Knowledge Transfer and Tool Development (KTTD): 2B-2024

DETAILS:

The following describes the structure of the package in more detail. A quick start guide, along with short descriptions of each method within the package, and links to the user documentation with more comprehensive descriptions are provided.

Quick Start

Installation

The sgsPy package is deployed on the python package index (PyPi), as such it can be installed using the pip command.

Installation command: 'pip install sgspy'

Files

Some helpful files to get started with can be found in the https://github.com/jbmeyer2001/sgsPy/tree/main/tests/files folder. Specifically mraster.tif, inventory_polygons.shp along with its associated .dbf, .prj, and .shx files, access.shp along with its associated .dbf, .prj, and .shx files, existing.shp along with its associated .dbf, .prj, and .shx files.

Raster and Vector data

The two classes used by the stratification, sampling, and calculating methods are the sgspy.SpatialRaster [SpatialRaster documentation] and sgspy.SpatialVector [SpatialVector documentation]. The beginning of any process using the sgspy package will involve creating an instance of one or more of these classes. This can be done by providing a file path. It can also be done by converting an object from a popular geospatial package.

```
import sgspy
#creating SpatialRaster and SpatialVector using file paths
rast = sgspy.SpatialRaster("mraster.tif")
vect = sgspy.SpatialVector("access.shp")

#plotting
rast.plot(band=0)
rast.plot(band="pzabove2")

#accessing underlying data as a numpy array
zq90_arr = rast.band("zq90")
first_band_arr = rast.band(0)

#to/from GDAL raster dataset
ds = gdal.Open("mraster.tif")
rast = sgspy.SpatialRaster.from_gdal(ds)
new_ds = rast.to_gdal()

#to/from rasterio dataset
```

```
ds = rasterio.open("mraster.tif")
rast = sgspy.SpatialRaster.from_rasterio(ds)
new_ds = rast.to_rasterio()

#to/from geopandas
gdf = gpd.read_file("inventory_polygons.shp")
vect = sgspy.SpatialVector.from_geopandas(gdf)
new_gdf = gpd.to_geopandas(vect)
```

Example

More extensive examples on how to use each method is available in the user documentation. The following shows how one might use three of the different methods available in sgspy to create a new sampling network.

```
import sgspy

# create SpatialRaster object from desired tif file
rast = sgspy.SpatialRaster("mraster.tif")

# use principal component analysis for dimensionality reduction (to two components or raster layers)
pca = sgspy.calculate.pca(rast, num_comp=2)

# stratifiy the two components into 5 equally sized quantiles, and map the two layers into a single mapped output
srast = sgspy.stratify.quantiles(pca, quantiles={"comp_1":5, "comp2":5}, map=True)

# use stratified random sampling to sample each (of the 25 total) mapped strata proportional to it's number of pixels in the raster band
# additionaly, plot the output and write it to samples.shp
samples = sgspy.sample.strat(srast, band="map_strat", num_samples=200, num_strata=25, allocation="prop", plot=True, filename="samples.shp")
```

Methods: Calculating

There are two 'calculating' methods, both designed to assist in the SGS process: pca() and distribution(). The pca() function conducts principal component analysis, and the distribution() function calculates the distribution of pixels in a raster, placing them into histogram bins.

Docs for pca(): https://jbmeyer2001.github.io/sgsPy/group_user_pca.html

Docs for distribution(): https://jbmeyer2001.github.io/sgsPy/group_user_distribution.html

Methods: Stratification

There are four stratification methods: breaks(), map(), poly(), and quantiles(). For all stratification functions, the output will be an sgspy.SpatialRaster which has an unsigned integer pixel type. The first stratification value will be '0', and the following will increase in steps of 1 ('1', '2', '3'). The breaks() method works by stratifying the input raster according to user defined break values. The map() method maps multiple input stratifications into a single output stratification. The poly() method stratifies a vector of polygons according to attribute values. The quantiles() method stratifies a raster depending on desired quantile increments, and utilizes a method for stream computation.

Docs for breaks(): https://jbmeyer2001.github.io/sgsPy/group_user_breaks.html

Docs for map(): https://jbmeyer2001.github.io/sgsPy/group__user__map.html

Docs for poly(): https://jbmeyer2001.github.io/sgsPy/group_user_poly.html

Docs for quantiles(): https://jbmeyer2001.github.io/sgsPy/group_user_quantiles.html

Methods: Sampling

There are four sampling methods: clhs(), srs(), strat(), systematic(). All sampling methods take input raster parameters alongside optional additional auxiliary inputs such as access networks or existing sampling networks, and output sgspy.SpatialVector objects which contain the sample points. The clhs() method conducts conditioned latin hypercube sampling on the input raster bands. The srs() method conducts simple random sampling. The strat() method conducts stratified random sampling. The systematic() method conducts systematic sampling.

Docs for clhs(): https://jbmeyer2001.github.io/sgsPy/group_user_clhs.html

Docs for srs(): https://jbmeyer2001.github.io/sgsPy/group_user_srs.html

Docs for strat(): https://jbmeyer2001.github.io/sgsPy/group_user_strat.html

Docs for systematic(): https://jbmeyer2001.github.io/sgsPy/group_user_systematic.html

TECH TRANSFER:

As discussed, a comprehensive set of user and developer documentation has been made open to facilitate the easy usage and maintenance of the package. Source code has also been made public alongside the documentation in this GitHub repository.