32.23123° N / -110.94909°W

UA Libraries Data Cooperative Unit's

GIS TUTORIALS

STARTING A PROJECT

R

SOFTWARE USED

1

TUTORIAL NUMBER

★★☆☆

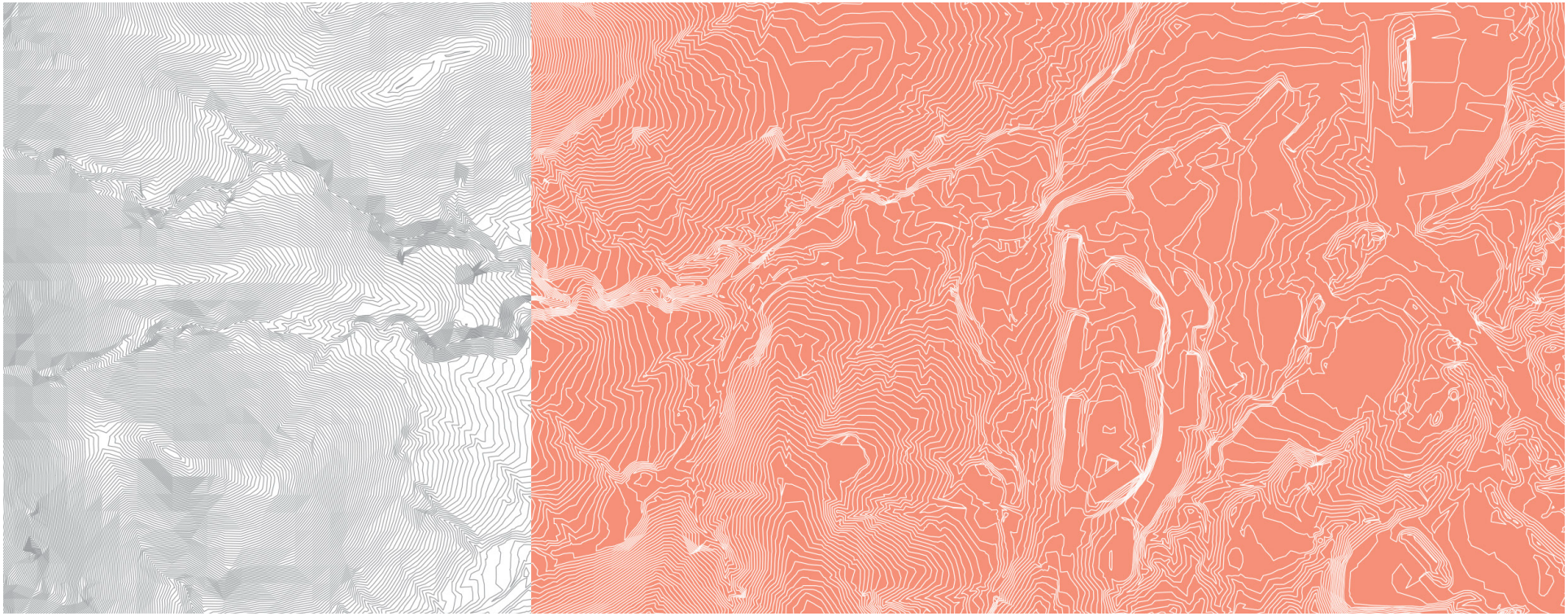
DIFFICULTY LEVEL

★★★★

LEVEL OF STOKE

HARDWARE NEEDED:	SOFTWARE NEEDED:
desktop or laptop computer	R
internet connection	RStudio

UNIVERSITY LIBRARIES
Data Cooperative



INTRODUCTION

1

The purpose of this tutorial is to teach you how to start a geospatial project in RStudio.

Upon completion of this tutorial, you should be comfortable:

1. Starting an RStudio project.
2. Installing and loading package in R.
3. Importing shapefiles into an RStudio project.
4. Exploring spatial data in R.

CREATING A NEW PROJECT

1. Open RStudio by clicking on the RStudio icon in your applications folder.

2. In the Main Menu click on File drop down menu and choose New Project...

RSTUDIO INTERFACE:

The RStudio IDE consists of a number of windows, with the default view consisting of the Console, Environments, and Files panes.

Additional tabs on panels can be added as needed depending on the R package and/or user preference.

Console

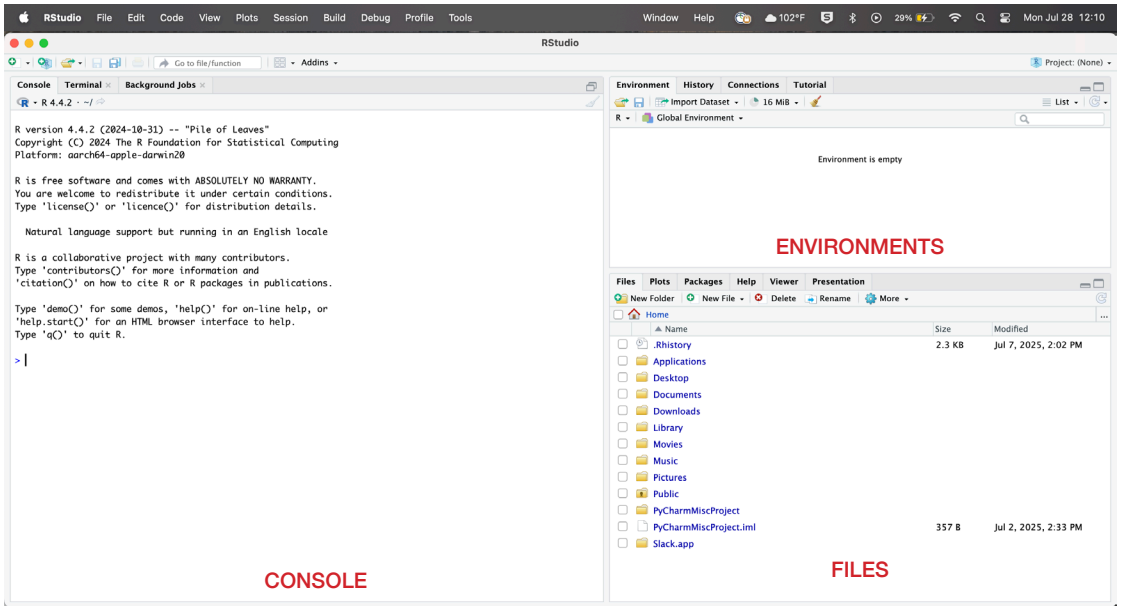
Displays the output of executed R code and allows for interactive execution of commands (R code).

