Geospatial analysis using Julia

Abstract: Geospatial technologies are related to the collection or processing of the data that is associated with the specified location. Data analysis is about inspecting and plotting the data to find out the characteristics of the area based on different parameters, with geospatial data we can also inspect the metadata which mean using the position of the data, we can learn something about the location. Using GDAL and ArchGDAL packages of Julia we are analyzing the geospatial data using the Raster data structure.

I. INTRODUCTION

Geospatial, it is the data which identifies the geographic locations and boundaries on Earth. In simple terms, Geospatial information is Geography and mapping. It is" Place Based "or" Locational Based" information. It is data tied to and portrayed on a map. Geospatial data is usually stored as coordinates and topology and accessed through Geographic Information Systems (GIS). This can also be analyzed or manipulated.

Geospatial analysis started in Canada for cataloging natural resources in the 1960s, using the first geographic information systems (GIS). [1] Geographic information systems are used to predict, manage and learn about all kinds of phenomena affecting the earth. Geospatial analyst filters out relevant from irrelevant data and apply it to conceptualize and visualize the order hidden within the apparent disorder of geographically sorted data.

Raster Data

Raster data set is made up of a matrix of pixels, which consists of the rows and columns of pixels. Each pixel represents geographical region whereas the value in the pixel represents the conditions and the characteristics of the area covered by the pixels. The value in the pixel can be continuous and categorical. The geospatial raster looks like a digital photo which is also contains the information about the location in the form of rows and columns and the locations coordinate reference system. [2] Raster data structure is advantageous because it can represent the continuous surfaces with the high level of details. cell by cell calculations are fast and gives the efficient results.

Interaction with GDAL

GDAL, known as GDAL/OGR, is a library of tools used for manipulating geospatial data.it works on both raster and vector data types, and is an incredible useful tool to be familiar with when working with geospatial data [3]. GDAL library provides two components: the GDAL component which supports the reading/writing/translation of numerous raster formats, and the OGR component which supports reading/writing/translation of numerous vector data formats.

Traditionally, the word "GDAL" was used to refer to the raster-related half of the library, while "OGR" referred to the vector part [4].

Commonly used commands

There are many commands available out of which the following four commands are frequently used.

Orginfo-Get information about vector dataset
gdalinfo-Get information about raster dataset
org2org-Convert vector data between file formats gdal_translate-Convert raster data between file formats.

Principles of ArchGDAL:

simplicity: without unnecessary additions or modifications. (I) It Stores GDAL Data Model and makes available GDAL/OGR methods without trying to mask them from the user. (II) minimal dependencies.

modernity: ArchGDAL strives to maintain the latest stable release versions of GDAL as long as systemic package breakage can be reasonably ignored. [5] It is mainly based on a rolling-release system, which allows a one-time installation with continuous upgrades.

pragmatism: It is a pragmatic distribution rather than an ideological one. The principles here are only useful guidelines. Ultimately, design decisions are made on a case-by-case basis through developer consensus. The large number of packages and build scripts in the various ArchGDAL repositories offer free and open source software for those who prefer it, as well as proprietary software packages for those who embrace functionality over ideology.

user-centrality: while other libraries are striving to be more user-friendly, ArchGDAL is user-centric. Instead of trying to appeal to as many users as possible, it is intended to fill the needs of those who contribute to it. Archers can add packages to the repositories of Arch User Repositories.

versatility: ArchGDAL is a distribution for general purposes. Only a command-line interface is presented upon installation instead of pulling out unnecessary and unwanted packages, the client is given the ability to build a custom program by selecting from thousands of high-quality packages for the x86-64 architecture provided in the official repositories. ArchGDAL, a plain, fast and lightweight package. ArchGDAL will strive to remain small in its assumptions about the range of user-needs, and to make it easy for users to build their own extensions/conveniences.

