isatis.py

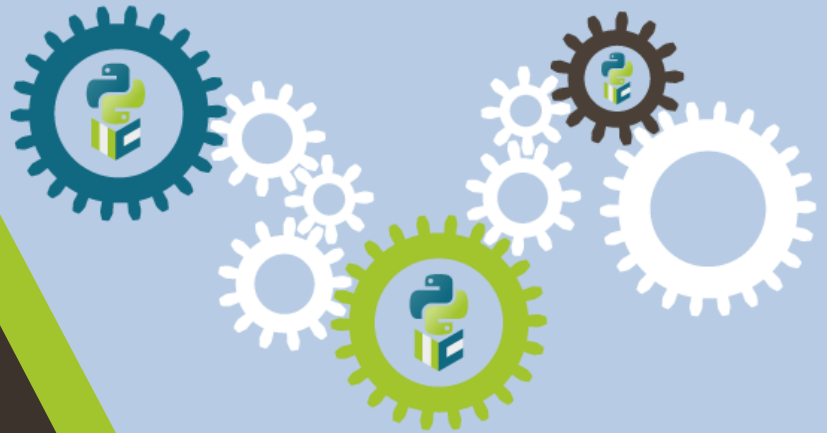Geovariances Python package for geostatistics

COMBINE THE POWER OF ISATIS.NEO WITH THE PYTHON ENVIRONMENT TO BUILD AND AUTOMATE YOUR GEOSTATISTICAL MODELING WORKFLOWS.

Isatis.py is the Geovariances Python geostatistical library that empowers geoscientists in mining, energy, and natural resource sectors to solve their modeling challenges.

ELEVATE YOUR MODELING WORKFLOWS WITH GEOVARIANCES PREMIUM GEOSTATISTICS

Isatis.py offers the robust and technologically advanced geostatistical algorithms found in Isatis.neo. This ensures that you obtain identical outcomes whether you use the software package or the library, thus facilitating testing. **Isatis.py** enables data loading and preparation, variographical analysis, interpolation, estimation with kriging, mineral resource modeling using Uniform Conditioning, and uncertainty analysis through conditional simulations. The library also allows loading and writing data directly from and in the Isatis.neo database.

INCREASE USER EFFICIENCY

Isatis.py benefits from the Python ecosystem and gives direct access to a vast range of Python libraries, including data science, artificial intelligence, and display functions. Developers can build and automate custom solutions and address specific requirements, even complex, more efficiently. **Isatis.py** uses a common language, making workflows easy to maintain and upgrade. It can be used separately in Python and Jupyter environments or integrated into software solutions on both Windows and Linux operating systems.

EASILY HANDLE BIG DATASETS

Isatis.py is parallelized and multithreaded, which makes it a particularly high-performance library that quickly manages datasets and models with millions of cells.

Release 1.1

Data preparation

– Data import to Pandas containers
– Desurveying, declustering, compositing
– Mesh import and selection from meshes
– Advanced neighborhood
– Anamorphosis
– Local anisotropies
– Principal Component Analysis (PCA), Projection Pursuit Multivariate Transform (PPMT)

Geostatistics

– Multivariate experimental variography and automatic fitting
– Block support correction
– Local parameter fitting
– (Co-)kriging, possibly with local anisotropies
– Rescaled (co-)kriging, kriging with measurement errors
– Sampling density variance
– Uniform Conditioning
– Continuous Simulations: Sequential Gaussian (SGS), Turning Bands (TBS), Direct Block (DBS), Stochastic Partial Differential Equations (SPDE)
– Categorical simulations: Sequential Indicators (SIS), Plurigaussian (PGS)

Graphical outputs

– Anamorphosis
– Variogram
– 2D maps and cross-sections

Technical specifications

– Can be used in a standalone Python or a Jupyter environment or integrated into third-party software solutions.
– Provided as a .whl file.
– Available on Windows 10 or Linux RHEL 8+, 64 bits.
– Compatible with Python 3.7, matplotlib, h5py, and ezdxf.
– Uses Pandas containers or HDF5 files as input and output.

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import isatis as isa
plt.rcParams['figure.figsize'] = [7, 7]

### Load and print conditioning data ###

data = pd.read_csv("./data/carajas.csv", sep=',')
coords = data[['X','Y','Z']].to_numpy()
values_Fe = data[['Fe']].to_numpy()
plt.scatter(coords[:,0], coords[:,1], c = values_Fe.flatten(), cmap='viridis')
plt.show()
```



```python
### Compute experimental variograms and fit models ###

indb = isa.DbPandas(df=data)

runner = isa.Variogram()
runner.set_data(indb, coors = ['X', 'Y', 'Z'], invars = ['Fe'])
vexp = runner.variogram(hndir = 2, hlag = 100., hmaxdist = 1000., vlag = 15., vmaxdist = 150.)

model = runner.fitting(var_exp = vexp)

isa.display_fitting(vexp, model, variances = 1.)
```



```python
### Create output grid and perform kriging ###

runner = isa.Kriging()
runner.set_input_data(indb, coors = ['X', 'Y','Z'], invars = ['Fe'])

grid = isa.GridGeom(origin = [500, 0, 400], cellsize = [25., 25., 15.], nxyz = [100, 100, 20])
outdb = isa.DbPandas(grid = grid)
runner.set_output_data(outdb, disc = [5, 5, 1])

db_res = runner.kriging(model = model)

isa.display_grid(dbgrid = db_res, names = ['Fe_OK'], iz = 9, vmin = 0, vmax = 80, title = "Kriging")
```