Environments

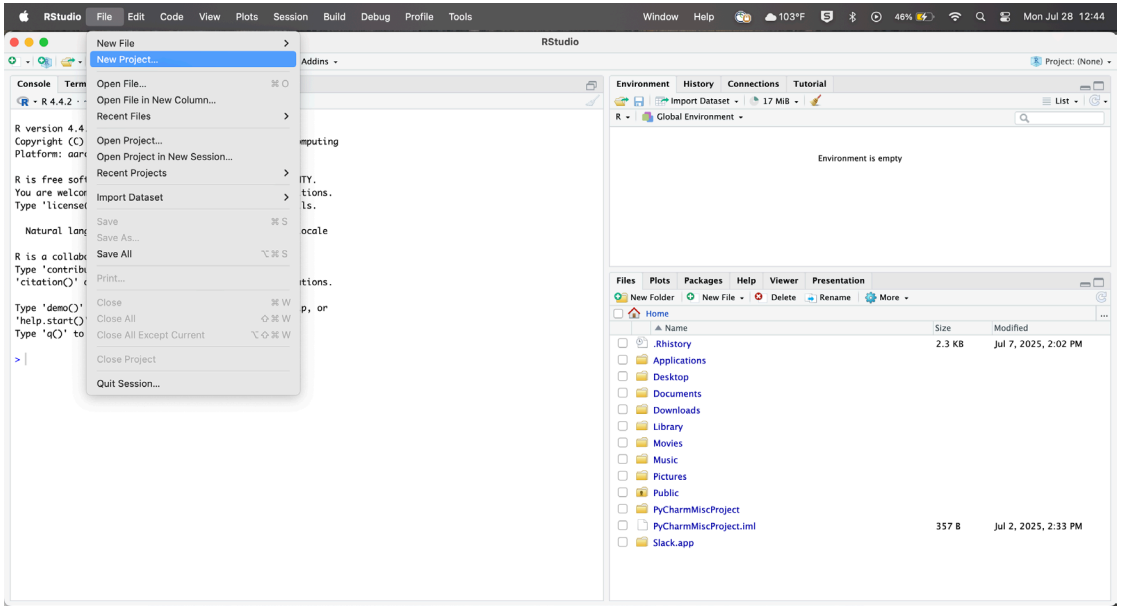
The “Environment” tab lists all active R objects, variables, and functions in the current session, while the “History” tab tracks previously executed commands.

Files

Offers tabs for navigating files, viewing plots, managing installed R packages, accessing R documentation and help files, and displaying dynamic plots or web content



1



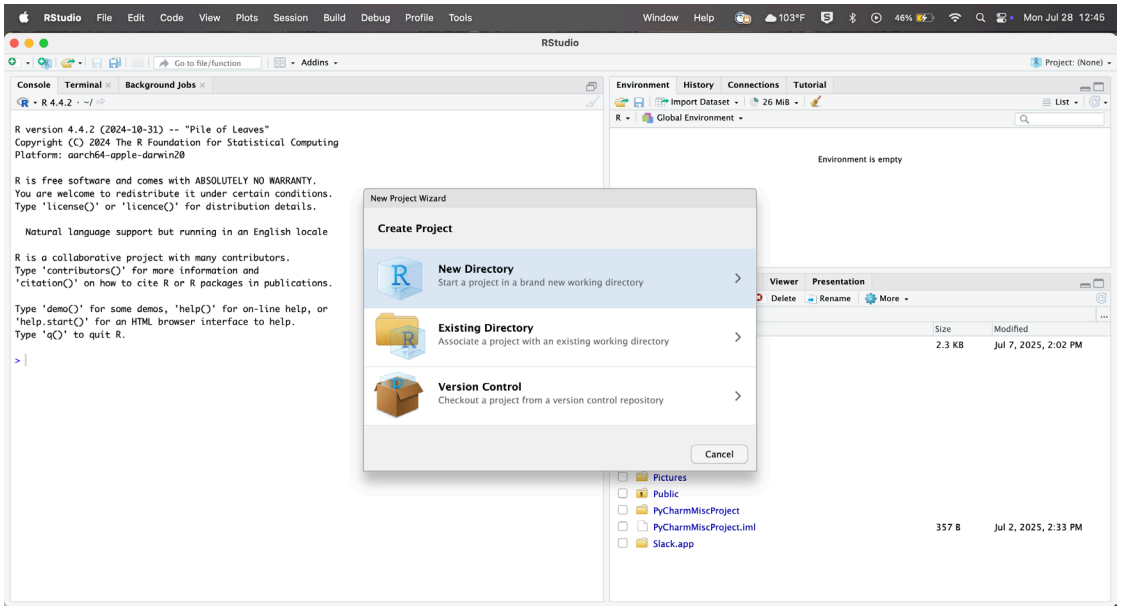
2

3. In the New Project Wizard window click on New Directory.

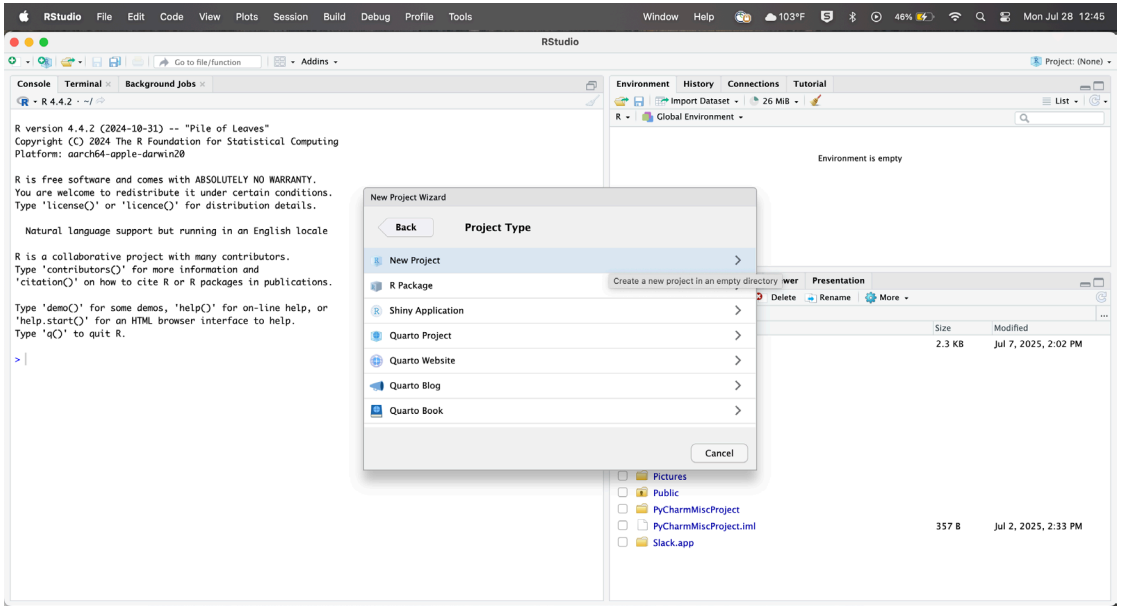
4. In the next New Project Wizard window click on New Project.