II. SOFTWARE AND LIBRARIES

Operating systems

macOS Catalina version 10.15.1

system configuration

Processor 1.8 GHz Dual-Core Intel Core i5
Memory 8 GB 1600 MHz DDR3

Programming Language

Julia version 1.2.0

Software

Jupyter Notebook open source software

Julia Packages

Pkg
GR
GDAL
Gtk
ArchGDAL
ImageView
Images

ImageMagick
Colors
Plots
BenchmarkTools

III. IMPLEMENTATION

Installing Packages

All the above-mentioned Julia packages need to be installed to for the analysis of the data, Though the major package that will be required is “ArchGDAL” the remaining packages support for the output and to plot the data.

```
julia> using Pkg

julia> Pkg.installed()
Dict{String,Union{Nothing, VersionNumber}} with 23 entries:
  "ArchGDAL"             => v"0.2.2"
  "MatrixMarket"         => v"0.3.1"
  "GR"                   => v"0.42.0"
  "Cubature"             => v"1.4.1"
  "Distributions"        => v"0.21.1"
  "BenchmarkTools"       => v"0.4.3"
  "Optim"                => v"0.19.5"
  "JuMP"                 => v"0.20.1"
  "LinearAlgebra"        => nothing
  "Roots"                => v"0.8.3"
  "Ipopt"                => v"0.6.1"
  "SymPy"                => v"1.0.7"
  "Images"               => v"0.19.1"
  "IJulia"               => v"1.20.0"
  "SparseArrays"         => nothing
  "ImageMagick"          => v"0.7.5"
  "Plots"                => v"0.27.1"
  "Colors"               => v"0.9.6"
  "DifferentialEquations" => v"6.9.0"
  "Cuba"                 => v"2.0.0"
  "ImageView"            => v"0.10.0"
  "GDAL"                 => v"0.2.0"
  "Gtk"                  => v"0.18.0"
```

Reading, writing and plotting Raster data

There are few properties which we tested first and later used in writing and plotting. Raster dataset often have the several bands of data and we can check the total number of bands of the raster, and also, we can check the height and width of the raster dataset, using height and width we can project the dataset. All these properties give us the details about the raster dataset.

(a) Number of bands
Num_raster = (ArchGDAL.nraster(dataset))

(b) Width and Height width = ArchGDAL.width(dataset)
height = ArchGDAL.height(dataset)

(c) Projection of dataset
Projection_data = AG.getproj(dataset)

1.Reading a Raster data

First step we did is to read the first band of raster dataset into an array which we named as array. To do this step we wrote a function named gdal_read() as below using the ArchGDAL package. To make ArchGDAL short we made constant AG and whenever ArchGDAL is required we can simple call AG.

Reading the raster data

```
In [2]: function gdal_read()
        AG.registerdrivers() do
        AG.read("Isle_wight.tif") do dataset
        band_1 = AG.getband(dataset,1)
        array = AG.read(band_1)
        end
        end
        end

Out[2]: gdal_read (generic function with 1 method)
```

2. Writing the raster data

We can create a new raster band and assign the projection and transformation parameters to the new band as follow.

Writing the new raster data

```
In [7]: AG.registerdrivers() do
            AG.read("Isle_wight.tif") do dataset
            band_1 = ArchGDAL.getband(dataset,1)
            ref = AG.getproj(dataset)
            new1 = ArchGDAL.read(band_1)
            geotransform = AG.getgeotransform(dataset)

            raster_band = AG.unsafe_create(
                "new_band.tif",
                AG.getdriver("GTiff"),
                width = ArchGDAL.width(dataset),
                height = ArchGDAL.height(dataset),
                nbands = 1,
                dtype = UInt8
            )

            AG.setgeotransform!(raster_band, geotransform)
            AG.setproj!(raster_band, ref)


            AG.write!(
                raster_band,
                new1,
                1,
            )
            AG.destroy(raster_band)
            end
        end

Out[7]: Ptr{Nothing} @0x0000000000000000
```

Here we created a new .tif file called new_band.tif with the parameter’s width, height, band type and band number. This is assigned to a variable raster_data.

