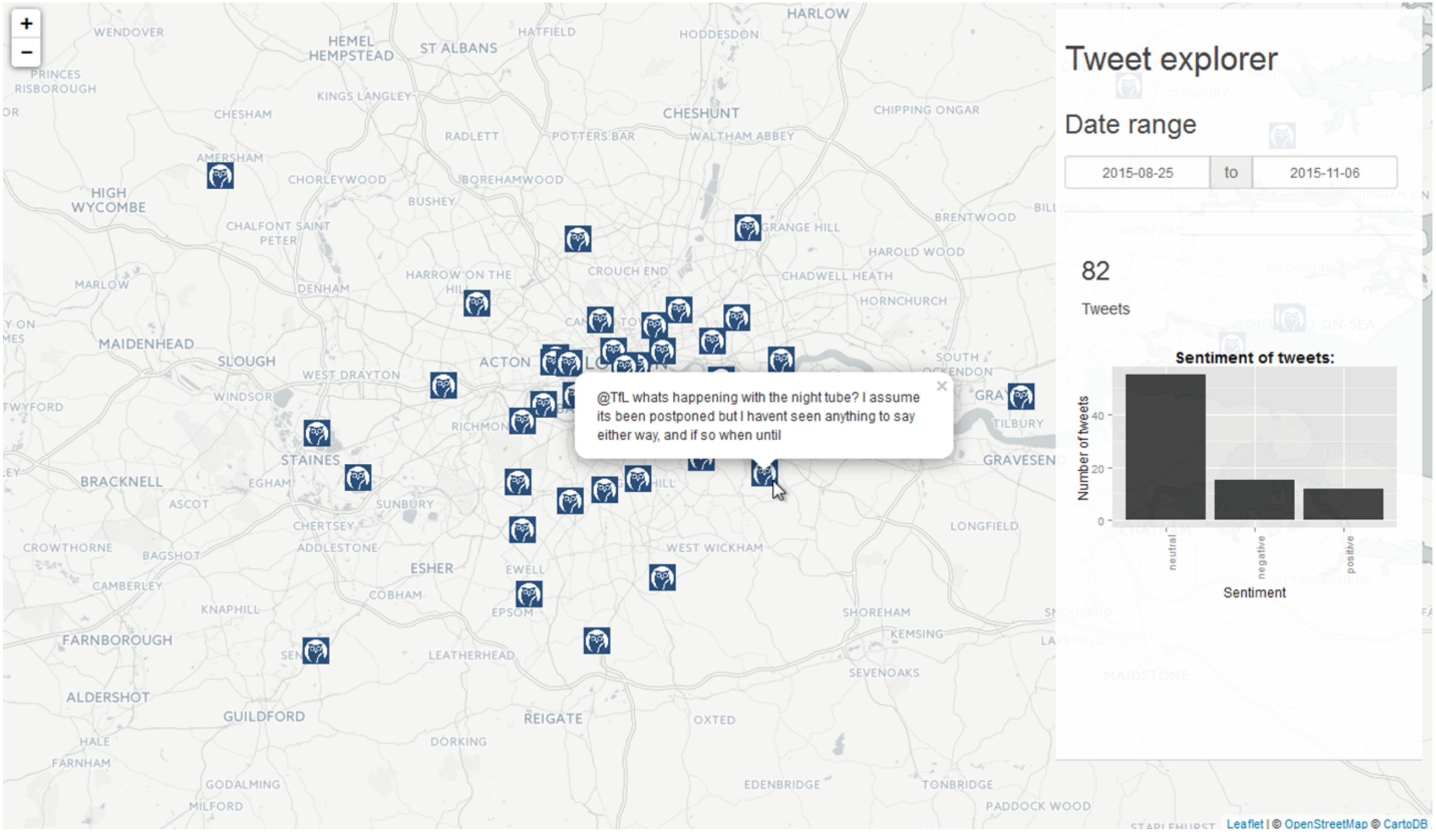
- The process of loading, cleaning, manipulating and visualising data is possible entirely within R
- Lowers barrier of entry to web development
- R bindings for JavaScript visualization libraries becoming available all the time
- Open source code encourages collaboration

Shiny app development at TfL

to estimate harm



to analyse tweets



to track time series

to monitor fare evasion

EOS Bus Enforcement Survey and DIR Viewer

Select a topic to view:

