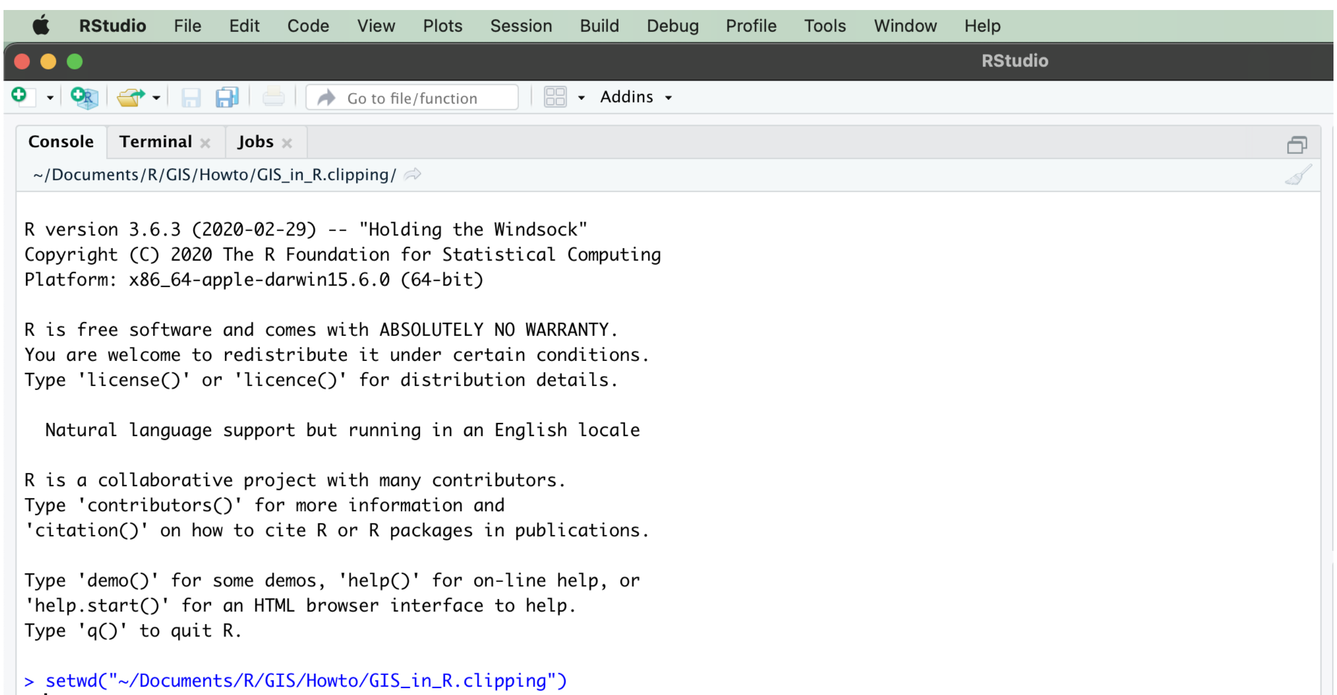
NYU DataServices

How to clip shapefile features to a study area in R

GIS in R: Clipping Shapefile

Clipping shapefile allows you to select certain features and attributes from a layer based on spatial extraction. You can use the sf package's `st_read()` function to read the shapefile you wish to clip and the shapefile you will use to restrict the original shapefile into R, and then convert the coordinate reference systems of the shapefiles to an appropriate one for the geography of your data by using the `st_transform()` function, and then clip the original spatial layer based on the other layer using `st_intersection()`. For example, you may be a transportation analyst interested in studying the distribution of subway entrances in an area. You might seek to answer the following questions: Are there enough subway entrances and are they equally distributed? Where can/should a new subway entrance be built? You now have a map of subway entrances throughout New York City, but suppose you are focused on Manhattan, and don't need to see the subway entrances in the other four boroughs. You can use the `st_intersection()` function to create a shapefile of subway entrances restricted to the Manhattan borough boundary. Here is the data for this exercise.

1. Set your working directory to the folder where your project locates using `setwd()`. Alternatively, you can also set the working directory using the dropdown menu **Session** in RStudio. Select the **Choose Directory** and then click the project folder. After you've set the working directory, you may verify it by calling the `getwd()` function.



2. Load the following packages in R. If you do not already have the packages installed, be sure to install them using command `install.packages()` before loading them.

```
library(sf)
library(tmap)
```

3. Use the `st_read()` function to read the shapefile you wish to clip and the shapefile you will use to clip the original layer into R. Optionally, you can use the mapping and spatial data visualization package called tmap to generate a simple outline of the shapefiles. The `tm_shape()` function specifies the file you wish to map, while the `tm_dots()` or `tm_polygons()` function tells tmap to draw the points or polygons that comprise the shapefiles. The `tmap_arrange()` function arrange multiple tmap objects in a grid layout. Note it is also possible to use different ways to call forth a visual representation of the shapefiles (e.g. by using ggplot).

```
NYC_subway_shapefile <- st_read("Subway_entrances.shp")
```

