“A User’s Guide to Importing Raster Data into PostGres
&
Tutorials on Raster Operations”

#Contributors:
Jerónimo Rodríguez
Jillian Long
Kevin Frech
Leonardo Hughes

//
RBB;

Introduction

Why Raster?

Spatial data types provide the information that a computer requires to reconstruct the spatial data in digital form. In the raster world, we have grid cells representing real world features.

Raster data is made up of pixels (also referred to as grid cells). They are usually regularly-spaced and square but they don’t have to be. Rasters often look pixelated because each pixel has its own value or class.

Advantages:

- Nature of data storage makes data analysis relatively easy and quick
- Useful for a wide range of applications (base maps, surface maps, thematic maps, etc)
- Advanced statistical use
- Can uniformly store points, lines, polygons and surfaces
- Great for continually varying information

Disadvantages:

- Cell size determines resolution, resolution can be bad, which = can’t analyze data
- Hard to represent linear features
- Most output maps from grid-cell systems do not conform to high-quality cartographic needs
- Understanding raster data requires a bit of training the eye. This can be intimidating.
- The environment can work against you (ex: clouds in an image). Removing these while maintaining data integrity can be very frustrating (J. Rodriguez, 2018).

raster2pgsql

The raster2pgsql is a raster loading package which loads GDAL supported raster formats into sql suitable for loading into a PostGIS raster table.

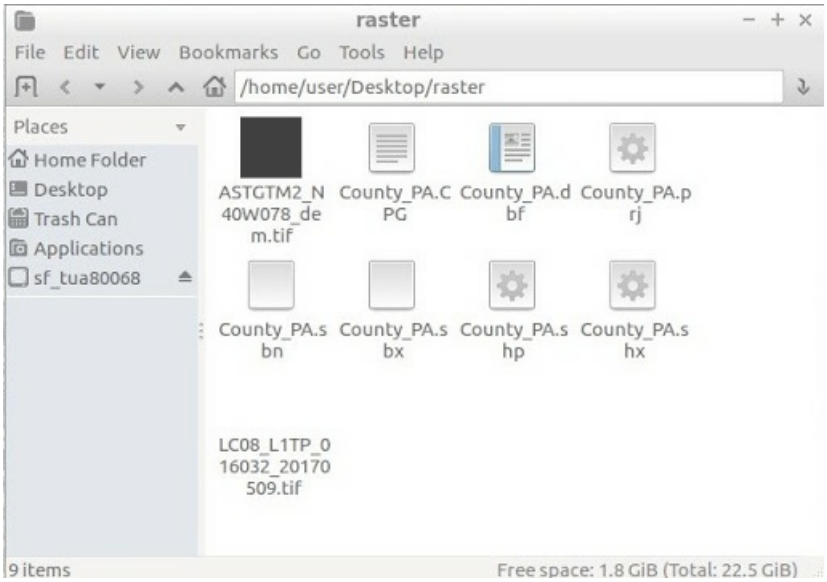
Since the raster2pgsql is compiled as part of PostGIS, most often the raster types supported by raster2pgsql will be the same as those compiled in the GDAL dependency library.

You can use the -G switch to get a list of raster types supported by raster2pgsql

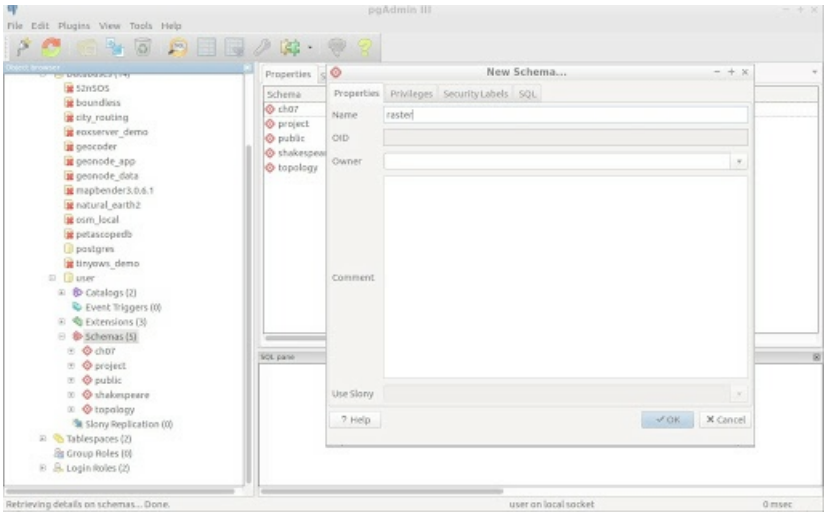
Create a Raster Folder!

