

Exercise # 3

Geospatial Data Abstraction Library (GDAL) is by far the most powerful utility that you can use for spatial data processing. If you are programming, in C, C++ or Python or if you are barely using the linux command line GDAL can be your best companion with that. There are utilities packaged with GDAL which work both with raster and vector datasets. For our workshop, we will be getting some hands-on experience, with GDAL and OGR tools. Once, we are familiar with how they work we will then head on to using these for handling data to manage it in our postgres database. The four important commands that we will be using in our exercise are.

1. **ogrinfo** - Get information about a vector dataset.
2. **gdalinfo** - Get information about a raster dataset.
3. **ogr2ogr** - Convert vector data between files formats.
4. **gdal_translate** - Convert raster data between files formats.

First, we are going to start with some vector data. Here we have provided you with some real data from a flood analysis that was done for the entire Indus basin back in 2015.

First you will need to open the terminal in your virtual machine and type the following command.

```
ogrinfo --formats
```

This will list a set of formats that is supported by OGR. To check if a certain format that you are looking for is supported or not you might want to run the following command.

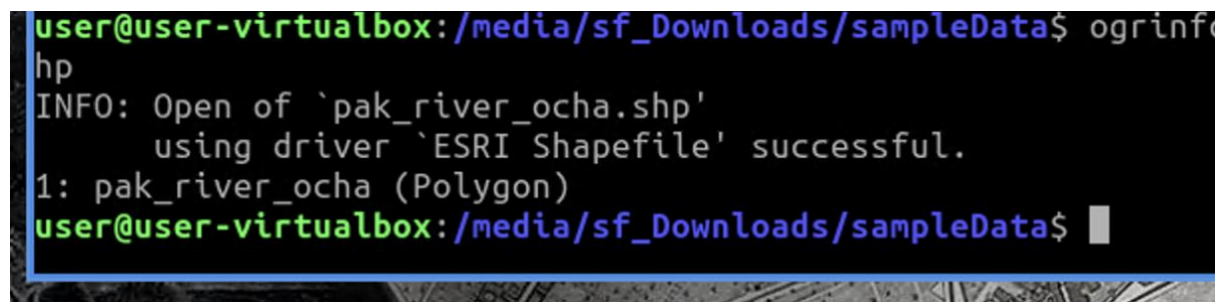
```
ogrinfo --formats | grep ESRI
```

you can try a few combinations and note the output in your daily log sheets.

You might also want to query a file provided to you, to do that, you can type the following command.

```
ogrinfo pak_river_ocha.shp
```

the following output will appear on your screen, depending on the file you have chosen.



```
user@user-virtualbox:/media/sf_Downloads/sampleData$ ogrinfo
hp
INFO: Open of `pak_river_ocha.shp'
      using driver `ESRI Shapefile' successful.
1: pak_river_ocha (Polygon)
user@user-virtualbox:/media/sf_Downloads/sampleData$
```

You might also want to get some additional information about your data, for which you can use the following command.

```
ogrinfo -al -so pak_river_ocha.shp
```

This will give you the following output.

```
user@user-virtualbox:/media/sf_Downloads/sampleData$ ogrinfo -so -al pak_river_ocha.shp
INFO: Open of `pak_river_ocha.shp'
      using driver `ESRI Shapefile' successful.

Layer name: pak_river_ocha
Metadata:
  DBF_DATE_LAST_UPDATE=2012-01-30
Geometry: Polygon
Feature Count: 1
Extent: (62.771213, 23.786926) - (78.760742, 35.645671)
Layer SRS WKT:
COMPOUNDCRS["WGS 84 + EGM96 height",
  GEOGCRS["WGS 84",
    ENSEMBLE["World Geodetic System 1984 ensemble",
      MEMBER["World Geodetic System 1984 (Transit)"],
      MEMBER["World Geodetic System 1984 (G730)"],
      MEMBER["World Geodetic System 1984 (G873)"],
      MEMBER["World Geodetic System 1984 (G1150)"],
      MEMBER["World Geodetic System 1984 (G1674)"],
      MEMBER["World Geodetic System 1984 (G1762)"],
      MEMBER["World Geodetic System 1984 (G2139)"],
      ELLIPSOID["WGS 84",6378137,298.257223563,
        LENGTHUNIT["metre",1]],
      ENSEMBLEACCURACY[2.0]],
    PRIMEM["Greenwich",0,
      ANGLEUNIT["degree",0.0174532925199433]],
    CS[ellipsoidal,2],
    AXIS["geodetic latitude (Lat)",north,
      ORDER[1],
      ANGLEUNIT["degree",0.0174532925199433]],
    AXIS["geodetic longitude (Lon)",east,
```