RSTUDIO PROJECT:

When working within RStudio it is a good practice to keep all of your files contained within a single project folder so that they are easier to access. When a project is created in RStudio it creates a working directory where you choose to store the project on your computer allowing for more efficient navigation to files that are placed within this folder. This also makes it simpler to resume work after closing out RStudio.



3



4

5. In the final New Project Wizard under Create New Project give your Directory name an appropriate name.

Use the Browse button to navigate to a location on your computer where you would like to store your project.

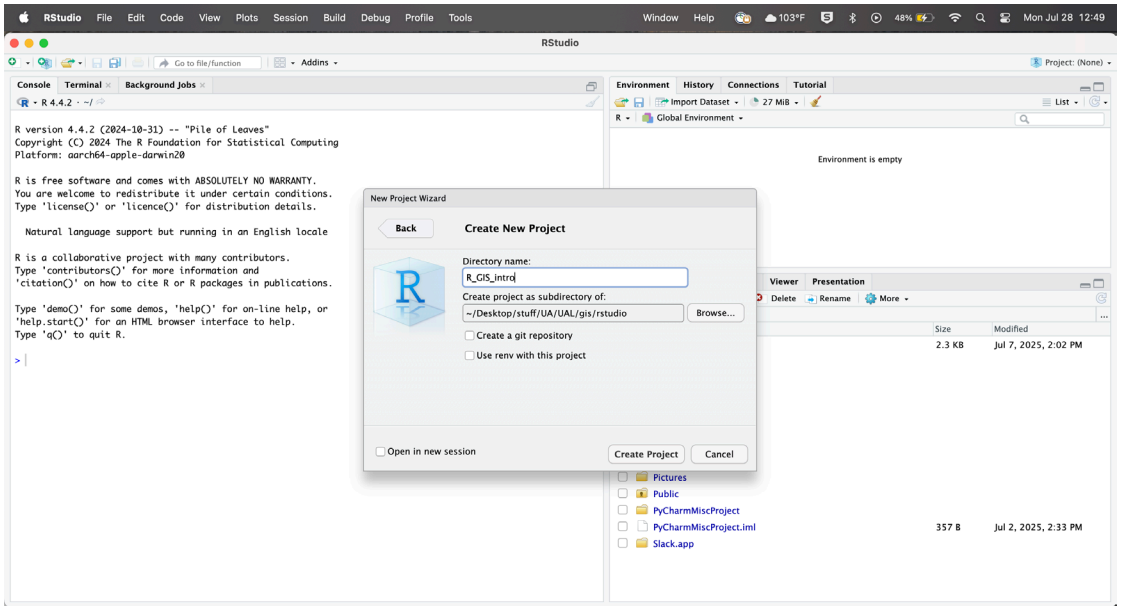
Click Create Project.

6. In the Files pane click on New Folder.

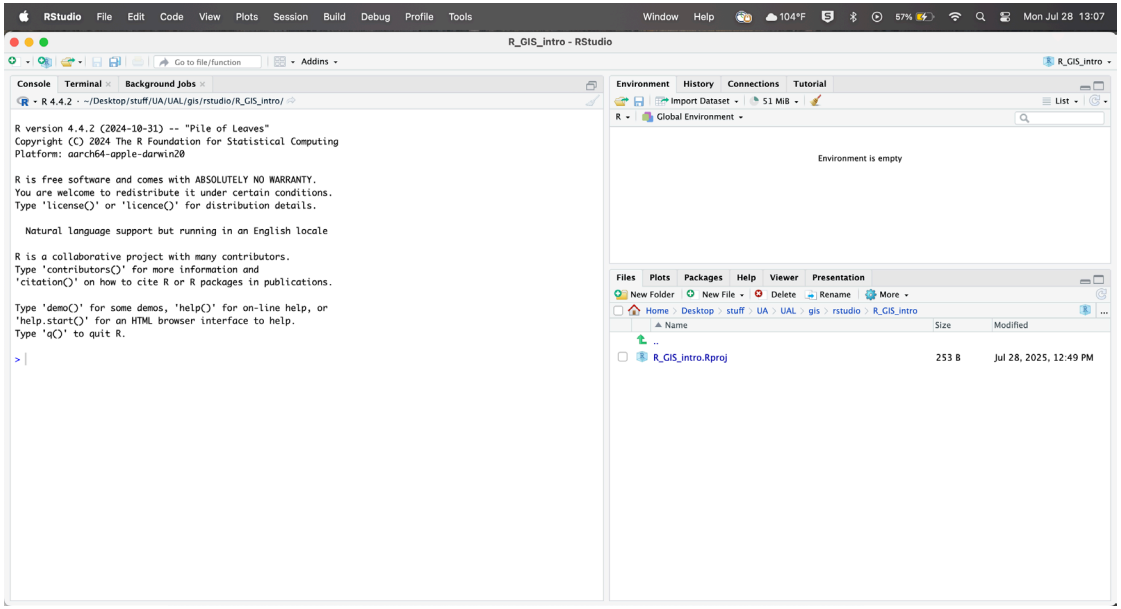
Also make note of the file path that is displayed under the New Folder button, this is the project's working directory.

WORKING DIRECTORY:

The working directory is used by R and RStudio to locate and save files that you save within your project. If you are using an RStudio Project it will automatically set the working directory for you when you open the project. The complete file path is located on the bar at the top of the Console pane.