Download and unzip `raster_tutorial.zip`
Create folder `/home/user/Desktop/raster`
Place the following items into newly created folder:

- County_PA
- ASTGTM2_N40W078_dem.tif
- LC08_L1TP_016032_20170509.tif



Create a Raster Schema!



Linux

All aboard the Raster Train!

Initiate the Linux Command Line

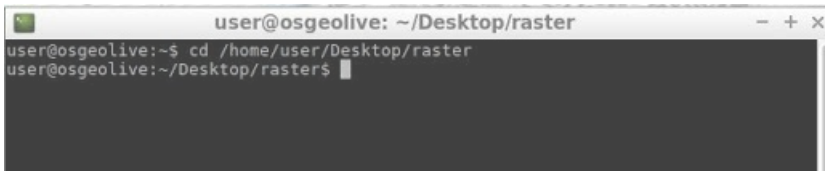
```
LS
```

Before data import, it is good practice to inspect your data in order to be sure that the layers to be added into your database will in uniform Coordinate Reference Systems as well as similar extents. The `gdalinfo` tool can be used for this purpose.

For this tutorial, the RasterBadBoyz have pre-processed the two TIF files, see the appendix for further reading on data processing.

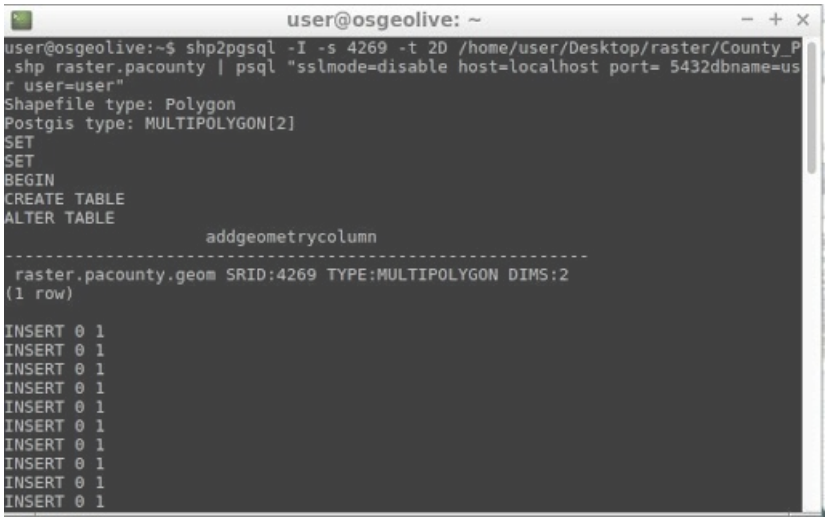
Set your working directory to your raster folder using the Linux command line!

```
cd /home/user/Desktop/raster
```