- Corridor
- Survey_map
- Survey_vs_DIR_map
- DIR_explorer

Select

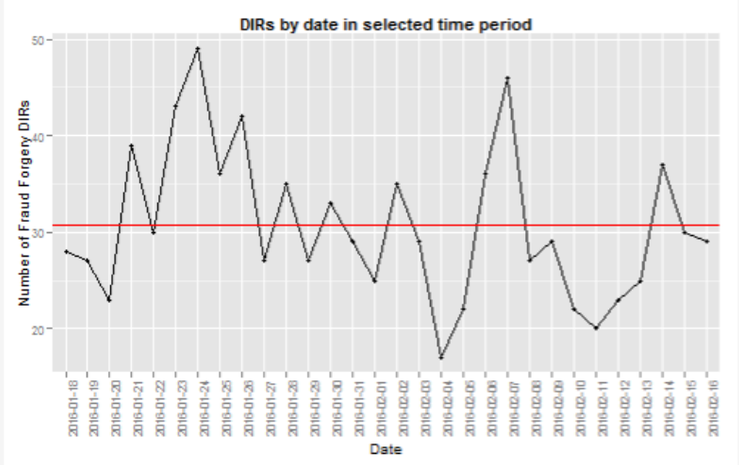
Select a date range

2016-01-18 to 2016-02-16

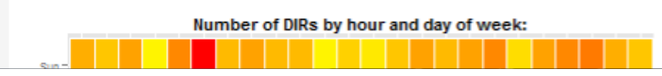
Update

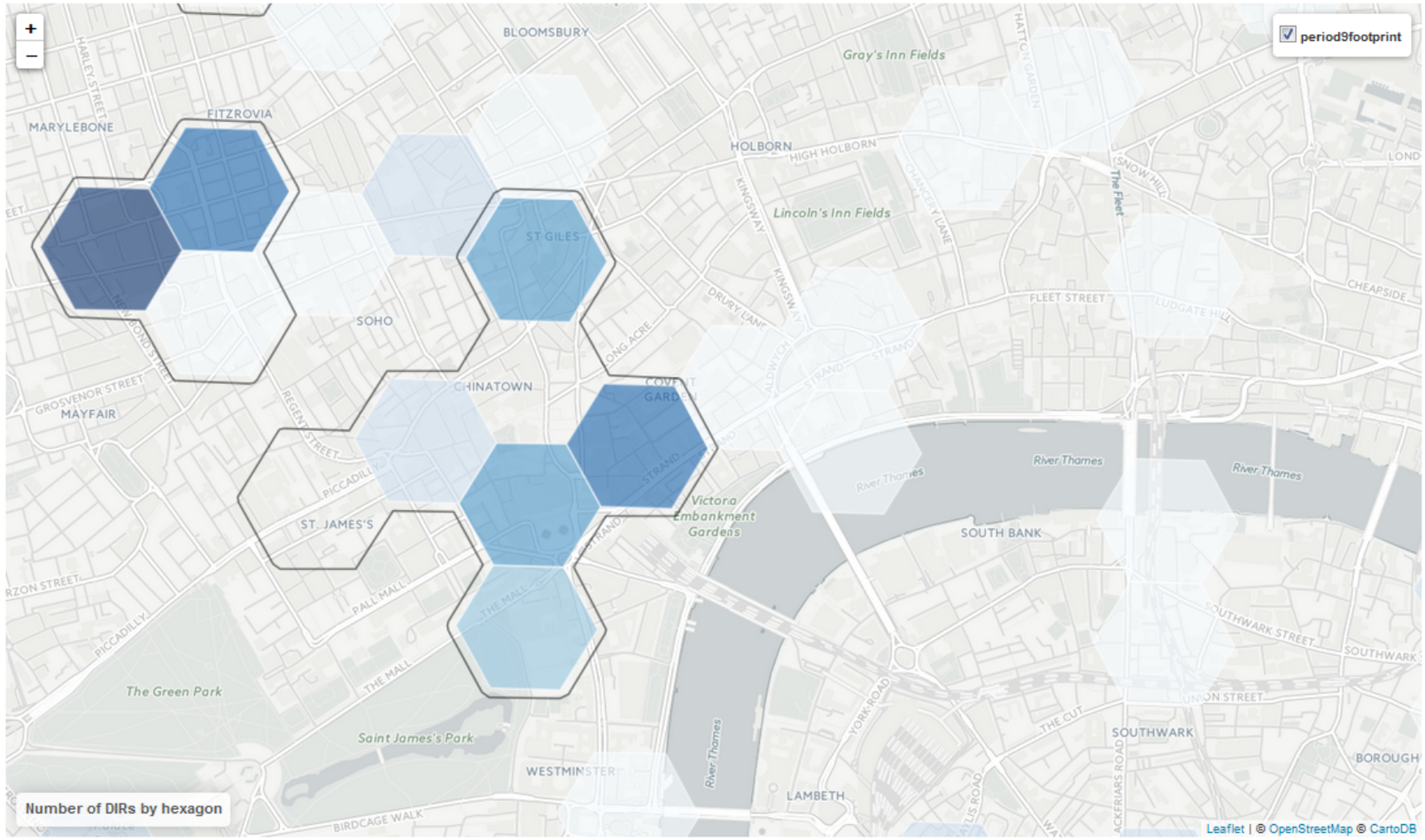
[1] "Showing stats for all of London"

Clear selection

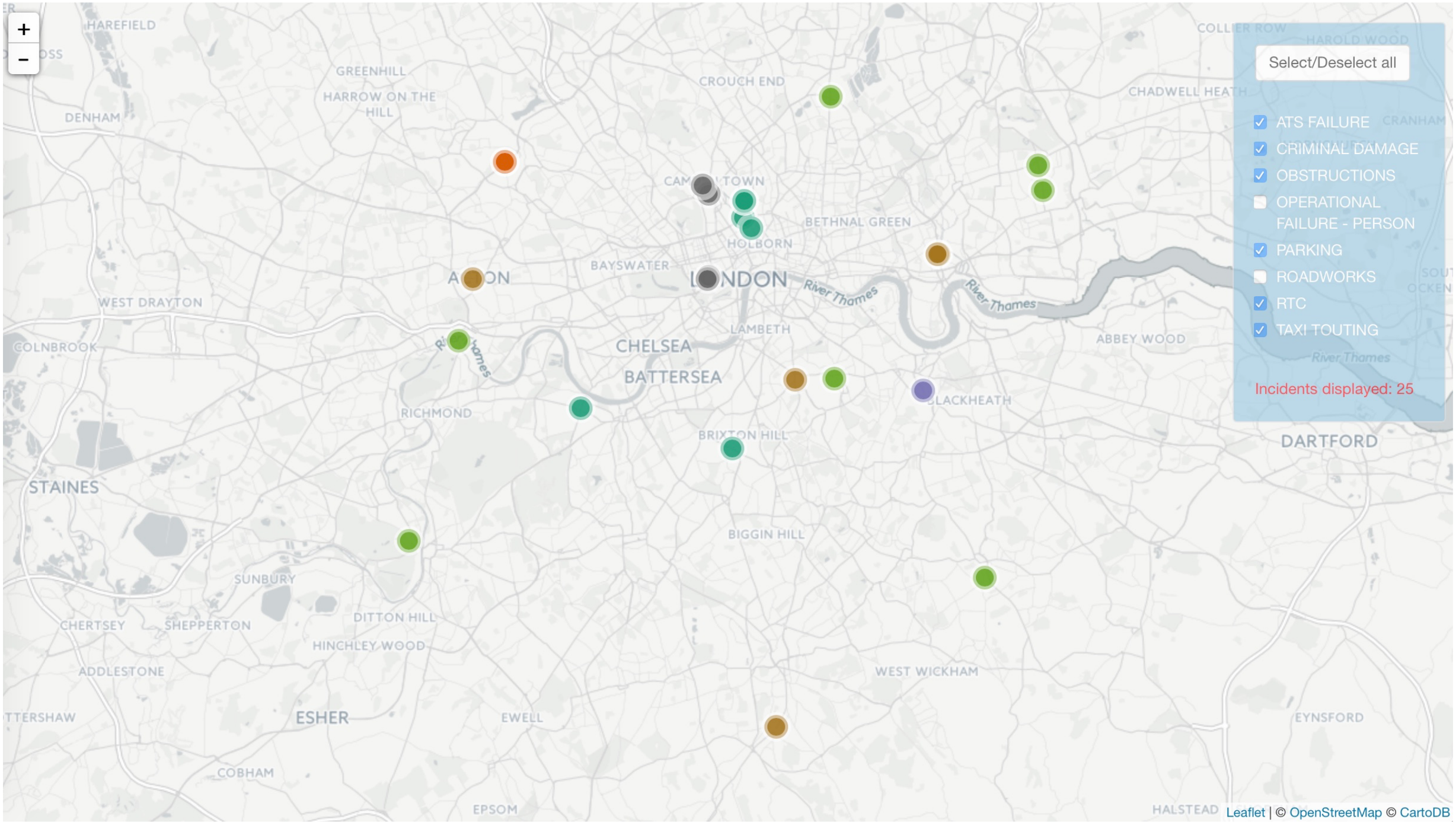


Total DIRs this time period: 920, which is down from same time period last year by 12
There were an average of 31 DIRs per day, which is down from same time period last year by 0





to interrogate databases



Hosting apps

- Run locally in an R session
- Deploy on RStudio's shinyapps.io
- Install Shiny server and host on local or cloud server

Building a Shiny app

Structure of an app

Each Shiny app has two components: a UI (web page) and a server function (live R session).

```
library(shiny)
ui <- fluidPage()
server <- function(input, output){}
shinyApp(ui, server)
```

The UI specifies the layout and user interface elements, e.g. HTML widgets like drop-downs, sliders, radio buttons etc., whilst the server specifies how to generate the output, e.g. table, plot, text.