5



6

7. In the final New Project Wizard under Create New Project give your Directory name an appropriate name.

Use the Browse button to navigate to a location on your computer where you would like to store your project.

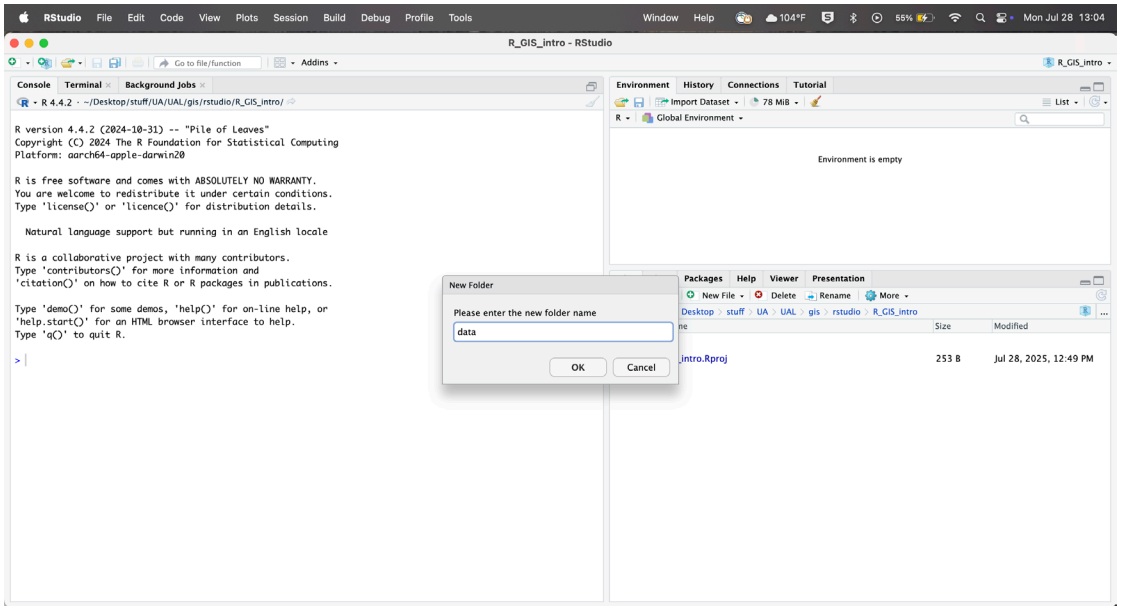
Click Create Project.

8. In the Files pane click on New Folder.

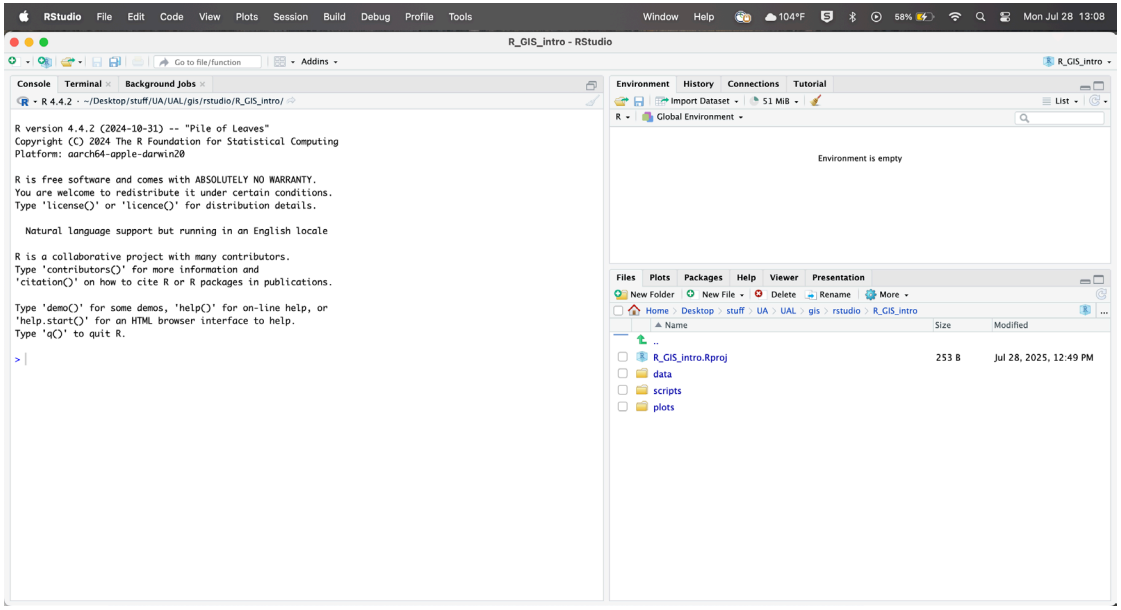
FOLDER MANAGEMENT:

When working with an RStudio Project you automatically have a root folder (the project itself). It is a good idea to organize files within the root folder into sub folders so that your root folder does not overly crowded. When you access files in these subfolders you can use the relative path to point to the sub folder and the file within instead of having to use absolute paths that require the entire file path of where the sub folder and file you are hoping to use exists on your computer.

The sub folder names are simply a recommendation and your workflow and/or work process may require different sub folder names or different sub folder types.