In case you would like to save this information somewhere, you can use the following command, and this output will be saved in a text file on your computer. These tricks are handy when you are using a large batch of data for your work and would like to get some useful information on it, together. More on that later.

```
ogrinfo -al -so pak_river_ocha.shp > pak_river_ocha.txt
```

to open the text file and check its contents, you can type the following command.

```
nano pak_river_ocha.txt
```

to exit the text editor type

```
ctrl+x
```

alternatively, you can type.

```
featherpad pak_river_ocha.txt
```

To only print the projection info, you can type the following command.

```
ogrinfo -so -al pak_river_ocha.shp | grep ELLIPSOID
```



```
user@user-virtualbox:/media/sf_Downloads/sampleData$ ogrinfo -so -al pak_river_ocha.shp | grep ELLIPSOID
ELLIPSOID[\"WGS 84\",6378137,298.257223563,\"
user@user-virtualbox:/media/sf_Downloads/sampleData$
```

Sometimes, you might need to convert the file formats. For that, the syntax is.

```
ogr2ogr -f <output format><destination filename><source filename>
```

for instance, if you would like to change the format to GeoJSON, you can type the following command in the terminal.

```
ogr2ogr -f GeoJSON pak_river_ocha.geojson pak_river_ocha.shp
```

Can you please try changing the same file to the following formats?

1. CAD -raster, vector- (rovs): AutoCAD Driver
2. FlatGeobuf -vector- (rw+v): FlatGeobuf
3. Geoconcept -vector- (rw+v): Geoconcept
4. GeoRSS -vector- (rw+v): GeoRSS
5. GPSTrackMaker -vector- (rw+v): GPSTrackMaker
6. VFK -vector- (ro): Czech Cadastral Exchange Data Format
7. PGDUMP -vector- (w+v): PostgreSQL SQL dump
8. OSM -vector- (rov): OpenStreetMap XML and PBF

To change projection of your dataset you can use the following syntax.

```
ogr2ogr <filename> -t_srs <"projection you want to select"> pak_river_ocha.shp
```

```
ogr2ogr pak_river_ocha_REPROJECTED.shp -t_srs "EPSG:4326" pak_river_ocha.shp
```

For raster data

You can use the following commands to access your raster data information.

```
gdalinfo <filename>
```

You can now switch to your Landsat9 image folder using the following command.

```
cd /media/sf_Downloads/sampleData/Landsat9Image/
```

and type

```
gdalinfo LC09_L2SP_149038_20230810_20230812_02_T1_SR_B1.TIF
```

You can much like the ogrinfo command use the gdalinfo command to list the formats that it can handle by using the following command.

```
gdalinfo --formats
```

That is quite a long list of formats, and to get an idea of the number of formats, gdal can handle you can type the following command.

```
gdalinfo -formats | wc -l
```

if you are using the latest version of osgeolive you will get 152 as the output on screen, this is number of formats that gdal can handle. You can print the same for ogrinfo and list the number of formats it gives you.

Much like ogrinfo gdal can convert between file formats. This is useful when you must save storage space, or conform to existing file formats in your inventory etc.

`gdal_translate -of <output format> <source filename> <destination filename>`

for instance, if you would like to convert your spatial data into png file format, you can use the following command.

```
gdal_translate -of png LC09_L2SP_149038_20230810_20230812_02_T1_SR_B1.TIF
LC09_L2SP_149038_20230810_20230812_02_T1_SR_B1.PNG
```

in an instance, when you have many files to convert from one forma to another you can use the following loop, this will ease your tedious jobs.

```
for i in *.TIF; do gdal_translate -of png $i `echo $i | cut -d . -f1`.png; done
```

Your screen will look like the following during the conversion.

[illegible]

Now the question remains as to why was this necessary to know about? Since most of the time, in case of spatial databases we need to port data, and the data needs to be in a good enough format for easy consumption, these commands come handy. There are also instances, when we might be required to convert the data from one projection system to another in bulk before it can be ingested into our database. Let's assume you got your data in UTM, and you already have a database that stores data in lat long projection system. For

that purpose, you will be required to convert the file projection to the desired projection system. Doing it in bulk is made easier by gdal and ogr tools. The syntax for changing between projection systems for a file is.

```
gdalwarp -t_srs EPSG:<code> <input> <output>
```

you can also use the following command in case you want to include other specifics such as projection type and resolution etc.

```
gdalwarp -tps -t_srs '+proj=longlat +datum=WGS84 input.tif output-ll.tif
```

Can you use the knowledge gained from this exercise to bulk reproject the files in the Landsat-9 folder to latitude and longitude grid?