Demo

```
library(shiny)

ui <- fluidPage()

server <- function(input, output) {}

shinyApp(ui, server)
```

open the template

```
library(shiny) ; library(rgdal) ; library(leaflet)

ui <- fluidPage()

server <- function(input, output) {}

shinyApp(ui, server)
```

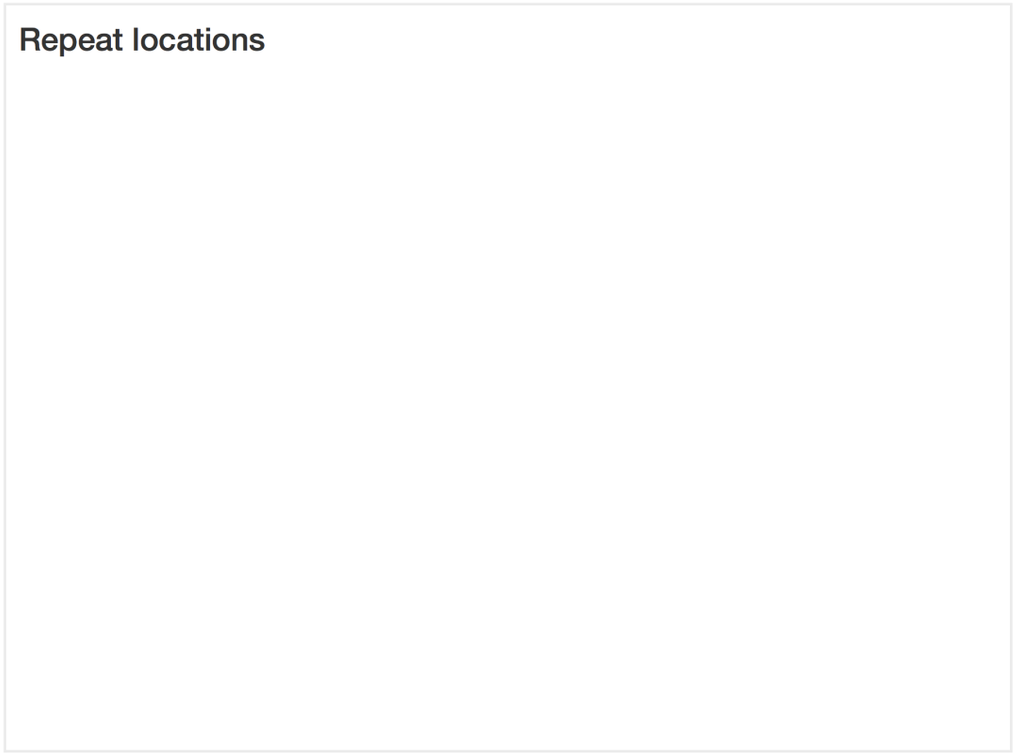
load the packages

```
library(shiny) ; library(rgdal) ; library(leaflet)

df <- read.csv("repeat_locations_Dec15.csv", header = T)
boundary <- readOGR("manchester.geojson", "OGRGeoJSON")

ui <- fluidPage()

server <- function(input, output) {}

shinyApp(ui, server)
```

read the data

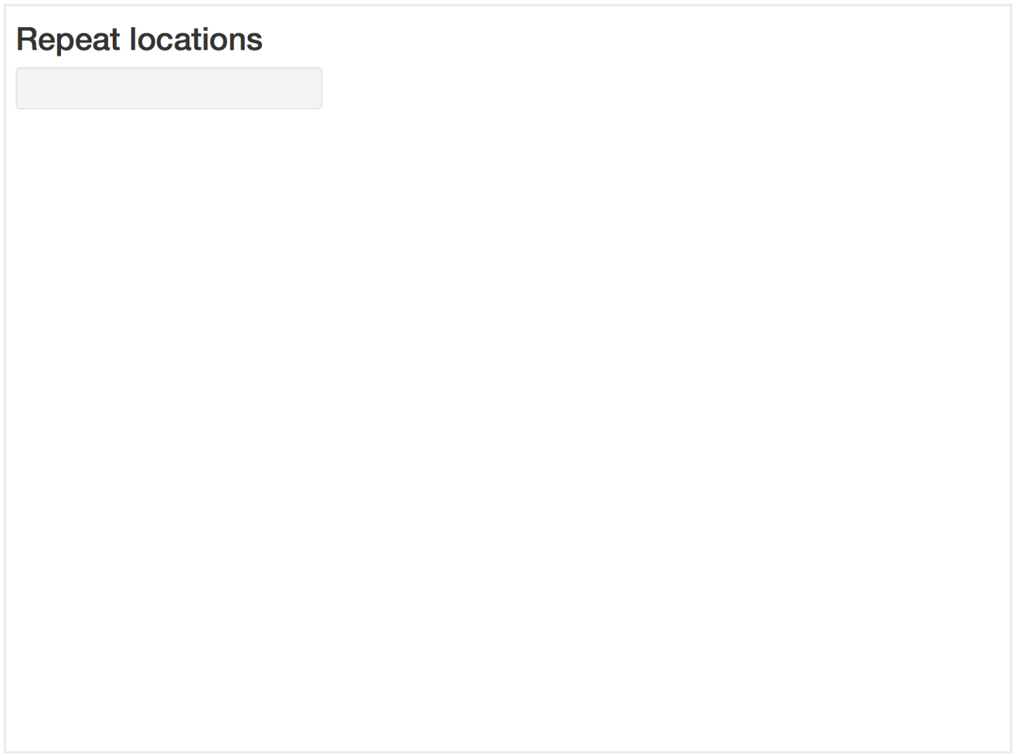
```
library(shiny) ; library(rgdal) ; library(leaflet)

df <- read.csv("repeat_locations_Dec15.csv", header = T)
boundary <- readOGR("manchester.geojson", "OGRGeoJSON")

ui <- fluidPage(titlePanel("Repeat locations")
  )

  server <- function(input, output) {}

  shinyApp(ui, server)
```

add a title



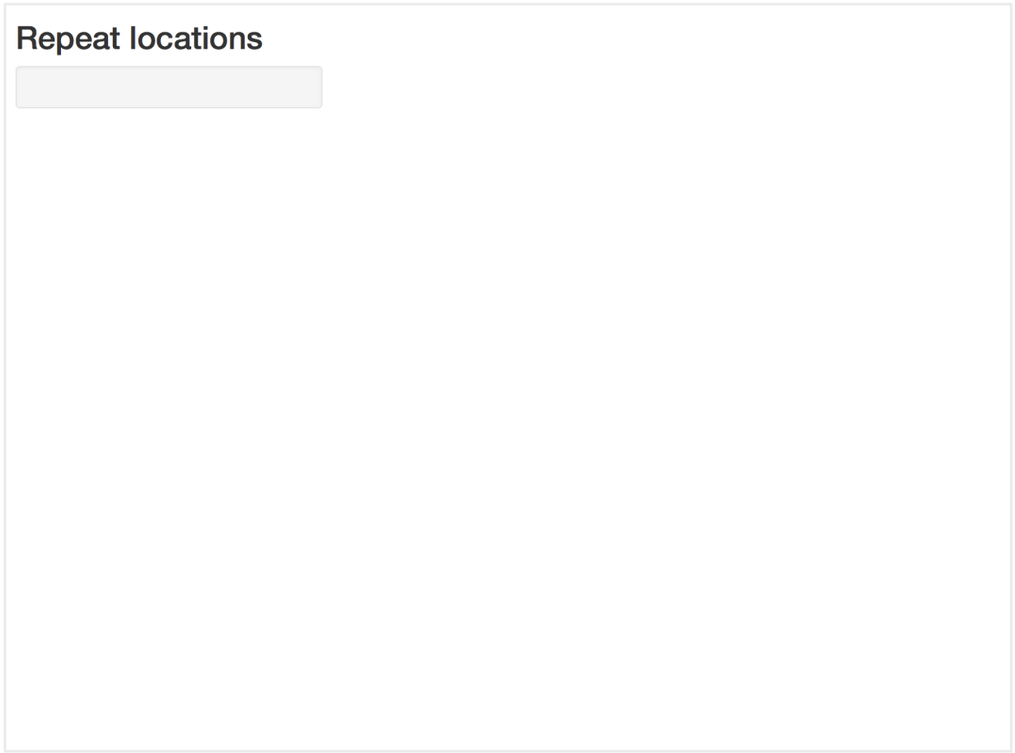
```
library(shiny) ; library(rgdal) ; library(leaflet)

df <- read.csv("repeat_locations_Dec15.csv", header = T)
boundary <- readOGR("manchester.geojson", "OGRGeoJSON")

ui <- fluidPage(titlePanel("Repeat locations"),
                sidebarLayout(
                  sidebarPanel(),
                  mainPanel()
                )
)

server <- function(input, output) {}

shinyApp(ui, server)
```