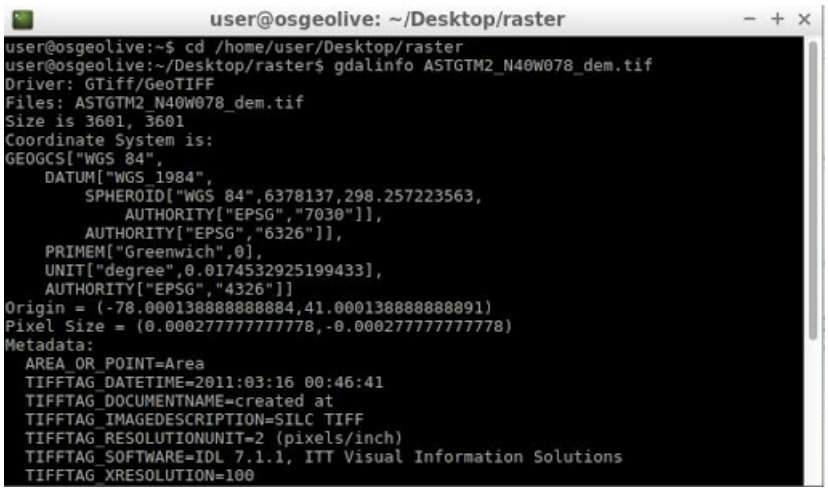
vector pit stop

```
shp2pgsql -I -s 4269 -t 2D /home/user/Desktop/raster/County_PA.shp raster.pacounty | psql
```

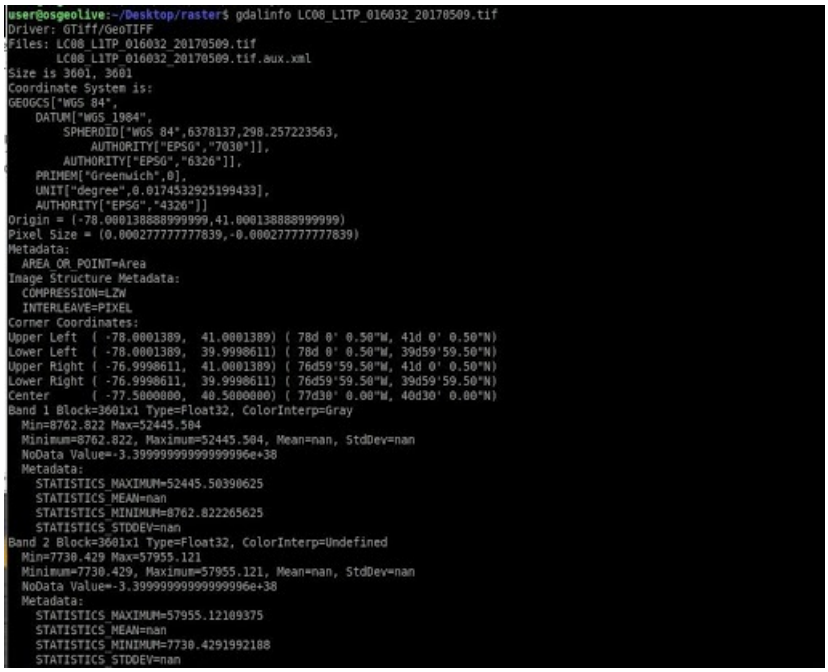


Cool, now lets implement `gdalinfo` to see the metadata.

```
gdalinfo ASTGTM2_N40W078_dem.tif
```




```
gdalinfo LC08_L1TP_016032_20170509.tif
```



Dummy Code ***DO NOT TYPE***

```
raster2pgsql -s 4326 filepath.tif -t tiledimension -F schema.table| psql -d database
```

