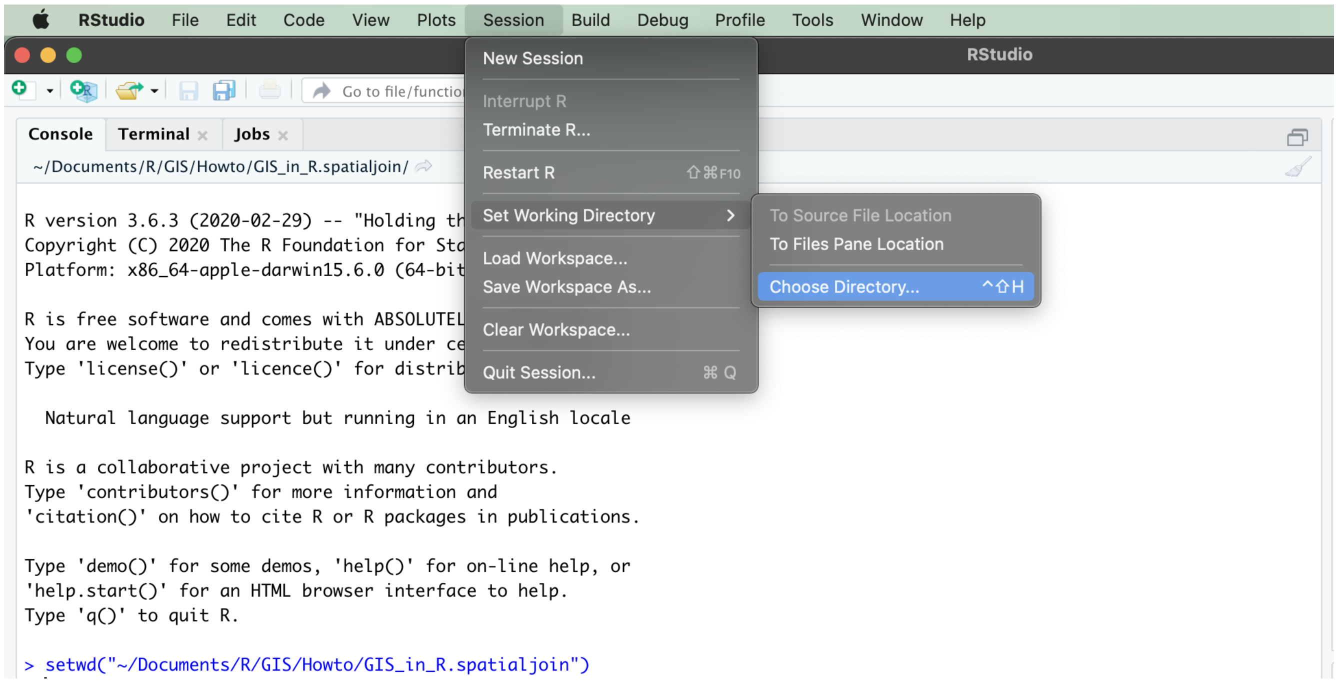
How to do spatial join in R

GIS in R: Spatial Join

Spatial join allows you to combine information from different shapefiles by using spatial relationships as the join key. You can use the the sf package's `st_read()` function to read shapefiles into R, and then convert the coordinate reference systems of the shapefiles into the same one that is appropriate for the geography of your data by using the `st_transform()` function, and then carry out a spatial join using the `st_join()` function. For example, you would like to study how many counties in each state in which a large proportion of Black or African American live in poverty. You now have two shapefiles at hand, one is a U.S. states shapefile without the attribute of poverty and the other is a shapefile of poverty rate by race at county-level. Also, you would like to conduct the analysis in R.

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.



```
R version 3.6.3 (2020-02-29) -- "Holding th
Copyright (C) 2020 The R Foundation for Sta
Platform: x86_64-apple-darwin15.6.0 (64-bit

R is free software and comes with ABSOLUTEL
You are welcome to redistribute it under ce
Type 'license()' or 'licence()' for distrib

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> setwd("~/Documents/R/GIS/Howto/GIS_in_R.spatialjoin")
```

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(dplyr)
library(sf)
```

3. Use the sf package's `st_read()` function to read your shapefiles into R.