add a layout



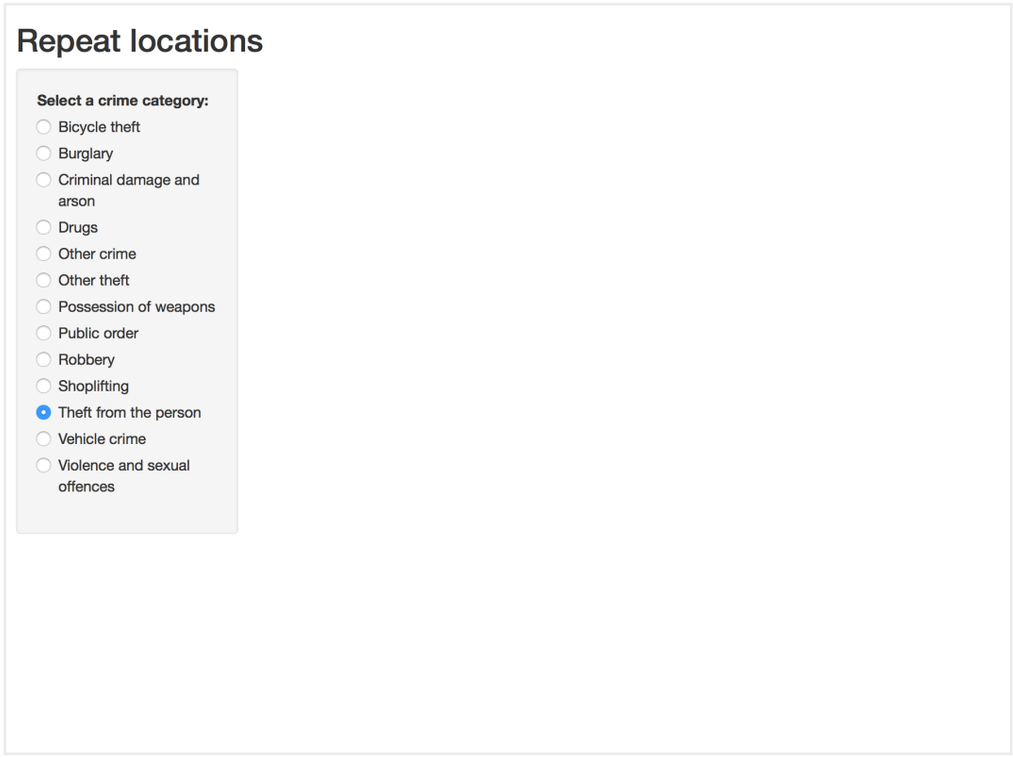
```
<div class="container-fluid">
  <h2>Repeat locations</h2>
  <div class="row">
    <div class="col-sm-4">
      <form class="well"></form>
    </div>
    <div class="col-sm-8"></div>
  </div>
</div>
```

add a layout

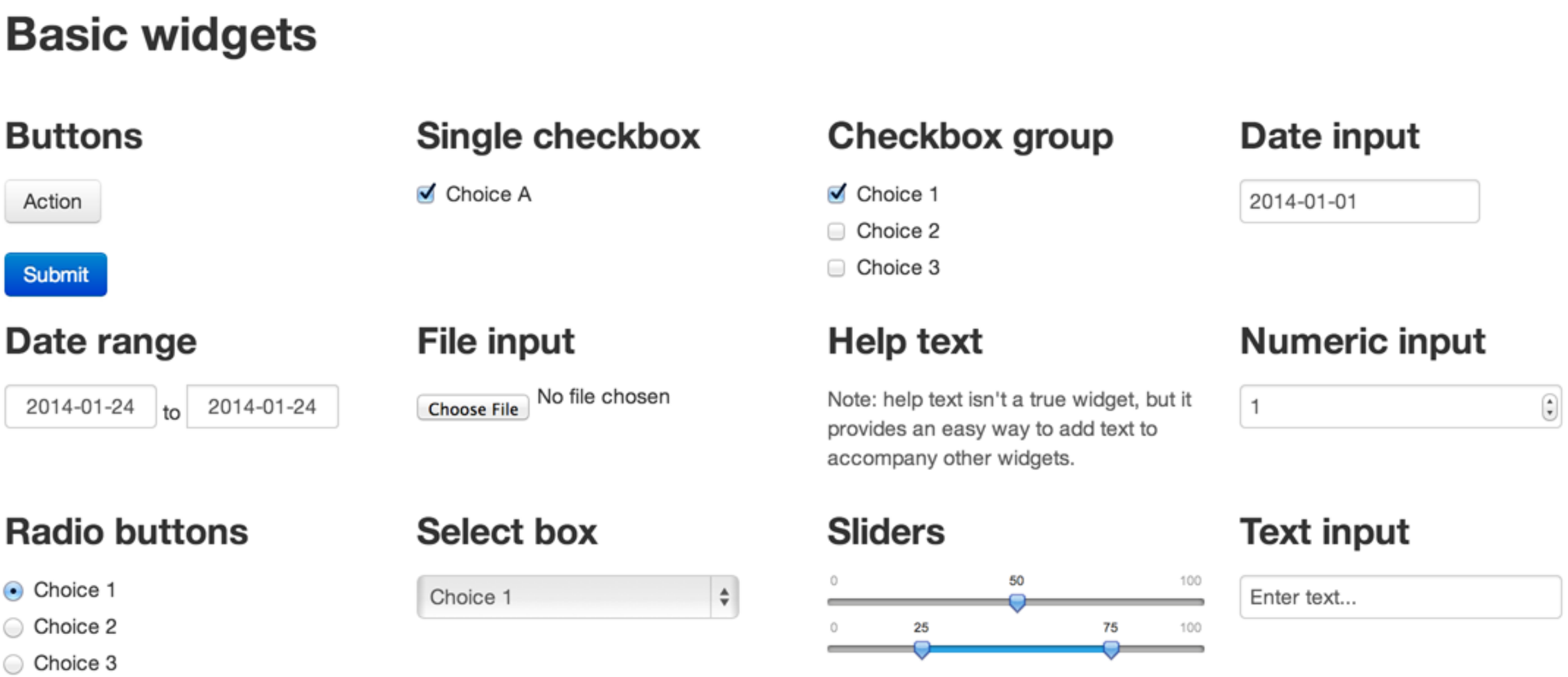


```
library(shiny) ; library(rgdal) ; library(leaflet)

df <- read.csv("repeat_locations_Dec15.csv", header = T)
boundary <- readOGR("manchester.geojson", "OGRGeoJSON")

ui <- fluidPage(titlePanel("Repeat locations"),
                sidebarLayout(
                  sidebarPanel(width = 3,
                               radioButtons(inputId = "category",
                                            label = "Select a crime category:",
                                            choices = levels(df$category),
                                            selected = "Theft from the person")),
                  mainPanel()
                  )
                )

server <- function(input, output) {}

shinyApp(ui, server)
```