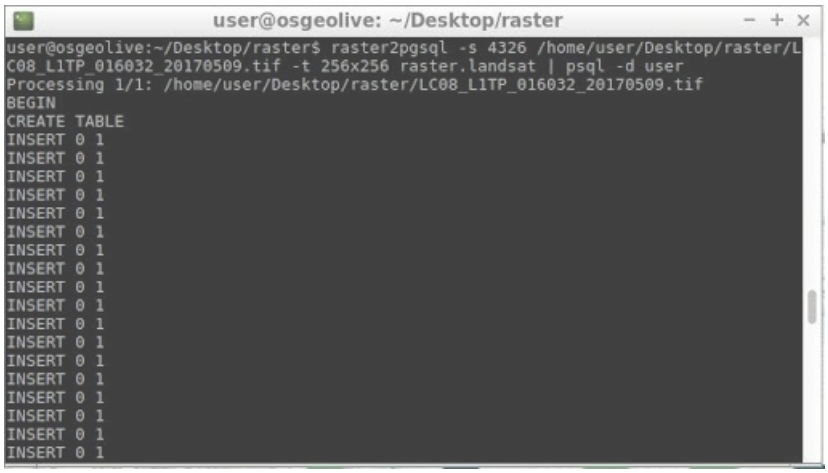
DO TYPE

Loading in the Digital Elevation Model (DEM) for study area using the raster2pgsl in the Linux command line!

```
raster2pgsql -s 4326 /home/user/Desktop/raster/ASTGTM2_N40W078_dem.tif -t 256x256 -F rast
```

Loading in the Landsat8 Multi-Band Raster for study area using the raster2pgsl in the Linux command line!

```
raster2pgsql -s 4326 /home/user/Desktop/raster/LC08_L1TP_016032_20170509.tif -t 256x256 r
```



Notes on Linux Ports

```
-d
```

Drops an existing table with the same name if there is one

```
-I
```

Adds a spatial index

```
-C
```

Enforces constrains - checks that data is registered properly

```
-M
```

Vacuum analyzes a table

```
-R
```

*If included, then the actual values of the raster are not placed in database but rather ‘registered’ and then read from the original file on disk when required

```
-s 4326
```

The SRID (Coordinate Reference System), usually an EPSG code

```
-t 256X256
```

The number of pixels per tile

```
-F
```

Inserts filename

Performing Spatial Queries

```
--- set searchpath
set search_path to raster, public;
show search_path;
```

Digital Elevation Model & Vector Data

```
--- return the metadata
SELECT rid, (foo.md).*
FROM (SELECT rid, ST_metadata(rast) AS md
FROM demrast256) AS foo;
```

Now onto some more advanced raster manipulations...

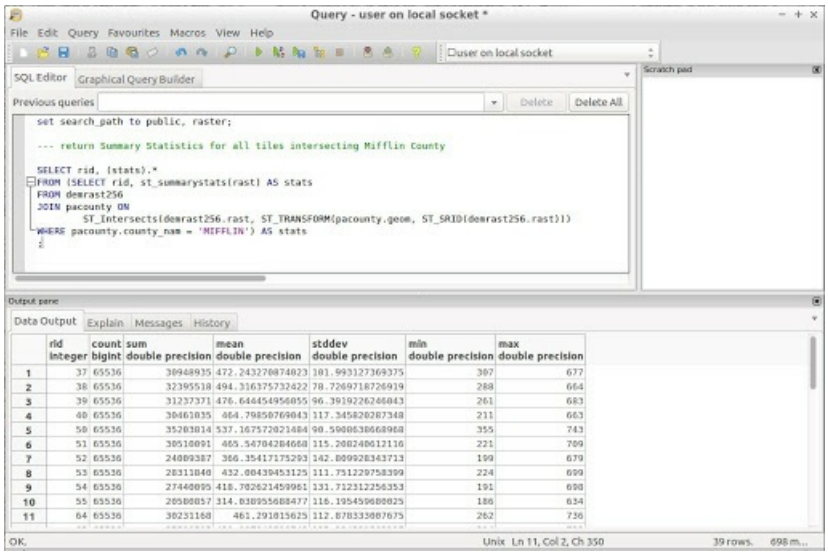
```
--- return stats of individual 256 resolution tiles

SELECT rid, (stats).*
FROM (SELECT rid, st_summarystats(rast) AS stats
FROM demrast256
) AS stats
;
```

What's a tile? 3600 by 3600? What did we just do?