7



8

DOWNLOADING DATA

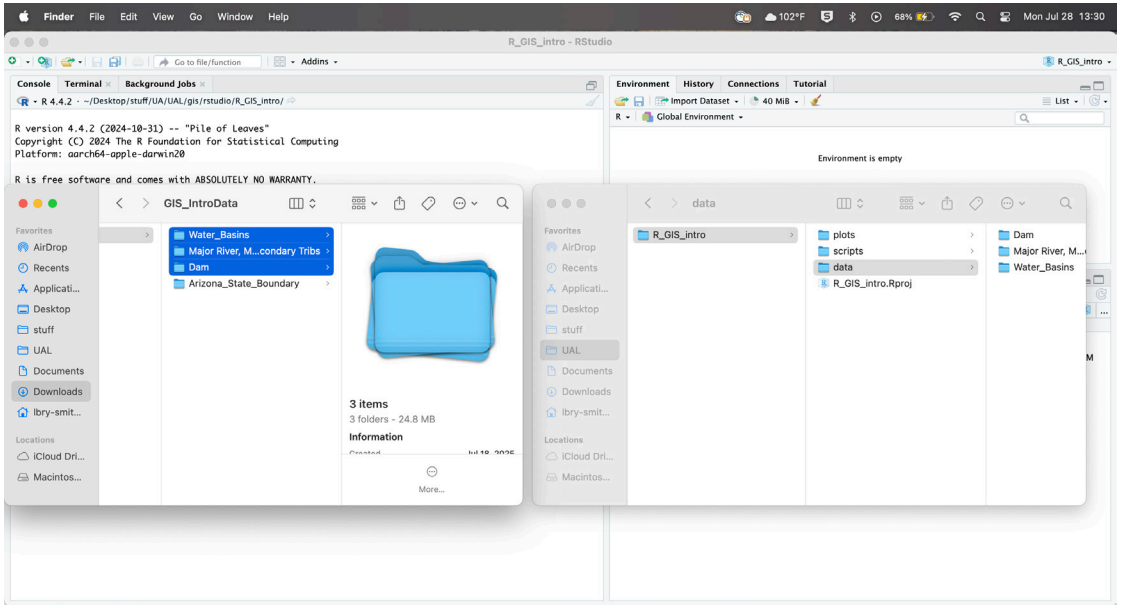
Download the data that will be used in this tutorial from the following link:

GIS IntroData.zip

Please note: All of the data used in this tutorial is available on the UA Library's GeoBlacklight Data Portal: https://geodata.lib.arizona.edu/

1. Unzip the GIS_IntroData folder and place the following folders in the data folder located within your project's folder:

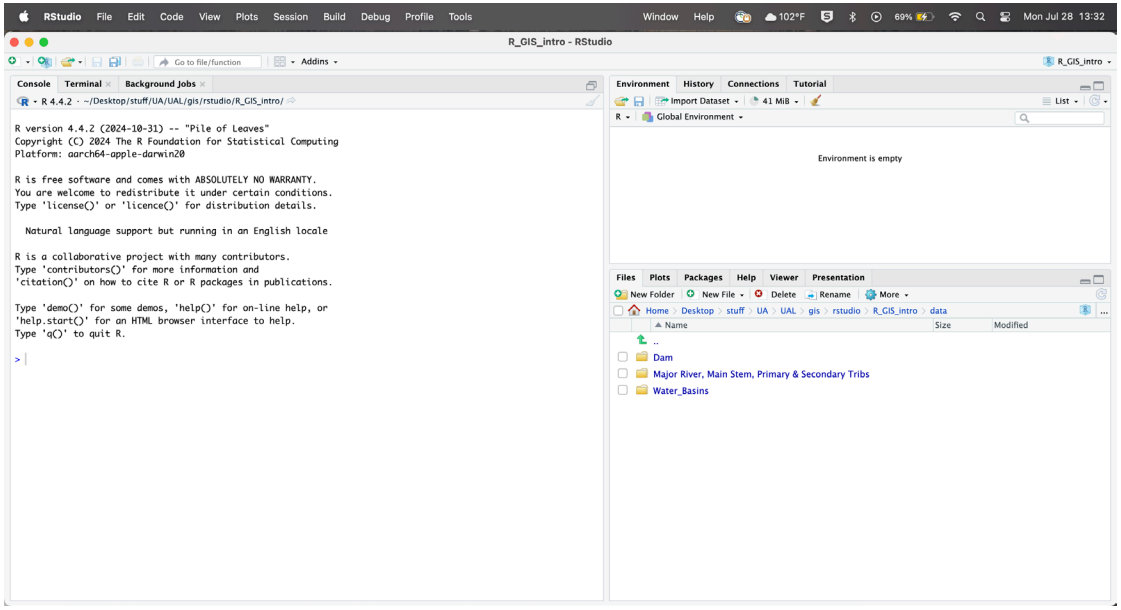
   Water_Basins
   Major River, M...econdary Tribs
   Dam



2. Return to RStudio and click on your data sub folder.

   Notice that the folders that you moved into the data folder are visible within RStudio.

   You can return to your project's root folder by clicking on the green up arrow above the sub folders in the Files pane.



CREATING AND SAVING A SCRIPT

1. In the Main Menu click on File and then New File.

In the New File pull out menu click on R Script.

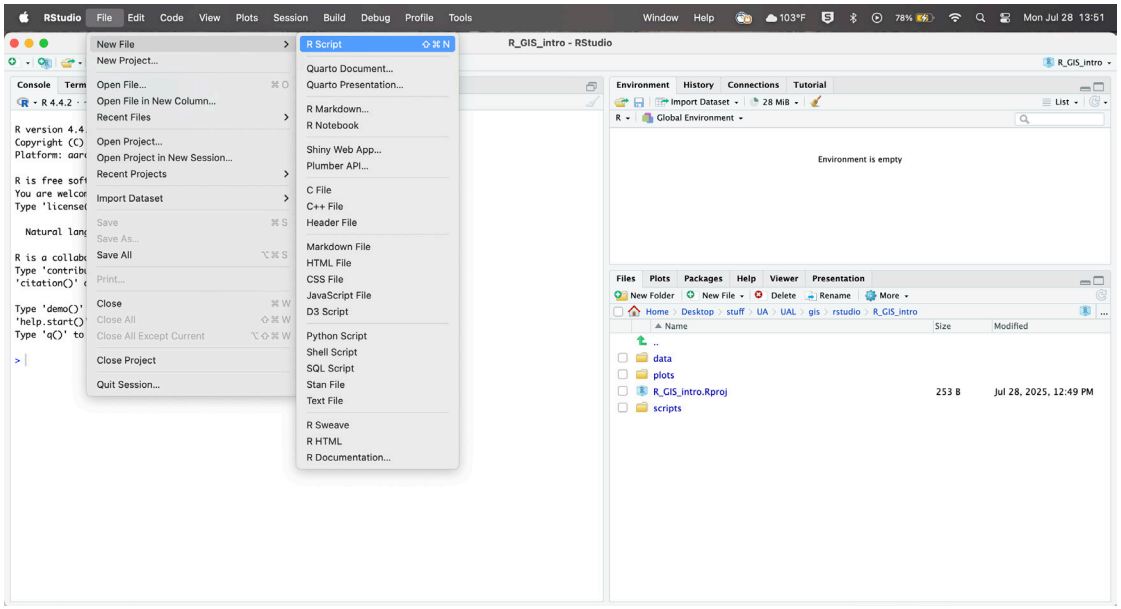
2. When the script pane opens in the Main Menu click on File and then Save.