add a reactive input

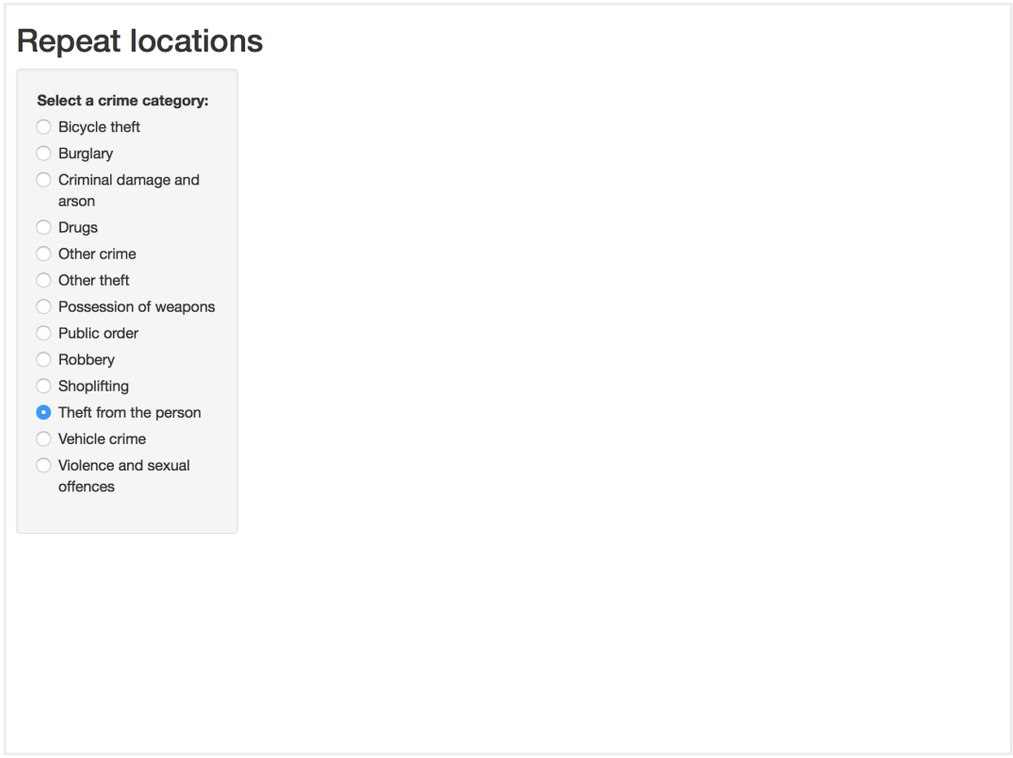


*Input() functions



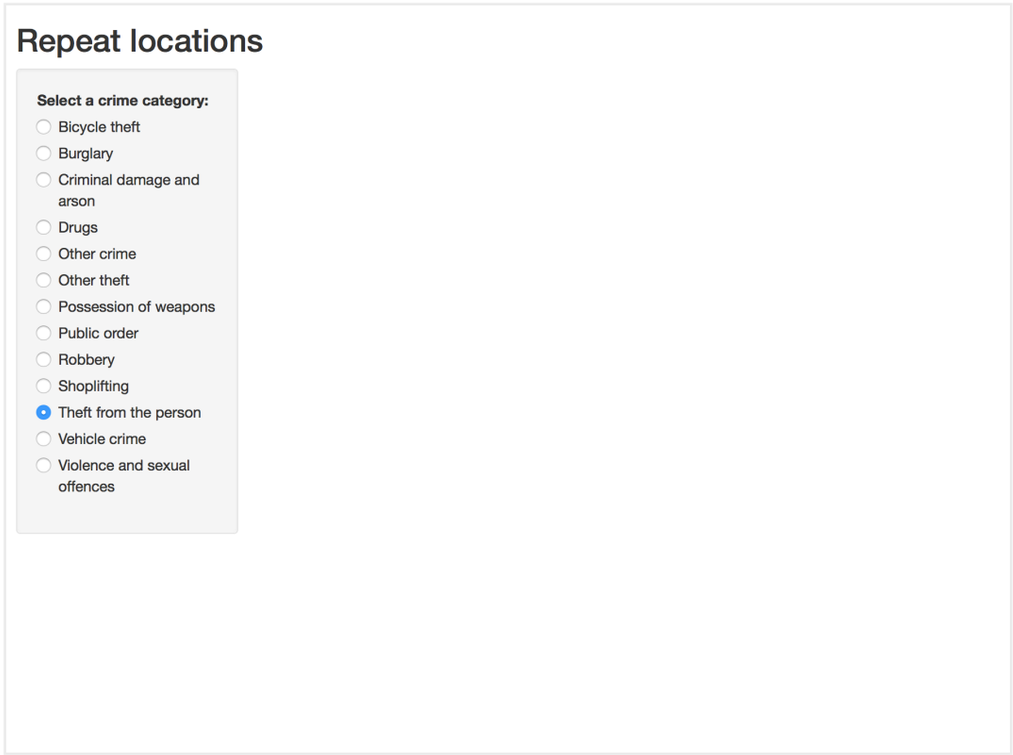
```
library(shiny) ; library(rgdal) ; library(leaflet)

df <- read.csv("repeat_locations_Dec15.csv", header = T)
boundary <- readOGR("manchester.geojson", "OGRGeoJSON")

ui <- fluidPage(titlePanel("Repeat locations"),
                sidebarLayout(
                  sidebarPanel(width = 3,
                               radioButtons(inputId = "category",
                                            label = "Select a crime category:",
                                            choices = levels(df$category),
                                            selected = "Theft from the person")),
                  mainPanel()
                  )
                )

server <- function(input, output) {

  points <- reactive({subset(df, category == input$category)})
}

shinyApp(ui, server)
```

add a reactive



```
library(shiny) ; library(rgdal) ; library(leaflet)

df <- read.csv("repeat_locations_Dec15.csv", header = T)
boundary <- readOGR("manchester.geojson", "OGRGeoJSON")

ui <- fluidPage(titlePanel("Repeat locations"),
                sidebarLayout(
                  sidebarPanel(width = 3,
                               radioButtons(inputId = "category",
                                            label = "Select a crime category:",
                                            choices = levels(df$category),
                                            selected = "Theft from the person")),
                  mainPanel(
                    leafletOutput(outputId = "map", width = "100%", height = "530")
                  )
                )

server <- function(input, output) {

  points <- reactive({subset(df, category == input$category)})
}

shinyApp(ui, server)
```