```r
US_shapefile <- st_read("tl_2017_us_state.shp", stringsAsFactors = FALSE)
```

```
## Reading layer `tl_2017_us_state' from data source `/Users/junhui/Documents/R/GIS/Howto/GIS_in_R.spatial
## Simple feature collection with 56 features and 14 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: -179.2311 ymin: -14.60181 xmax: 179.8597 ymax: 71.43979
## CRS:            4269
```

```r
US_poverty_shapefile <- st_read("US_county_poverty.shp", stringsAsFactors = FALSE)
```

```
## Reading layer `US_county_poverty' from data source `/Users/junhui/Documents/R/GIS/Howto/GIS_in_R.spatia
## Simple feature collection with 840 features and 8 fields
## geometry type:  POINT
## dimension:      XY
## bbox:           xmin: -5903301 ymin: -2335390 xmax: 3587055 ymax: 2870408
## CRS:            2163
```

4. Reproject the point layer and polygon layer into a projection appropriate for the geographical area of your data. This is becasue the sf package can only carry out a spatial join between vector 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:2163 would be an appropriate projection for USA (http://epsg.io/2163). In the code, "US_shapefile" and "US_poverty_shapefile" are the names of the objects we want to convert into another coordinate system, and 2163 is the EPSG code of the coordinate system to which we want to convert "US_shapefile" and "US_poverty_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: https://docs.google.com/document/d/15kOALmDWGI00Hsu-gDthW_b2sm2Auv-MI9qmrjk4h-4/edit#

```r
US_shapefile_projected <- st_transform(US_shapefile, 2163)
US_poverty_shapefile_projected <- st_transform(US_poverty_shapefile, 2163)
```

5. Once the two spatial objects have been converted into the same coordinate reference system, you can use the `st_join()` function to carry out a spatial join between the two layers. The two arguments in the function are the two spatial objects you want to join.

```r
US_poverty_spatialjoin <- st_join(US_shapefile_projected, US_poverty_shapefile_projected)
```

6. Now that the attribute tables of the two spatial objects have been joined together, we would like to determine the number of counties that more than 20 percent of Black or African American were below poverty level in 2019 within each state. We can do this easily through some basic data manipulation using the dplyr package. In the code below, we first convert the data type of the field we want to calculate (Blk.AfA) to numeric, and then filter the outcome of the spatial join from the last step ("US_poverty_spatialjoin") to only include rows with the percentage of Black or African American below poverty level greater than 20 (Blk.AfA > 20), and then group this data by state name (NAME), and then generate a summary table with a field that indicates the count number of distinct county IDs (ID) associated with each state name (i.e. the grouping variable), and finally sort the result to be descending based on the count number (number_of_counties). By using `head()` function, the first few rows can be printed and you may see that the state having the most counties with more than 20 percent of Black and African American living below the poverty line in 2019 is North Carolina. For more information on working with dplyr, please see the documentation for NYU Data Services' "Data Wrangling in R" tutorial: https://guides.nyu.edu/c.php?g=851842&p=6096248

```
US_poverty_spatialjoin$Blk.AfA <- as.numeric(US_poverty_spatialjoin$Blk.AfA)

poverty_county_per_state <- US_poverty_spatialjoin %>% filter(Blk.AfA > 20) %>%
  group_by(NAME) %>% summarize("number_of_counties" = n_distinct(ID)) %>%
  arrange(desc(number_of_counties))

head(poverty_county_per_state)
```

```
## Simple feature collection with 6 features and 2 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: 1184265 ymin: -2090979 xmax: 2309786 ymax: 344335
## CRS:            EPSG:2163
## # A tibble: 6 x 3
##   NAME       number_of_counti~                                        geometry
##   <chr>                  <int>                              <MULTIPOLYGON [m]>
## 1 North Car~                22 (((1570933 -826582.7, 1570941 -826558.1, 1570944~
## 2 Ohio                      16 (((1267318 -328422.2, 1267286 -328239.2, 1267278~
## 3 Pennsylva~                16 (((1595332 -159299.3, 1595312 -159221.4, 1595284~
## 4 Georgia                   15 (((1346866 -1115974, 1346864 -1115966, 1346626 --~
## 5 New York                  15 (((2144820 -136214.7, 2144822 -136211.6, 2144818~
## 6 Florida                   14 (((1734278 -2086763, 1734277 -2086758, 1734220 --~
```