SCRIPT EDITOR:

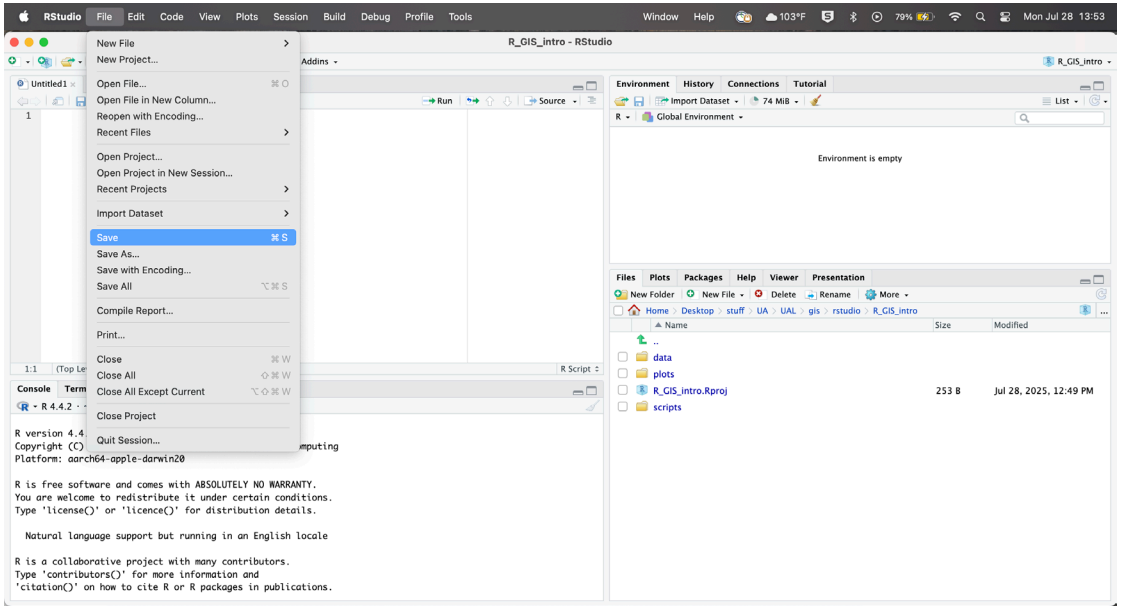
The Script Editor pane allows you to write and save multiple lines of code. Unlike the console, where you can only enter single code blocks and run them one at a time, the Script Editor allows you to enter multiple lines of code. You can also pick and choose what lines of code you would like to run. Additionally, it is much easier to edit code within the Script Editor since you do not have to retrieve it from the console. You interact with the code directly.

SCRIPTS:

RStudio scripts are plain text files that contain code written in the R programming language. An RStudio script allows you to save, organize, edit, and execute your code. You can also add notes or comments within your code by using the # or ## commands that lets RStudio know not to run those lines of code on execution.



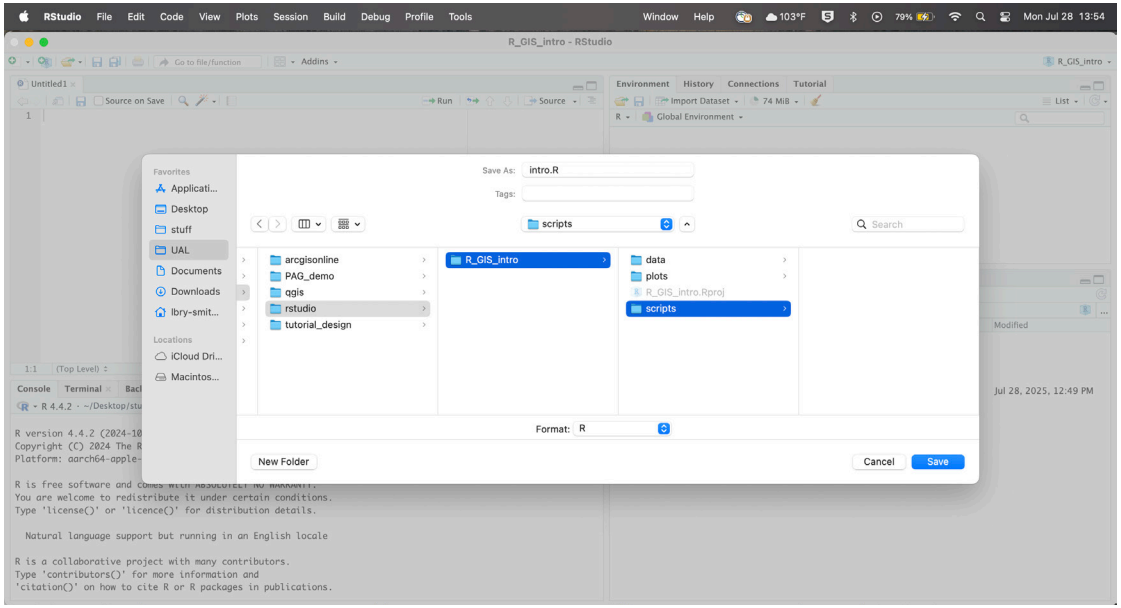
1



2

3. In the Save window give your .R file an appropriate name.

Notice that RStudio automatically directed you to your RStudio Project folder that you created in previous steps. Place your .R script file in your scripts sub folder.



3

CODING IN RSTUDIO

1. For the remainder of the RStudio tutorials you will be working in the RStudio script editor. As such, the tutorial will now include panels that are dedicated to the R code you should be entering into the Script Editor (SCRIPT EDITOR). If there is a plot (map) that you will be creating in a certain step it will appear in the PLOT panel.

These numbered instructional lines will also contain some explanation about the code you are writing. As with previous pages of this tutorial the sidebar will contain further explanations and helpful hints related to the tutorial.

SCRIPT EDITOR

```
## get the project's working directory

getwd()
```

PLOT

```
none
```

CODING IN SCRIPT EDITOR:
When you enter R code in the Script Editor pane you are creating a reproducible record of your code. Writing code in the Script Editor pane is similar to working in most text or word processing programs in that you are able to save your progress, make edits, and close the file and reopen it at the point where you last left off.