add a reactive output

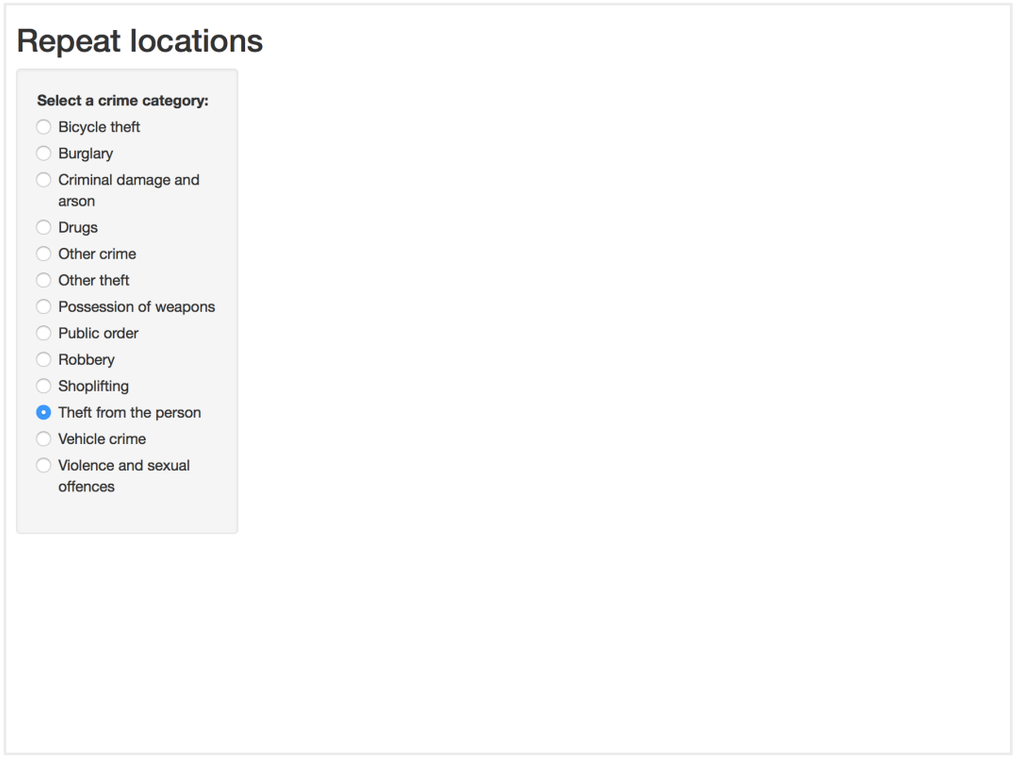


*Output() functions

*Output()	Inserts
plotOutput()	plot
tableOutput()	table
textOutput()	text
uiOutput()	a Shiny UI element
leafletOutput()	leaflet map

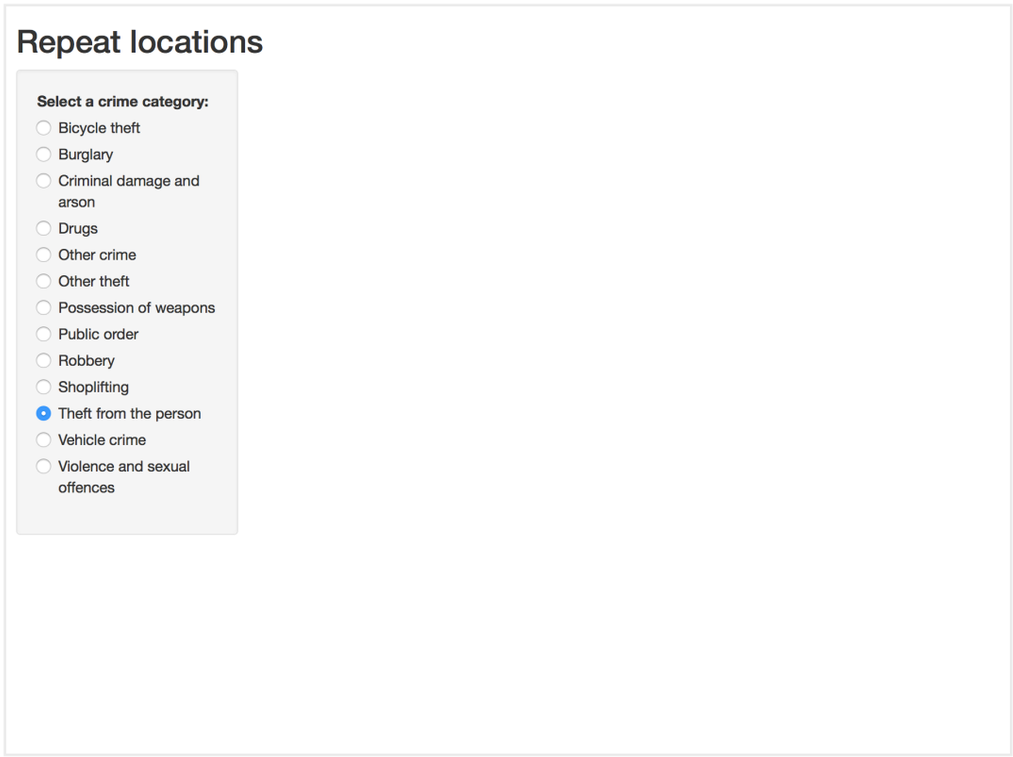
```
library(shiny) ; library(rgdal) ; library(leaflet)

df <- read.csv("repeat_locations_Dec15.csv", header = T)
boundary <- readOGR("manchester.geojson", "OGRGeoJSON")

ui <- fluidPage(titlePanel("Repeat locations"),
                sidebarLayout(
                  sidebarPanel(width = 3,
                               radioButtons(inputId = "category",
                                            label = "Select a crime category:",
                                            choices = levels(df$category),
                                            selected = "Theft from the person")),
                  mainPanel(
                    leafletOutput(outputId = "map", width = "100%", height = "530")
                  )
                )

server <- function(input, output) {

  points <- reactive({subset(df, category == input$category)})

  output$map <-
}

shinyApp(ui, server)
```

save output



```
library(shiny) ; library(rgdal) ; library(leaflet)

df <- read.csv("repeat_locations_Dec15.csv", header = T)
boundary <- readOGR("manchester.geojson", "OGRGeoJSON")

ui <- fluidPage(titlePanel("Repeat locations"),
                sidebarLayout(
                  sidebarPanel(width = 3,
                               radioButtons(inputId = "category",
                                            label = "Select a crime category:",
                                            choices = levels(df$category),
                                            selected = "Theft from the person")),
                  mainPanel(
                    leafletOutput(outputId = "map", width = "100%", height = "530")
                  )
                )

server <- function(input, output) {

  points <- reactive({subset(df, category == input$category)})

    output$map <-
    renderLeaflet({
     })
}}

shinyApp(ui, server)
```