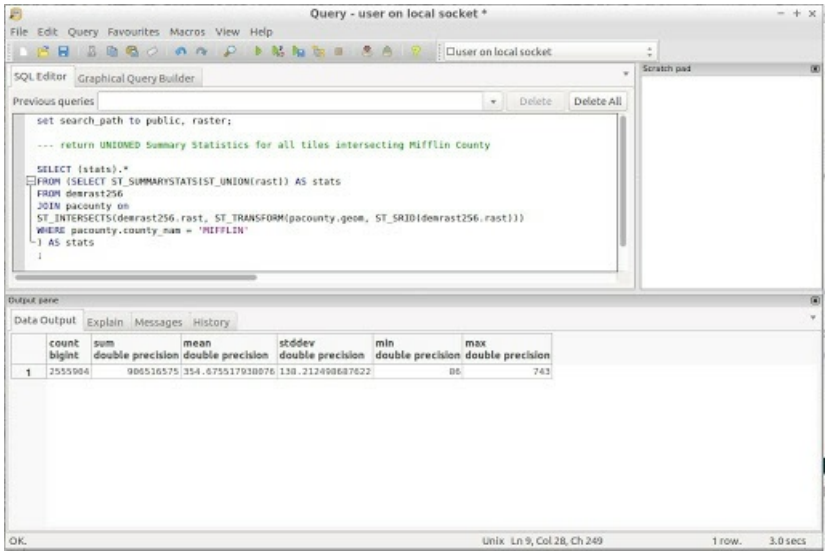
```
--- return stats of individual 256 res tiles intersecting a county

SELECT rid, (stats).*
FROM (SELECT rid, ST_SUMMARYSTATS(rast) AS stats
FROM demrast256
JOIN pacounty ON
ST_INTERSECTS(demrast256.rast, ST_TRANSFORM(pacounty.geom, ST_SRID(demrast256.rast)))
WHERE pacounty.county_nam = 'MIFFLIN') AS stats
;
```




```
--- return UNIONED Summary Statistics for all tiles intersecting Mifflin County

SELECT (stats).*
FROM (SELECT ST_SUMMARYSTATS(ST_UNION(rast)) AS stats
FROM demrast256
JOIN pacounty on
ST_INTERSECTS(demrast256.rast, ST_TRANSFORM(pacounty.geom, ST_SRID(demrast256.rast)))
WHERE pacounty.county_nam = 'MIFFLIN'
) AS stats
;
```




```
--- clipped and unioned

SELECT (stats).*
FROM (SELECT ST_SUMMARYSTATS(st_union(st_clip(demrast256.rast, pacounty.geom))) AS stats
FROM demrast256
JOIN pacounty ON
ST_INTERSECTS(demrast256.rast, ST_TRANSFORM(pacounty.geom, ST_SRID(demrast256.rast)))
WHERE pacounty.county_nam = 'MIFFLIN'
) AS stats


;
```

