Spatial Data Science with R: Plotting a Shapefile with sf

R 4.2.2

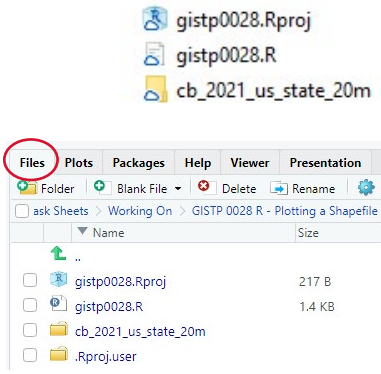
Welcome to the Data Science Task Sheet Series. This series supplements the Iowa State University Extension and Outreach Geospatial Technology Training Program's workshops and short courses by providing quick and easy instructions for performing a variety of mapping, data science, analysis, and visualization tasks.

In this task sheet you will install and use the sf package in R to work with spatial data. The sf package implements the Simple Features standard – an open-source model for storing and accessing vector data (point, line, and polygon-based geometry) commonly used by geographic information systems. You will install the sf package and visualize spatial data provided by the United States Census Bureau in shapefile format, a file type frequently used in spatial analysis and map-making. This task sheet also uses concepts covered in GISTP 0025 - Filtering and Selecting Data with dplyr and GISTP 0026 - Mutating and Piping Data with dplyr.

1. Getting Started & Downloading the Data

a. Download the data used in this task sheet from: https://isueogtp.github.io/GISTaskSheets/SpatialDataScience_r/gistp0028.zip.

b. When the download is complete, you will need to unzip the **gistp0028** folder in order to access the files in RStudio. The folder contains **gistp0028.Rproj**, a project file for RStudio; **gistp0028.R**, a completed R script; and **cb_2021_us_state_20m**, a folder containing shapefile data of state boundaries provided by the U.S. Census Bureau.



c. Open **RStudio** by double-clicking on the **gistp0028.Rproj** file. Next, create a new R script and install **sf** by running the command `install.packages("sf")`. If necessary, install the Tidyverse packages by running `install.packages("tidyverse")`. Optionally, you may open and run the completed R script from the Files tab.

d. Type and run `library(sf)` in the script to load the **sf** package into your R session; next, add `library(dplyr)` to load **dplyr**.

2. Using the Simple Features Package

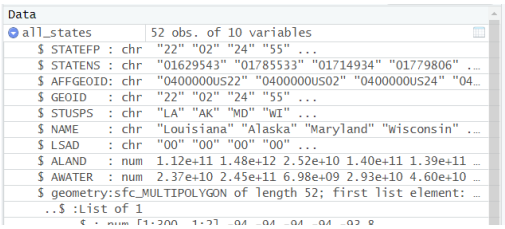
a. Next, type `all_states <- read_sf("cb_2021_us_state_20m/cb_2021_us_state_20m.shp")` to load the state boundaries shapefile. When the data is loaded, a new entry appears in the **Environment** pane. You will see there are 52 observations (sometimes referred to as features) and 10 variables.

```
> all_states <- st_read("cb_2021_us_state_20m/cb_2021_us_state_20m.shp")
Reading layer `cb_2021_us_state_20m' from data source
  `C:\Users\jaym\Box\GIS Task Sheets and Training Docs\Task Sheets\Working On\GISTP 0
OXX R - Basic Mapping\r_mapping_sandbox\cb_2021_us_state_20m\cb_2021_us_state_20m.sh
p'
  using driver `ESRI Shapefile'
Simple feature collection with 52 features and 9 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: -179.1743 ymin: 17.91377 xmax: 179.7739 ymax: 71.35256
Geodetic CRS:  NAD83
```