build reactive output

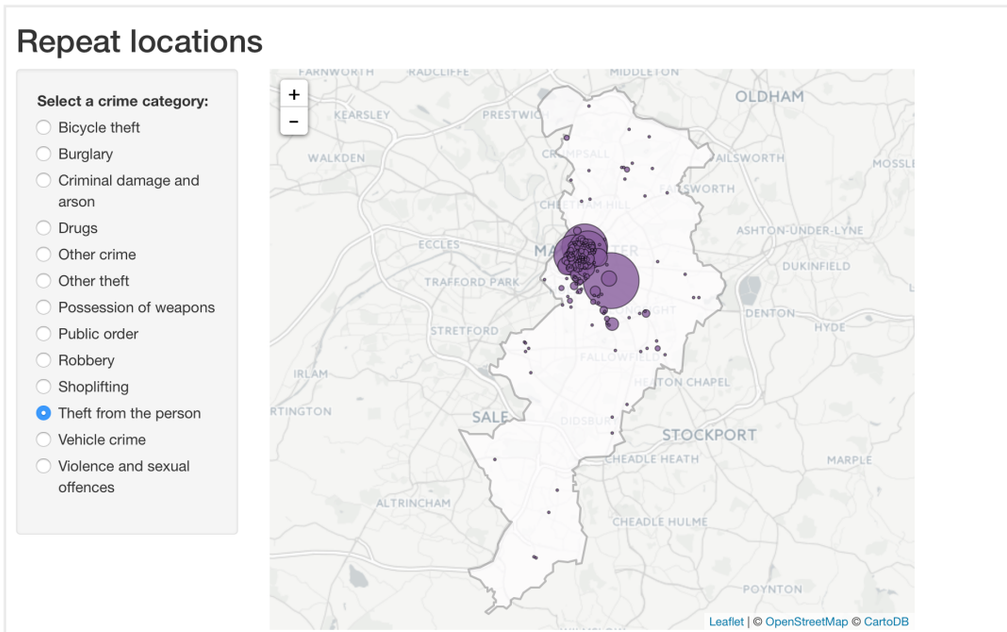


render*() functions

render*()	*Output()
renderPlot()	plotOutput()
renderTable()	tableOutput()
renderText()	textOutput()
renderUI()	uiOutput()
renderLeaflet()	leafletOutput()

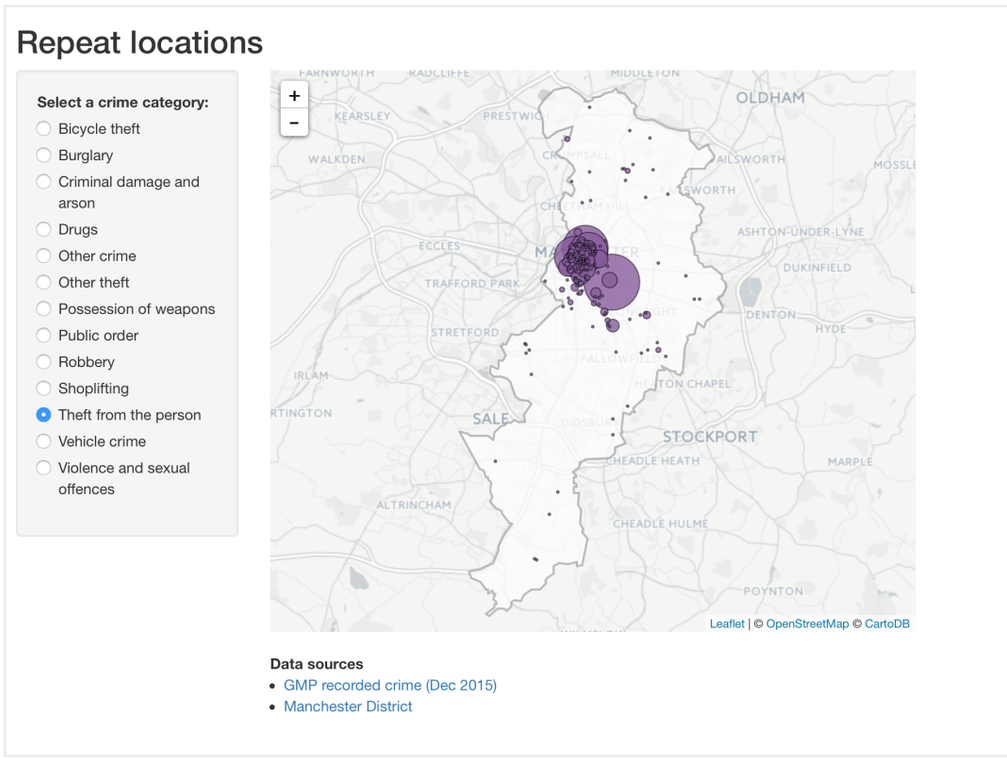
```r
library(shiny) ; library(rgdal) ; library(leaflet)

df <- read.csv("repeat_locations_Dec15.csv", header = T)
boundary <- readOGR("manchester.geojson", "OGRGeoJSON")

ui <- fluidPage(titlePanel("Repeat locations"),
                sidebarLayout(
                  sidebarPanel(width = 3,
                               radioButtons(inputId = "category",
                                            label = "Select a crime category:",
                                            choices = levels(df$category),
                                            selected = "Theft from the person")),
                  mainPanel(
                    leafletOutput(outputId = "map", width = "100%", height = "530")
                  )
                )

server <- function(input, output) {

  points <- reactive({subset(df, category == input$category)})

  output$map <-
    renderLeaflet({
      popup <- paste0("<strong>Category: </strong>", points()$category,
                      "<br><strong>Location: </strong>", points()$location,
                      "<br><strong>Frequency: </strong>", points()$n)
      factpal <- colorFactor("Paired", points()$category)
      leaflet() %>%
        addProviderTiles("CartoDB.Positron") %>%
        addPolygons(data = boundary,
                    fillColor = "white", fillOpacity = 0.7,
                    color = "grey", weight = 2) %>%
        addCircleMarkers(data = points(), ~long, ~lat,
                         stroke = TRUE, color = "black", weight = 1,
                         fillColor = ~factpal(category), fillOpacity = 0.8,
                         radius = ~n*1.2, popup = popup)})
}

shinyApp(ui, server)
```

access input values



```
library(shiny) ; library(rgdal) ; library(leaflet)

df <- read.csv("repeat_locations_Dec15.csv", header = T)
boundary <- readOGR("manchester.geojson", "OGRGeoJSON")

ui <- fluidPage(titlePanel("Repeat locations"),
                sidebarLayout(
                  sidebarPanel(width = 3,
                               radioButtons(inputId = "category",
                                            label = "Select a crime category:",
                                            choices = levels(df$category),
                                            selected = "Theft from the person")),
                  mainPanel(
                    leafletOutput(outputId = "map", width = "100%", height = "530"),
                    br(),
                    tags$div(class = "header", checked = NA,
                             tags$strong("Data sources"),
                             tags$li(tags$a(href="https://data.police.uk",
                                            "GMP recorded crime (Dec 2015)")),
                             tags$li(tags$a(href="https://data.gov.uk/data/map-based-search",
                                            "Manchester District"))))
                  )
                )

server <- function(input, output) {

  points <- reactive({subset(df, category == input$category)})

  output$map <-
    renderLeaflet({
      popup <- paste0("<strong>Category: </strong>", points()$category,
                      "<br><strong>Location: </strong>", points()$location,
                      "<br><strong>Frequency: </strong>", points()$n)
      factpal <- colorFactor("Paired", points()$category)
      leaflet() %>%
        addProviderTiles("CartoDB.Positron") %>%
        addPolygons(data = boundary,
                    fillColor = "white", fillOpacity = 0.7,
                    color = "grey", weight = 2) %>%
        addCircleMarkers(data = points(), ~long, ~lat,
                         stroke = TRUE, color = "black", weight = 1,
                         fillColor = ~factpal(category), fillOpacity = 0.8,
                         radius = ~n*1.2, popup = popup)})
}

shinyApp(ui, server)
```