HELPFUL HINTS:
As you get more familiar with writing code you will start to develop your own coding style. For example, I tend to use a return between my comments and code as well as spaces between commands. You do not have to copy the code directly in the manner that it is presented in this tutorial. It is also generally better to enter your code manually so that you can better understand it and develop your own coding style.

When writing a script, it's useful to add comments to describe what you are doing by inserting a pound sign # in front of a line of text (I use double pound signs and generally place it above my code block) that provides context for what your code is doing. R sees anything with a # as text and does not try to run it.

INSTALLING AND LOADING PACKAGES

1. Install the `sf` and `ggplot2` packages.

These are the two packages that are needed to read geospatial files into the R enviroment and to plot a simple map to explore what the geospatial data looks like.

SCRIPT EDITOR

```
## install needed packages

install.packages("sf")
install.packages("ggplot2")
```

PLOT

```
none
```

2. Load the `sf` and `ggplot2` packages to your R project.

When you reopen this R script you will only need to run the `library()` function to load the packages that you need in your current session. After installing packages some people may choose to remove the `install.packages()` function(s) from their script.

SCRIPT EDITOR

```
## load needed packages

library(sf)
library(ggplot2)
```

PLOT

```
none
```

RUNNING CODE:
To run a code block you can place your cursor within the code block, at the end of the code block, or highlight all the lines of your code and choose from the following:

Click on the Run button above the editor panel

Press Ctrl+Enter in Windows, Ctrl+Return in Linux, or Cmnd+Return in Mac OS X.

INSTALLING AND LOADING PACKAGES:
Since R is an open-source programming language there a host of developers that develop packages that are free to use. R has a number of preloaded packages that are automatically loaded when you open RStudio. External packages are stored in repositories and you use the install.packages function to download and install these packages on your computer in the RStudio library. Once the package is downloaded you use the library function to find the package in your library and activate it for the current working session.

SF PACKAGE:
The sf package uses the "Simple Features" standard which specifies a storage and access model of spatial geometries such as points, lines, and polygons. A feature geometry is called simple when it meets certain criteria.

GGPLOT2 PACKAGE:
The ggplot2 package is a package that you can use to create data visualizations based on the grammar of graphics. In ggplot you provide the data and then tell the package how to map your data.

LOADING AND EXPLORING DATA

1. Import the Dam.shp file (shapefile) as a simple feature (sf) object into R.

You are creating an simple features (sf) object called `west_dams_sf`.

Notice that in your file path to the Dam.shp file you only need to supply the relative file path that starts within your project's root folder. The `/` tells R that it should look beyond the root folder into sub folders that are contained within it.

SCRIPT EDITOR

```
## import the dam.shp file

west_dams_sf <- read_sf("data/Dam/Dam.shp")
```

PLOT

```
none
```

2. Explore the west_dams simple feature object.

When you run the `west_dams_sf` object by itself you are provided with a summary of some of the object's key geographic features in the Console pane including:

```
Dimension: XY
Geometry type: POINT
Projected CRS: NAD83 / UTM zone 12N
```

You can scroll down in the Console pane to see additional characteristics of the data.

SCRIPT EDITOR

```
## explore west_dams_sf

west_dams_sf
```

PLOT

```
none
```

SHAPEFILES:
Shapefiles, also known as vector data, is a file storage format developed by Esri that stores the geographic location, shape, and attributes (table) of geographic features of the same geometry type.

Points are used to represent the location of something on the Earth's surface through the use of the location's latitude and longitude. Lines consist of nodes (points) and arcs (connect nodes to one another) and can be measured using length. Finally, polygons also consist of nodes and points but are closed and can be measured using area.

SMIPLE FEATURES:
Simple features are a standardized way for R to work with and encode spatial vector data (in this case shapefiles) using the OpenGIS Simple Features Access standard.

DIMENSION:
There are four dimensions possible dimensions: The four possible cases then are:
1. Two-dimensional points XY
2. Three-dimensional points as XYZ
3. Three-dimensional points as XYM
4. Four-dimensional points as XYZM (the third axis is Z, fourth M)

GEOMETRY TYPE:
There are multiple geometry types in the standard, with the most basic being point, line, and polygon.

PROJECTED CRS:
The Projected Coordinate Reference System (CRS) is transformational algorithm that is used to translate the geographic location of a feature on the Earth's surface to the projected location of the feature on a map's surface.

3. Plot a simple map of the `west_dams_sf` object.

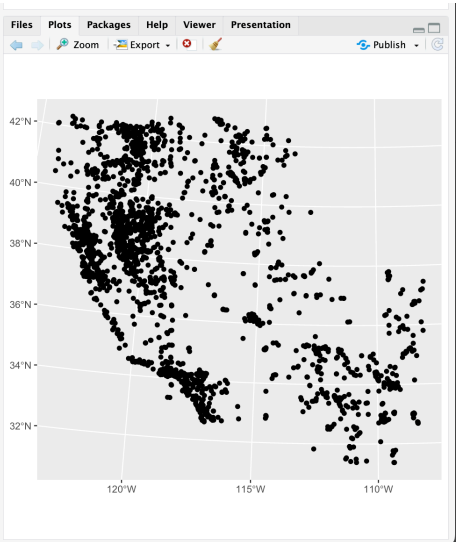
When you create a plot the Plot tab will automatically open in the Files pane displaying your plot.

The `geom_sf()` function from the `sf` package as a sf layer to your plot.

SCRIPT EDITOR

```
## create a basic map of the west dams

ggplot() + geom_sf(data = west_dams_sf)
```

PLOT



HELPFUL HINT:
One of the advantages of the writing R code in the Script Editor is that you can insert and run multiple lines of code at one time. It is more efficient to write your code blocks in chunks that you can run all at the same time.

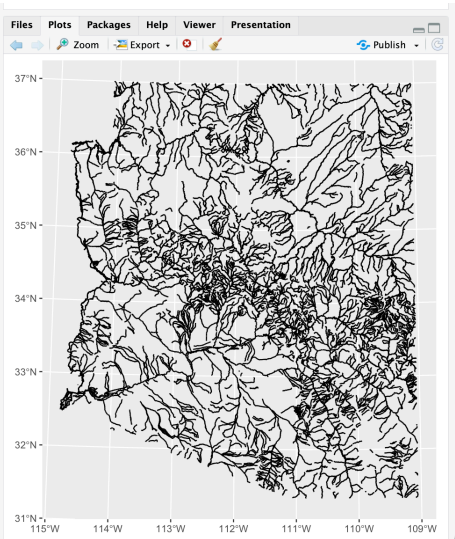
4. Complete the previous steps to create a simple feature (sf) object for the CO4.shp file.