b. There are 50 states in the United States, but this dataset has 52 features. Type `all_states$NAME` in the script to see a list of features present in the datset. Although they are not states, District of Columbia and Puerto Rico are included in this shapefile.

```
> all_states$NAME
 [1] "Louisiana"            "Alaska"               "Maryland"
 [4] "Wisconsin"            "Florida"              "Georgia"
 [7] "Tennessee"            "Minnesota"            "Iowa"
[10] "Missouri"             "Michigan"             "Idaho"
[13] "California"           "Connecticut"          "Texas"
[16] "Virginia"             "New York"             "Illinois"
[19] "Montana"              "Kentucky"             "Oregon"
[22] "District of Columbia" "Ohio"                 "Arkansas"
[25] "Washington"           "Puerto Rico"          "Wyoming"
[28] "Maine"                "New Mexico"           "Utah"
[31] "West Virginia"        "Kansas"               "Nevada"
[34] "Mississippi"          "New Hampshire"        "Alabama"
[37] "South Dakota"         "Pennsylvania"         "Oklahoma"
[40] "North Carolina"       "Massachusetts"        "North Dakota"
[43] "Delaware"             "Colorado"             "South Carolina"
[46] "Hawaii"               "Vermont"              "Rhode Island"
[49] "Indiana"              "New Jersey"           "Arizona"
[52] "Nebraska"
```

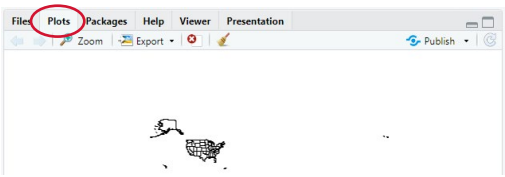
c. Click on the **blue circle** in the **Environment** pane to expand the information about **all_states**. The first nine variables are attribute data (**GEOID**, **NAME**, **ALAND**, etc) and contain data about each feature. The last variable, **geometry**, contains the spatial coordinates describing each feature's physical location on the earth.



d. Type **`st _ geometry(all _ states)`** to access the spatial data in **all_states**. Although **st_geometry()** runs on the entire data set, only a portion of the first five entries will be output to the console. The **st_geometry()** function is commonly used with other **st_*** functions of the **sf** package to work with *spatial type* data.

```
> st_geometry(all_states)
Geometry set for 52 features
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: -179.1743 ymin: 17.91377 xmax: 179.7739 ymax: 71.35256
Geodetic CRS:  NAD83
First 5 geometries:
MULTIPOLYGON (((-94.04305 32.69303, -94.04303 3...
MULTIPOLYGON (((179.4813 51.9753, 179.5829 52.0...
MULTIPOLYGON (((-76.04621 38.02553, -76.00734 3...
MULTIPOLYGON (((-86.93428 45.42115, -86.83575 4...
MULTIPOLYGON (((-81.81169 24.56874, -81.75127 2...
```

e. We can visualize the spatial data by using the **plot()** function. Type **`all _ states %>% st _ geometry() %>% plot()`** and a map of the United States will appear in the **Plots** tab.



3. Working With Spatial Data Frames

a. Type **`class(all _ states)`** in your script the output includes "sf", "tbl", "tbl_df", and "data.frame." **all_states** has inherited all the properties and beviours of those class types, and any functions that work with **sf** objects, **tibbles** (data structures used in Tidyverse packages), or **data frames** can use **all_states** as input.

b. The **head()** function returns the first six entries in a data frame and the **ALAND** attribute gives the land area in square meters for each feature. Type **`all _ states %>% arrange(desc(ALAND)) %>% head() %>% st _ geometry() %>% plot()`** to create a plot of the six largest states in the **all_states** data frame. *Note: You can specify an output of* **`n`** *objects by using* **`head(n)`**.

c. Make a subset of Iowa and the neighboring states by typing **`ia _ surrounding <- all _ states %>% filter(NAME %in% c("Iowa", "Nebraska", "Minnesota", "South Dakota", "Illinois", "Missouri", "Wisconsin")) %>% select("GEOID", "NAME", "geometry")`**. You can plot this new variable by typing **`ia _ surrounding %>% st _ geometry() %>% plot()`**. A new image will appear in the **Plots** tab.

d. The **plot()** function is good for quickly inspecting your spatial data, but other packages are better suited to producing interactive and visually pleasing maps. The packages **ggplot**, **tmap** and **leaflet** provide more options for advanced cartography and will be covered in future Spatial Data Science with R task sheets.

Contact: Jay Maxwell, Data Analyst, and Professor Christopher J. Seeger, PLA, GISP can be reached at geospatial@iastate.edu. Additional task sheets and information about the Geospatial Technology and Spatial Data Science Programs are available at www.extension.iastate.edu/communities/gis.