3.Testing the newly created band

After creating the new band, we checked the raster band and the result is as follow

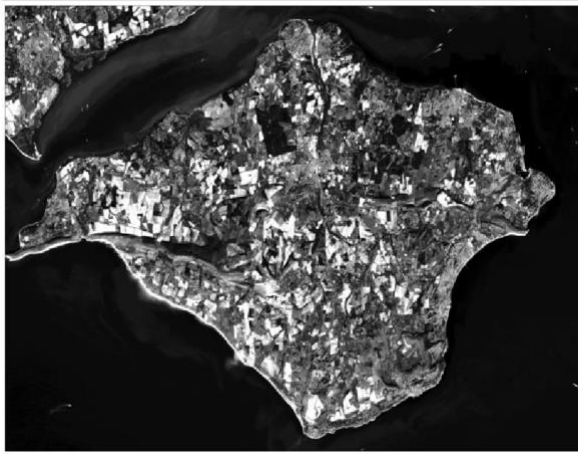
Checking the newly created raster band

```
In [8]: AG.registerdrivers() do
        AG.read("new_band.tif") do dataset
            band = AG.getband(dataset,1)
            new = AG.read(band)
            band = AG.getband(dataset,1)
            band_array = AG.read(band)
            band_array_adjust = permutedims(band_array, (2,1))
            band_array_float = band_array_adjust / 255
        image_1 = Gray.(band_array_float)

            end
        end
```

Result:

Out[8]:



4.Plotting raster data

For plotting the raster data, first we read the dataset and extract its width and height. we then used this information to build an array and use a loop to iterate over bands.Then using the colorview we plot the image of the raster data, I assigned a variable image_2

Plotting the raster data ¶

```
In [9]: ArchGDAL.registerdrivers() do
        ArchGDAL.read("isle_wight.tif") do dataset
            number_rasters = (ArchGDAL.nraster(dataset))
            width = ArchGDAL.width(dataset)
            height = ArchGDAL.height(dataset)
            m = Array{Float64}(undef, height, width, number_rasters)
            for i = 1:number_rasters
              band = ArchGDAL.getband(dataset,i)
                b = ArchGDAL.read(band)/255
                b_n = permutedims(b, (2,1))
                m[:, :, i] = b_n
            end
            typeof(m)
            image_2 = colorview(RGB, permuteddimsview(m, (3,1,2)))

            end
        end
```

IV APPLICATIONS

There are huge number of applications for the geospatial technologies in many different fields of science and engineering. Few are them are below.

1.Alcohol density and violence

By using geospatial analysis, we can checkout places where the alcohol exists and in the second step again using the geospatial analysis, we can check out the violence happening over there. Finally, by comparing both the results of alcohol existence and violence we can conclude how much alcohol existing sites contribute for the violence in those places.[6]

2.Environmental health risk analysis

The geospatial analysis can also be used to check the environmental conditions, this is done by considering a sample of location data and analyzing it using raster data structure. This helps us to study the change in environment and risks due to change in the environment.

3.Biosecurity and health informatics

A geospatial database of the biological species at a specific location would give the details about the existence and dependencies of the species on the location.

4.Urban Planning

Geospatial analysis can also be used to check if the location can be urbanized and also the proximity to which level the location can be developed

5.sociological research

By analyzing the landscapes of the location, a social research can be done on how people

live and organize their lifestyle on the location.

These are the few applications of the geospatial analysis, the analysis can be integrated with technologies like cloud, internet of things, bigdata and automation for the many more innovative applications in the field of geospatial analytics.[8]

Conclusion

GDAL library provides the tools to read write and plot the geospatial data. Geospatial data which is based on the co-ordinates of the locations gives the information about the location based on various parameters. The data being displayed in the pixel format gives the detailed information about the location, which helps the researchers for the detailed study. We considered a sample image and then tried to read write and plot the data using raster data structures. As a part of geospatial data manipulation, we made use of GDAL and ArchGDAL packages of Julia which provided the inbuilt functions for the analysis of data.

References

[1]https://whatis.techtarget.com/definition/geospatial-analysis

[2] https://datacarpentry.org/organization-geospatial/01-intro-raster-data/

[3] https://www.dataone.org/software-tools/gdal-geospatial-data-abstraction-library

[4]https://developers.planet.com/planetschool/getting-started-with-gdal/

[5]http://yeesian.com/ArchGDAL.jl/latest/

[6]https://academic.oup.com/alcalc/article/39/4/369/139368

[7]https://sci-hub.tw/10.1080/00045608.2012.674900

[8] https://www.cleantech.com/innovative-applications-in-geospatial-analytics/