Route Analysis using R and Google Maps

Setup

Open up R Studio (click **Start** > **All Programs** > **RStudio** > **RStudio** or double-click the icon on the desktop).

You don't need any data to run this tutorial - we generate it all online. However, it is good practice to set-up a new project for our R routing experiments. Do this in RStudio now (if you're new to RStudio, take a look around and you'll soon see how).

All but one of the libraries (**stplanr**) we will use in this package can be downloaded from CRAN, R's online store of packages.

- **ggmap**: a package for plotting geographical data
- **sp**: a package defining spatial objects in R
- **devtools**: a package we'll use to install **stplanr**!

All the packages can be installed by clicking **Tools** > **Install Packages**. Alternatively you can install these packages with the following commands:

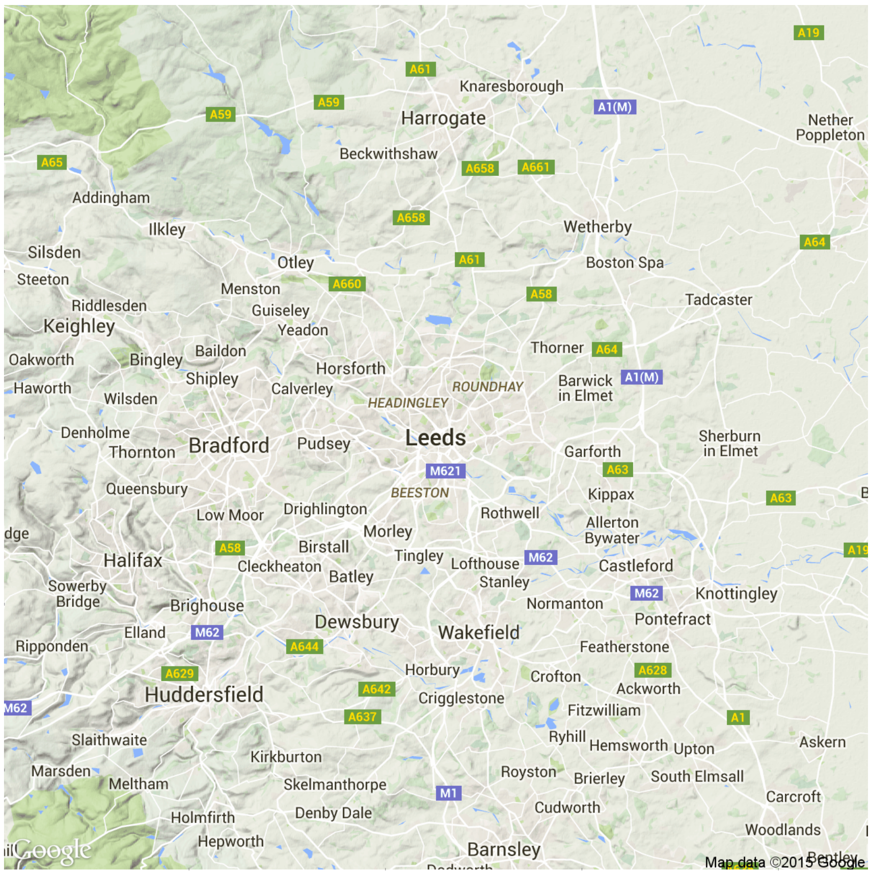
```r
pkgs <- c("ggmap", "sp", "devtools") # create an object for the packages
install.packages(pkgs) # install the packages - requires the internet
```

ggmap allows for quick and easy map creation and an interface to Google Maps within R. First, we need to load the library, so type into the console:

```r
library(ggmap) # use install.packages("ggmap") if not already installed
```

This will also load various other libraries and you will probably see information messages. We can then create a quick map of Leeds with the `qmap()` function. You should see the map appear in the 'Plots' window to the right of the R Studio window.

```r
qmap('Leeds')
```



We can also search for anything we would search for in the Google Maps search box - e.g. different locations, or postcodes. For example, try:

```r
qmap('L69 3BX')
```

Try different postcodes or locations to search for, and see what happens. You will see that the scale of the map stays the same - this is fine for Leeds, but of limited use for somewhere smaller (like Norwich) or larger (like Ireland). The default zoom is level 10, but we can change this to something more useful. Higher numbers zoom in more (i.e. show a smaller area). Try:

```r
qmap('LS2 9JT', zoom = 17)
```

This shows us the map centred on the Parkinson building (although it isn't marked on the map).