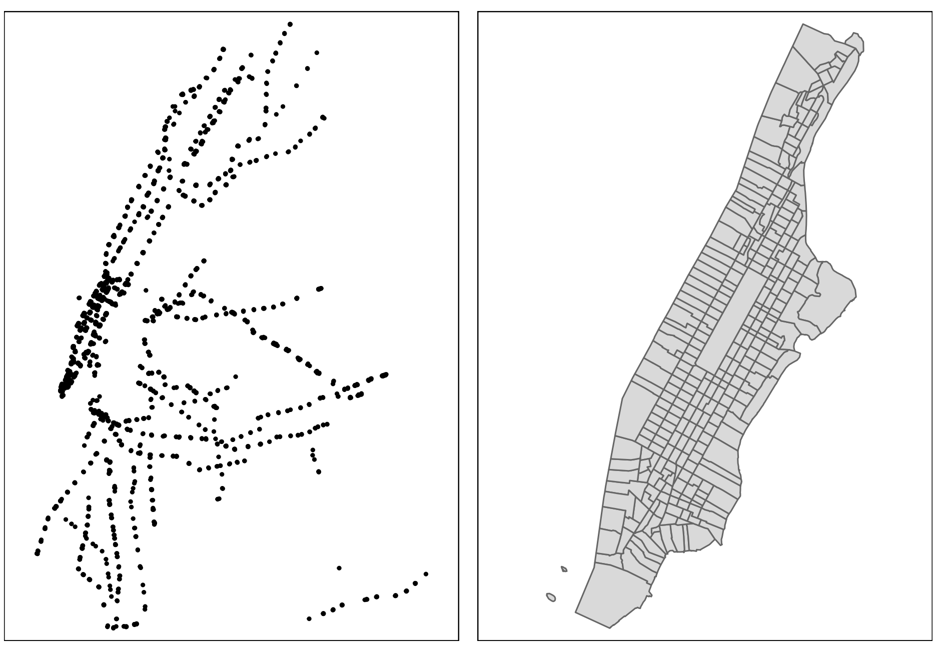
Reading layer `subway_entrances' from data source`/Volumes/GoogleDrive/My Drive/subway_entrances.shp' using driver 'ESRI Shapefile' Simple feature collection with 1868 features and 32 fields geometry type: POINT dimension: XY bbox: xmin: 975537.1 ymin: 148947.1 xmax: 1052509 ymax: 268518.2 CRS: 2263

```
Manhattan_tracts_shapefile <- st_read("Manhattan_tracts.shp")
```

Reading layer `Manhattan_tracts' from data source`/Volumes/GoogleDrive/My Drive/Manhattan_tracts.shp' using driver 'ESRI Shapefile' Simple feature collection with 288 features and 13 fields geometry type: MULTIPOLYGON dimension: XY bbox: xmin: -74.04728 ymin: 40.67955 xmax: -73.907 ymax: 40.88221 CRS: 4269

```
NYC_subway_amp <- tm_shape(NYC_subway_shapefile) + tm_dots()
Manhattan_tracts_map <- tm_shape(Manhattan_tracts_shapefile) + tm_polygons()

tmap_arrange(NYC_subway_amp, Manhattan_tracts_map)
```



4. Reproject your spatial objects into a projection appropriate for the geographical area of your data. This is because the sf package can only carry out a clipping using layers that are in the same projected coordinate. You can change the coordinate reference systems of spatial objects from one coordinate reference system to another by using the `st_transform()` function. For this example, a bit of research suggests that EPSG:2263 would be

an appropriate projection for New York city (https://epsg.io/2263). In the code, "NYC_subway_shapefile" and "Manhattan_tracts_shapefile" are the names of the objects we want to convert into another coordinate system, and 2263 is the EPSG code of the coordinate system to which we want to convert "NYC_subway_shapefile" and "Manhattan_tracts_shapefile". For more information and resources on coordinate systems and map projections, please see Appendix 1 in NYU Data Services' QGIS tutorial, which is available here. A list of commonly used EPSG codes can be found here.

```
NYC_subway_shapefile_projected <- st_transform(NYC_subway_shapefile, crs = 2263)
Manhattan_tracts_projected <- st_transform(Manhattan_tracts_shapefile, crs = 2263)
```

5. Use `st_intersects()` to select the intersection between the spatial objects you wish to clip and you use to restrict or bound the original object. In this example, "NYC_subway_shapefile_projected" is the spatial object we wish to clip and "Manhattan_tracts_projected" is the object we use to clip the "NYC_subway_shapefile_projected". The returned result is all subway entrance points that intersect with the Manhattan census tracts polygons, saved in a new object "subway_entrance_clipped".

```
subway_entrance_clipped <- st_intersection(NYC_subway_shapefile_projected, Manhattan_tracts_projected)
```

You can now use tmap to generate a map with the clipped spatial object overlaying the object you use for clipping. To overlay the two objects, we first plot the Manhattan census tract polygon layer and then append the code to plot the clipped subway entrance point layer afterwards. In this way we add the point layer on top of the polygon layer. You will see only clipped features remain and you can now only study the area of your interest.

```
Manhattan_subway_map <-
  tm_shape(Manhattan_tracts_projected)+
    tm_polygons()+
  tm_shape(subway_entrance_clipped)+
    tm_dots()
Manhattan_subway_map
```