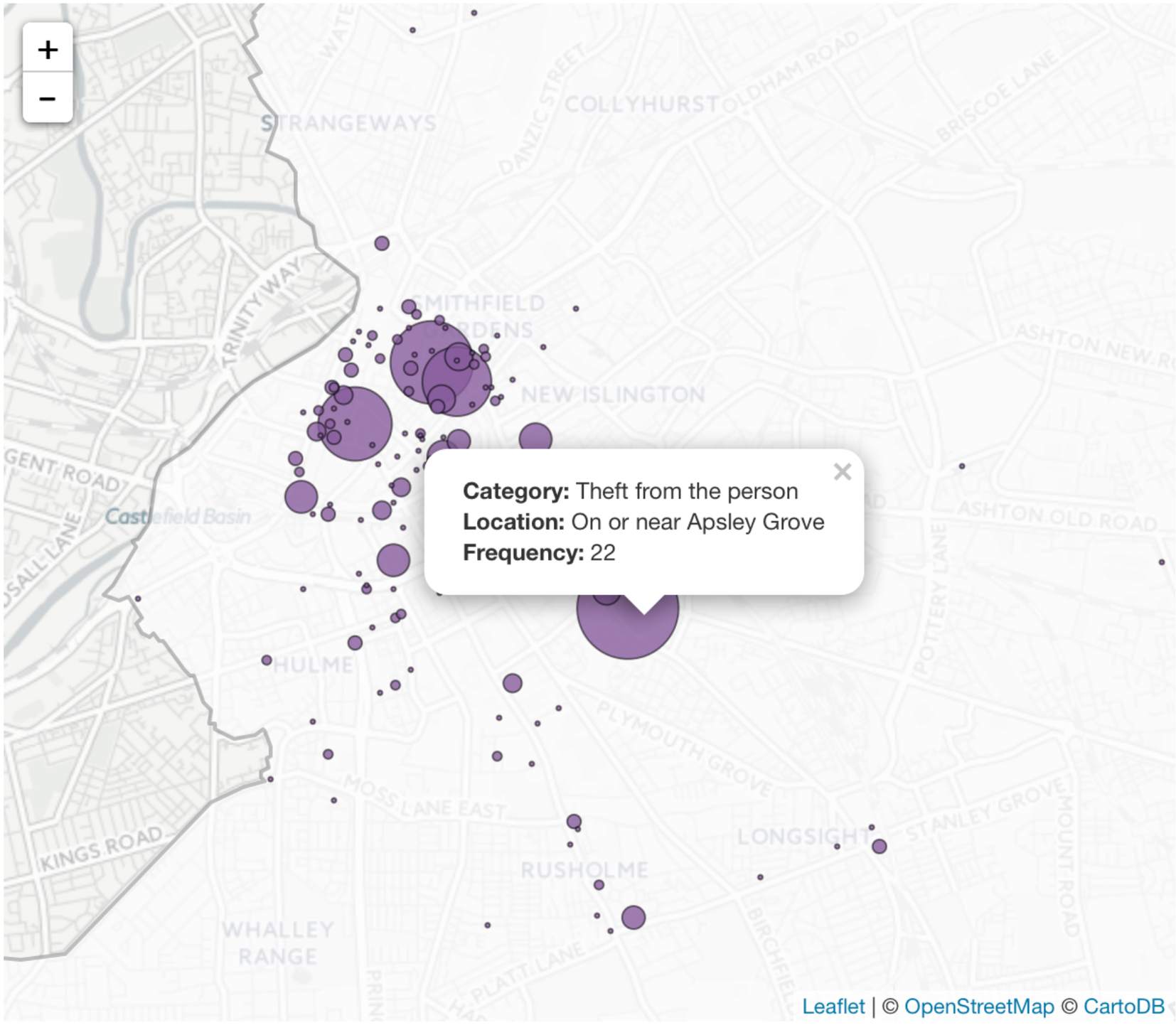
add HTML elements



Repeat locations

Select a crime category:

- ○ Bicycle theft
- ○ Burglary
- ○ Criminal damage and arson
- ○ Drugs
- ○ Other crime
- ○ Other theft
- ○ Possession of weapons
- ○ Public order
- ○ Robbery
- ○ Shoplifting
- ◉ Theft from the person
- ○ Vehicle crime
- ○ Violence and sexual offences



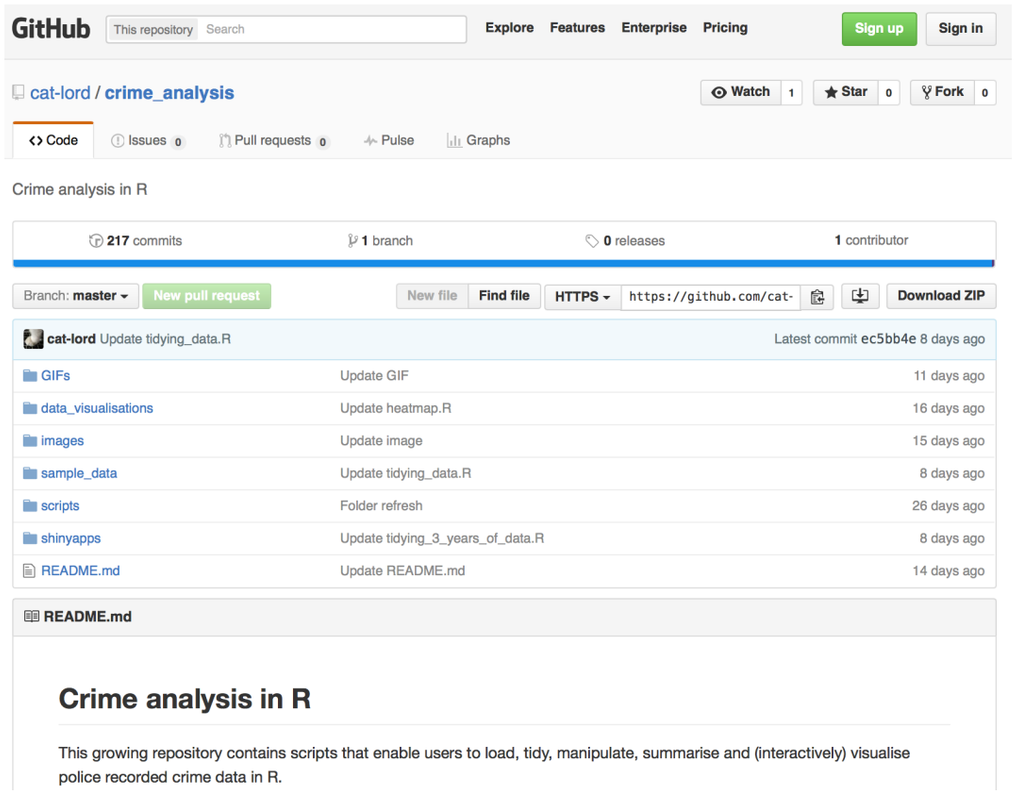
Data sources

- GMP recorded crime (Dec 2015)
- Manchester District

Learning Shiny

- RStudio's online Shiny tutorials, webinar and cheatsheet
- Dean Attali's interactive tutorial
- The Shiny Google discussion group

and crime analysis ...



Materials for crime analysis in R including sample crime data, scripts and Shiny apps are available in a repository on my GitHub page.

Acknowledgements

- RStudio for the Shiny R package and the slide template
- Andy Bartlett, Sead Taslaman and my colleagues in TfL's R User Group for promoting R and Shiny.

Questions?