As well as the maps layer, we can access the satellite and hybrid map types from Google Maps:

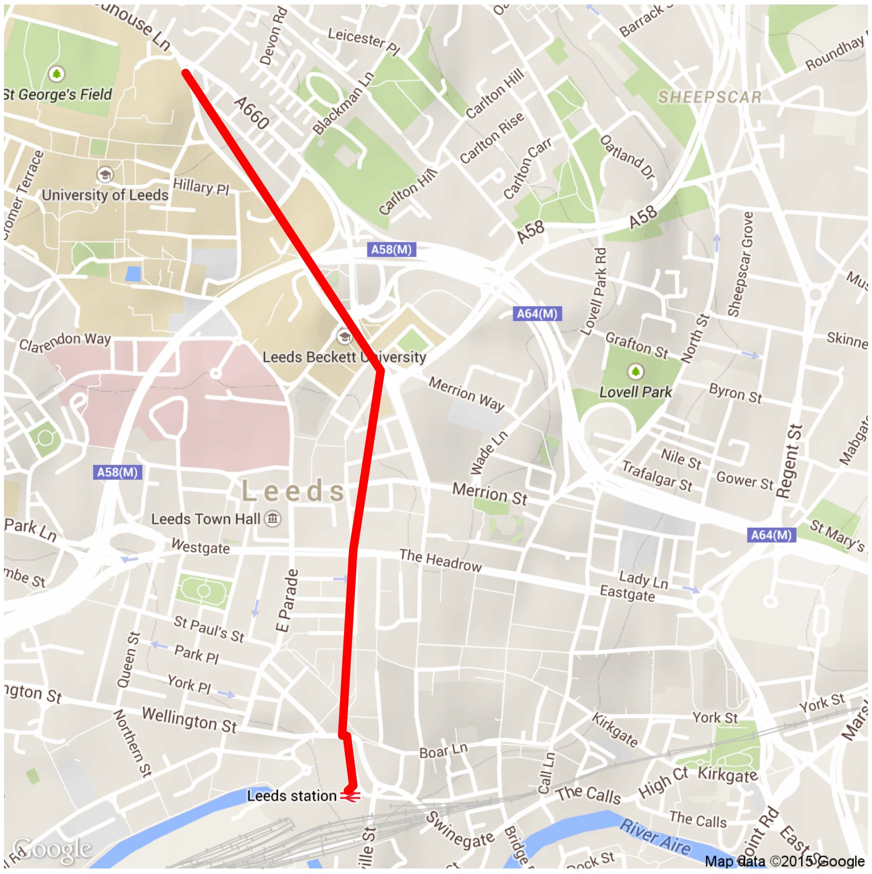
```r
qmap('LS2 9JT', zoom = 17, maptype = 'satellite')
```

One useful shortcut when using R Studio (or R on its own) is that pressing the 'up' key on the keyboard will show your previous command. You can then edit this and press return to run it, which avoids the need to type the whole command out again! Try it with the command below:

```r
qmap('LS2 9JT', zoom = 17, maptype = 'hybrid')
```

Routing

Just as you can get directions from the Google Maps website, you can also access the same tools through the API. The code below passes an origin and destination to the routing command and then shows the route on a map. Try running this code, and then try changing to origin and/or destination and try running it again. You may need to adjust the location of the map to show the whole route.



With a bit of extra R code, we can create a data frame of different origin and destination pairs, and then write the code to get R to look through the API and produce a route for each of these.