For the Water_Basins.shp file the sub folder has a very long name. Make sure that you type the full name of the sub folder in your relevant path and do not break the line. In this case I entered a return after the `read_sf` function so that the relevant path would appear on one line.

SCRIPT EDITOR

```
## import the CO4.shp file

az_tributaries_sf <- read_sf
        ("data/Major River, Main Stem, Primary & Secondary Tribs/CO4.shp")

az_tributaries_sf

## create a basic map of az tributaries

ggplot() + geom_sf(data = az_tributaries_sf)
```

PLOT

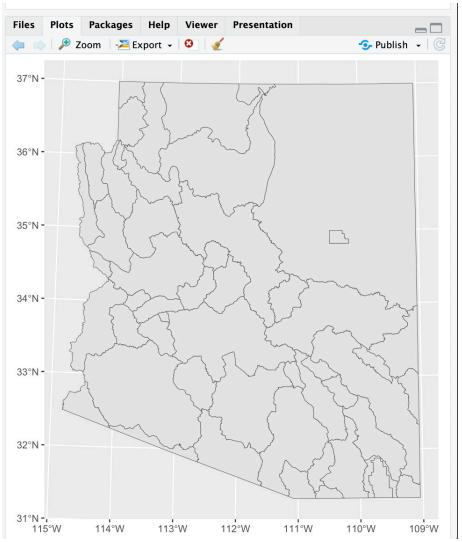


5. Complete the previous steps to create a simple feature (sf) object for the Water_Basins.shp file.

SCRIPT EDITOR

```
## import the Water_Basins.shp file

az_basins_sf <- read_sf("data/Water_Basins/Water_Basins.shp")

az_basins_sf

## create a basic map of az basins

ggplot() + geom_sf(data = az_basins_sf)
```

PLOT



HELPFUL HINT:
When you are plotting multiple sf objects with different geometry types on a map the order of how the objects should be polygons (bottom), lines (middle), and points (top). The layers will be plotted in the order that they appear in the code.

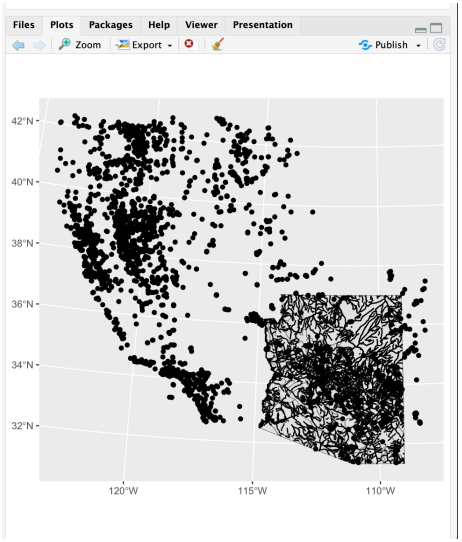
6. Plot a simple map of all the simple feature (sf) objects.

The `+` operator has a number of uses in the `ggplot2` package. In this example you are using the operator to add multiple layers (sf objects) to a plot.

SCRIPT EDITOR

```
## create a basic map with all sf objects

ggplot() +
  geom_sf(data = az_basins_sf) +
  geom_sf(data = az_tributaries_sf) +
  geom_sf(data = west_dams_sf)
```

PLOT



EXPLORING DATA IN INTERACTIVE MAPS

1. Install and load the `mapview` package.

Create a basic interactive Leaflet map of `west_dams_sf` object. Notice that the map opened in the Viewer tab instead of the plot tab.

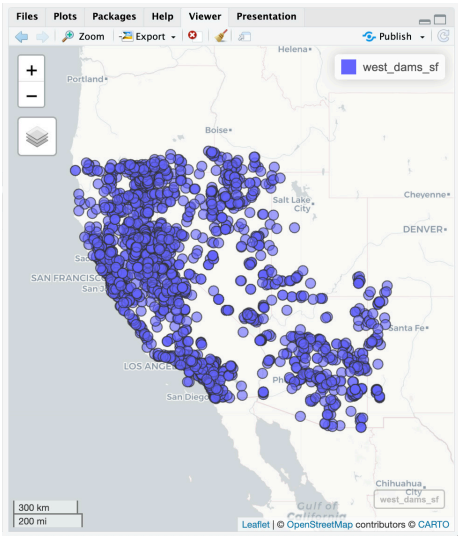
You can interact with the plot including zooming in and out, changing basemaps, and clicking on individual features to view a pop-up that is populated using data table (attribute table).

Plot your other sf objects on interactive maps.

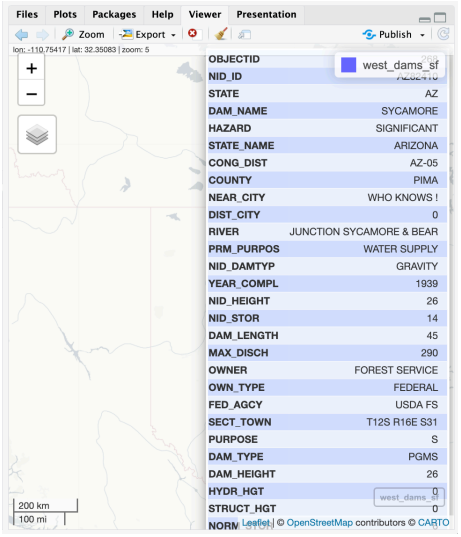
SCRIPT EDITOR

```
## install and load the mapview package

install.packages("mapview")
library(mapview)

## create a basic interactive map of west dams

mapview(west_dams_sf)
```

PLOT A



PLOT B



MAPVIEW PACKAGE:
The mapview package uses the leaflet package to quickly plot geospatial data on an interactive map that allows you to explore the data on a basemap to ensure that it is plotted in the correct location

LEAFLET MAPS:
Leaflet maps use the Leaflet Javascrpit library to create mobile- and web-friendly interactive maps.

HELPFUL HINT:
In addition to exploring the geographic features of your simple feature it is also a good idea to plot the simple features on leaflet maps so that you can determine if they are being projected in the correct location.

END