Multi-Band Raster:
Normalized Vegetation Index

```
set search_path to raster, public;
show search_path;


select *
from landsat
limit 10


SET postgis.gdal_enabled_drivers = 'ENABLE_ALL';
--- SET postgis.gdal_enabled_drivers = 'DISABLE_ALL';

SELECT short_name, long_name
FROM ST_GdalDrivers()
Order by short_name


SELECT rid, ST_NumBands(rast) AS numbands, filename AS filename
FROM landsat
LIMIT 10
;


--- select band metadata
SELECT rid , (foo.md).*
FROM (SELECT rid, ST_bandmetadata(rast) AS md
FROM demrast256 WHERE rid =2) AS foo
;

--- check the geometry type
SELECT ST_Geometrytype(geom) AS result
FROM pacounty
;

--- convert the SRID of county
ALTER TABLE pacounty
ALTER COLUMN geom TYPE
geometry(MultiPolygon,4326)
USING ST_Transform(geom,4326)
;


--- create a new table. necessary when creating to keep the primary key
CREATE TABLE raster.intersects AS
SELECT a.rast, b.COUNTY_NAM
FROM demrast256 AS a, pacounty AS b
WHERE ST_Intersects(a.rast, b.geom) AND b.COUNTY_NAM ILIKE 'mifflin'
;


--- Add serial primary key
ALTER TABLE raster.intersects
```

```
ADD COLUMN rid SERIAL PRIMARY KEY
;

--- add spatial index
CREATE INDEX idx_intersects_rast_gist ON raster.intersects
USING gist (ST_ConvexHull(rast));

--- add constraints
SELECT AddRasterConstraints('raster'::name, 'intersects'::name,'rast'::name)
;

--- reconstituiting the original files from the tiles
Create TABLE raster.dem_reconstitued AS
Select filename,
COUNT(rast) As num_tiles,
ST_union(rast) as rast
FROM demrast256
Group by filename;
alter table raster.dem_reconstitued
add column rid SERIAL PRIMARY KEY
;


CREATE INDEX idx_dem_reconstitued_rast_gist ON raster.dem_reconstitued
USING gist (ST_ConvexHull(rast));

SELECT AddRasterConstraints('raster'::name, 'dem_reconstitued'::name,'rast'::name);

SELECT *
FROM dem_reconstitued

Create TABLE raster.intersects_rec AS
SELECT a.rast, b.COUNTY_NAM
FROM raster.dem_reconstitued AS a, raster.pacounty AS b
WHERE ST_Intersects(a.rast, b.geom) AND b.COUNTY_NAM ilike 'mifflin';
ALTER TABLE raster.intersects_rec
ADD COLUMN rid SERIAL PRIMARY KEY;

--- clip

CREATE TABLE raster.clip AS
SELECT ST_Clip(a.rast, b.geom, true), b.COUNTY_NAM
FROM raster.demrast256 AS a, raster.pacounty AS b
WHERE ST_Intersects(a.rast, b.geom) AND b.COUNTY_NAM LIKE 'MIFFLIN';

ALTER TABLE raster.clip
ADD COLUMN rid SERIAL PRIMARY KEY;

--- clip landsat

CREATE TABLE raster.landsat_mifflin AS
SELECT ST_Clip(a.rast, b.geom, true), b.COUNTY_NAM
FROM raster.landsat AS a, raster.pacounty AS b
WHERE ST_Intersects(a.rast, b.geom) AND b.COUNTY_NAM like 'MIFFLIN';


--- create primary key
```

```
ALTER TABLE raster.landsat_mifflin
ADD COLUMNrid SERIAL PRIMARY KEY;


--- add constraints
SELECT AddRasterConstraints('raster'::name, 'landsat_mifflin'::name,'st_clip'::name);

SELECT *
FROM landsat_mifflin
LIMIT 10

--- map algebra. Calculate NDVI
CREATE TABLE raster.mifflin_ndvi AS
WITH r AS (
SELECT a.rid,st_clip(a.rast, b.geom,true) AS rast
FROM raster.landsat AS a, raster.pacounty AS b
WHERE b.county_nam ILIKE 'mifflin' AND ST_Intersects(b.geom,a.rast)
)
SELECT
r.rid,ST_MapAlgebra(
r.rast, 2,
r.rast, 5,
'([rast2.val] - [rast1.val]) / ([rast2.val] + [rast1.val])::float','32BF'
) AS st_clip
FROM r;

--- create primary key
ALTER TABLE raster.mifflin_ndvi
ADD COLUMN rid2 SERIAL PRIMARY KEY;


--- create index
CREATE INDEX idx_mifflin_ndvi_rast_gist ON raster.mifflin_ndvi
USING gist (ST_ConvexHull(st_clip));

SELECT *
FROM mifflin_ndvi


--- add constraints
SELECT AddRasterConstraints('raster'::name, 'mifflin_ndvi'::name,'st_clip'::name);
```

fin.

Additional Readings

PostGIS
http://www.cef-cfr.ca/uploads/Membres/PierreRacine_FOSS4G_2011.pdf

https://duncanjg.wordpress.com/2012/11/20/the-basics-of-postgis-raster/**

https://www.systutorials.com/docs/linux/man/1-tiffcp/

http://blog.light42.com/wordpress/wp-content/uploads/2012/11/postgis_raster.pdf

DEM Sources
https://gisgeography.com/free-global-dem-data-sources/

Landsat
https://earthexplorer.usgs.gov/