First of all, add the code to get R to save the map as a PDF. Run this code, and R will save the map as a PDF file in your working folder. (Run `getwd()` if you are not sure where your working folder is).

```r
#save as PDF
pdf(file="image.pdf")
  from <- 'Leeds station, New Station Street, Leeds LS1 5DL, United Kingdom'
  to <- 'LS2 9JT'
  route_df <- route(from, to, structure = 'route', mode = 'walking')
  qmap('Merrion Centre', zoom = 15) +
    geom_path(
      aes(x = lon, y = lat),  colour = 'red', size = 1.5,
      data = route_df, lineend = 'round')
#stop saving as PDF
dev.off()
```

Then we can add a data frame and a loop, to run the code for each entry.

```r
#setup variable with list of origins
  origins <- c("Leeds station, New Station Street, Leeds LS1 5DL","Leeds","Manchester",
               "Manchester Oxford Road")
  destinations <- c("LS2 9JT","Manchester","Liverpool","Manchester Picadilly")
  map_center <- c("Merrion Centre","Marsden, UK","Warrington","Sackville St, Manchester")
  zoom_level <- c(15,10,10,16)
#combine origins and destinations to create a data frame
  df_orig_dest <- data.frame(origins,destinations,map_center,zoom_level)
#loop through for each map
for (i in 1:length(df_orig_dest)) {
  #setup file name
  filename <- paste0("route_",df_orig_dest[i,1],"_to_",df_orig_dest[i,2],".pdf")
  #save as PDF and set file name
  pdf(file=filename)
    from <- as.character(df_orig_dest[i,1]) #check whether this is needed
    to <- as.character(df_orig_dest[i,2])
    route_df <- route(from, to, structure = 'route', mode = 'walking')
    print(qmap(as.character(df_orig_dest[i,3]), zoom = df_orig_dest[i,4]) +
      geom_path(
        aes(x = lon, y = lat),  colour = 'red', size = 1.5,
        data = route_df, lineend = 'round'))
  #stop saving as PDF
  dev.off()
#end loop
}
```

Try experimenting with different parts of the code, to, for example, change the colour for each route, or add more entries. You can also save the maps as images - for example, type `?jpeg` into the console to see how to get the map as a JPEG.

Route Analysis

We have plotted a few maps now, but we can also do some analysis of the data as well. The variable `route_df` contains various pieces of information about the route we have plotted. Run `head(route_df)` to see the first six rows of this variable. The output may be slightly different if you have been experimenting with different routes.

```r
head(route_df) #show the first six rows of route_df
```

We can add up the values from specific columns, using `colSums` to extract the total distance or time, for example.

```r
colSums(route_df, na.rm = TRUE) #calc sum for each column, omitting missing values
```

Some of these values are just gibberish and mean nothing (e.g. `lat` and `lon`) but others are quite useful - total time or distance, for example. Distance is particularly useful, and is one that I used in my Home to School travel work.

Routes produced by Google Maps are useful, provide accurate indications of route distance and reasonable estimates of times. You can even specify what mode of transport through the *argument* `mode`. (See what happens when you add `, mode = "bicycling"`: `walking` and `driving` are the alternatives.) However, the plotted routes do not follow the travel network, simply snapping to key *nodes* where the road changes.

Routing with CycleStreets.net

To overcome this problem, we can use alternative route finding services such as graphhopper and osrm. CycleStreets is a routing service designed by cyclists for cyclists that we will use to illustrate routing using other external services in R. It has various advantages over the `route()` function (and some disadvantages) described above. You need a CycleStreets.net 'API key' for it to work.

For a project funded by the UK's Department for Transport (DfT), an R interface to the CycleStreets.net routing service has been created. This is included in the R package **stplanr**. Let's download this package to reproduce the routes using CycleStreet.net.

```r
library(sp) # enables spatial data classes
library(devtools) # to install packages from github
install_github("robinlovelace/stplanr") # install the stplanr package
library(stplanr)
```

```r
cckey <- "79769a3a2d2a0caf" # add your API key
```

Convert the origin destination matrix into straight lines

To convert the text string origins and destinations into routes, we first must convert them into objects of the `SpatialLinesDataFrame` data class:

```r
# Extract the attribute data
f <- df_orig_dest[-3,]
# Set up coordinates
coords <- geocode(c(origins, destinations))
# Create SpatialLinesDataFrame
cents <- SpatialPointsDataFrame(coords = as.matrix(coords),
  data = data.frame(code = c(origins, destinations)))
rlines <- gFlow2line(flow = f, zones = cents)
# Plot - uncomment to see result (delete the #)
# plot(rlines[1,], col = c("green", "red", "blue", "black"))
```

We have successfully saved our 'desire lines'. Now let's convert them to navigable cycle routes and plot them:

```r
rquiet <- gLines2CyclePath(l = rlines, plan = "quietest")
plot(rquiet[1,]) # route from Leeds station to Leeds University (North - South)
plot(rquiet[2,]) # route from Leeds to Manchester!
```

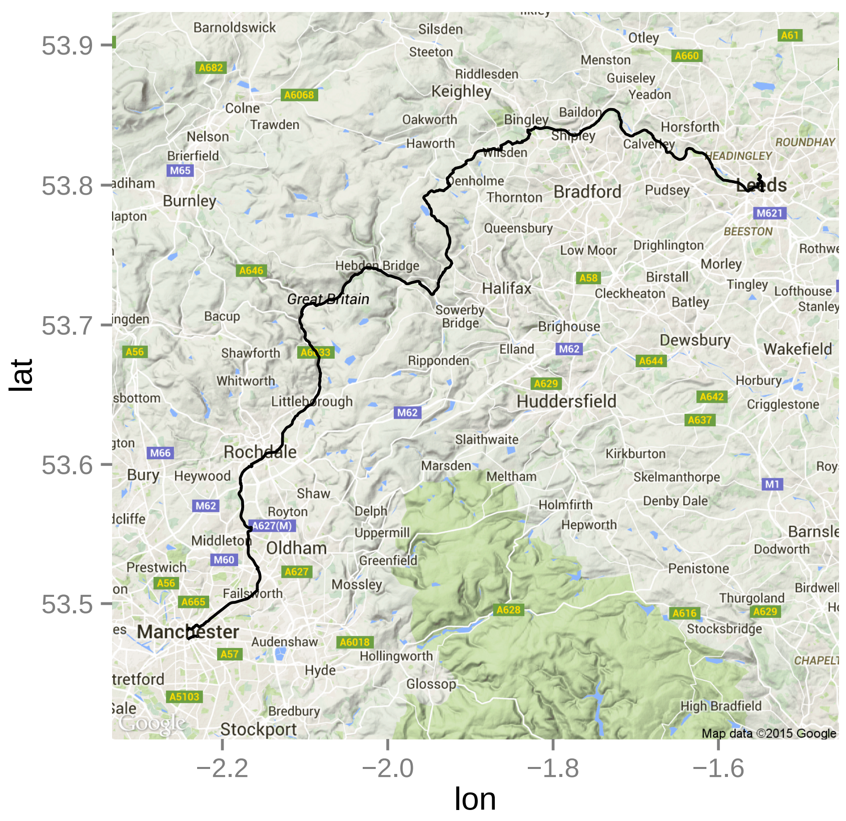
To see the quietest cycling route from Manchester to Leeds we can combine our **ggplot** skills with the new R object using fortify:

```r
rquiet_f <- fortify(rquiet)
bb <- make_bbox(lon = long, lat = lat, data = rquiet_f)
m1 <- ggmap(ggmap = get_map(location = bb))
```

```
## Warning: bounding box given to google - spatial extent only approximate.

## converting bounding box to center/zoom specification. (experimental)
## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=53.664081,-1.894526&zoom=10&size=
```

```r
m1 + geom_path(aes(x = long, y = lat, group = group), data = rquiet_f)
```



If you don't understand all the steps used in the above code block do not despair: very few people do! All we can recommend is you continue your 'journey' in R for transport applications by checking out the references listed below.

References

Bearman, N. (2015). Introduction to Using R for Spatial Analysis. Retrieved from http://www.nickbearman.me.uk/2015/01/introduction-to-using-r-for-spatial-analysis/

Kahle, D. and Wickham, H. ggmap: Spatial Visualization with ggplot2. The R Journal, 5(1), 144-161. URL: http://journal.r-project.org/archive/2013-1/kahle-wickham.pdf

Lovelace, R., & Cheshire, J. (2014). Introduction to visualising spatial data in R. National Centre for Research Methods Working Papers, 14(03). Retrieved from https://github.com/Robinlovelace/Creating-maps-in-R

Further Information

This PDF was created on 11th April 2015 by Nick Bearman and Robin Lovelace for GIS for Transport Applications workshop (https://github.com/Robinlovelace/GIS4TA) at GISRUK 2015. It is available at https://github.com/nickbearman/transport-workshop.