Mineral resource estimation with python

AN INTRODUCTION TO PYTHON, PYGSLIB, AND PARAVIEW FOR GEOLOGISTS
The presenter

Adrian Martinez Vargas
CSA Global Senior Resource Geologist

Opengeostat project: http://opengeostat.com

PyGSLIB project https://github.com/opengeostat/pygslib

P.Geo, Ph.D. in Geological Sciences, ISMM Moa. Specialist in Geostatistics (CFSG), Paris Mining School.
B.Eng. Geology, ISMM Moa

Adrian is both a geologist and a geostatistician. He has worked as a consultant since 2002 covering many commodities including gold, copper, nickel, chromium, and raw material for cement industry. He has experience in the application of multiple indicator kriging for resource estimation of gold deposits with high nugget and domaining issues. Additionally, he has experience with non-linear geostatistics and conditional simulations for various applications. Adrian has worked as an Assistant Professor in Cuba and Ethiopia, teaching geology and geostatistics.

Adrian produces open source software for geostatistics and mineral resources in Python, Fortran, and Cython.
The attendees

Your name
Your background and current role

What do you know about Mineral resource estimation, Python, GSLIB, PyGSLIB, Pandas, VTK, and Paraview?

What you would like to get out of this course?
Course Outline

Introduction
- Introduction to mineral resource estimation (MRE). Why using python to do a MRE?
- Introduction to Python: language fundamentals, python distributions, python IDE, and python packages useful to do MRE.
- Hello world in Jupyter Notebook/Lab and importing drillhole tables.
- Introduction to PyGSLIB
- Introduction to Paraview

Mineral resource estimation with PyGSLIB (big exercise with explanations)
- Creating and visualizing drill holes
- Modeling geological units using implicit functions operations
- Drillhole intervals tagging, compositing, declustering, statistical analysis, and variography
- Block modeling and estimation
- Block model validation
- Reporting
Introduction

A NECESSARY OVERVIEW OF MINERAL RESOURCE ESTIMATION
The mineral resource estimation (MRE)

Informing data (Drilling) → Geological modeling → Statistical Analysis → Interpolation and validation → Reporting and Auditing

Recurrence

- Assay
- Survey
- Collar
The mineral resource estimation (MRE)
The mineral resource estimation (MRE)

Tables → Drill holes → Solids and surfaces → Sample/domains → Blocks/domain

Desurvey → Wireframing → Tagging

GCOS/Swath plots/sections → Estimation parameters and estimated values → Variogram/stats/plots

Validation → Interpolation → Validation

Re-estimate → New data? Testing different approaches? Validation issues?

Compositing → Composites
The mineral resource estimation (MRE)

Advantages of using scripts/macros
1. Repeatability
2. Auditability and peer review
3. Save you time. You can reuse code from other projects.

Why open source tools?
1. No black box policy
2. It is “free”

Alternatives to macros/scripts:
1. Parameter files
2. Just doing manual work

Python Open Source (PyGSLIB)
Python Proprietary (Micromine)
TCL (Surpac)
Datamine macros
Introduction

ALL ABOUT PYTHON IN 10 MINUTES
Introduction to python

- General aspects
  - What is python?
  - What is a python version
  - Python distributions
  - Python integrated development environments (IDE)

- Python programming language
  - Syntax
  - Python objects, classes, functions and modules (pure python and compiled)
  - Built-in data types and data structures (list, dictionaries, tuples, sets, strings)
  - Modules for efficient data structure: numpy and pandas
Introduction to python (general aspects)

- General aspects
  - What is python?
  - What is a python version
  - Python distributions
  - Python integrated development environments (IDE)

Python Distribution

Standard python console showing version and build platform
Introduction to python (general aspects)

- General aspects
  - What is python?
  - What is a python version
  - Python distributions
  - Python integrated development environments (IDE)

Standard python console showing version

Python embedded in Paraview
Introduction to python (general aspects)

- Python integrated development environments (IDE)

Jupyter Lab/Notebook IDE. The best option for interactive computing and scripting. We will use Jupyter Notebook in this course.

Atom IDE (Good for development)
Introduction to python (general aspects)

- Using Anaconda/Miniconda distributions

https://anaconda.org/opengeostat/pygslib

![Anaconda installer screenshot]

- Conda

- Files

- Installers

- conda install -c opengeostat pygslib

win-64/pygslib-0.0.0.3.9.0-py27h39e3cac_0.tar.bz2

linux-64/pygslib-0.0.0.3.9.0-py36hf687e67_0.tar.bz2
Introduction to python (the language)

- Syntax

```python
In [1]:
   # This is a comment
   # This is a multiline string
   # used as a doc string
   a = 1  # an integer
   b = 2  # another integer
   c = a/b  # question, what is c?

   # if statement
   if c==0.5:
       print("this is python 3")
   elif c==0:
       print('this is python 2')
   else:
       pass

this is python 3

In [5]:
   # create a list of block IJK (GSLIB style)
   # The model definition
   nx = 5
   ny = 4
   nz = 3

   ijk = []
   for iz in range(nz):
       for iy in range(ny):
           for ix in range(nx):
               ijk.append(iz*nx*ny + iy*nx + ix)

   print(ijk)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59]
```
Introduction to python (the language)

- Python objects, classes, functions and modules (pure python and compiled)

```python
In [12]:
    # all in python is an object
    print('Hello is a: ', type('Hello'))
    print('1 is a: ', type(1))
    print('1.0 is a: ', type(1.0))

Hello is a: <class 'str'>
1 is a: <class 'int'>
1.0 is a: <class 'float'>

# this is a function
def div(a,b=1):
    """Hi, this is the help, my name is 'doc string' ""
    return a / b

?div    # only valid in Ipython/Jupyter, it shows help
div (1,2) # is 0.5
div (1)   # is 1, b has default value 1
div (b=2, a=1) # passing named parameters
div (a=1, b=0) # this is an error
```
Introduction to python (the language)

- Python objects, classes, functions and modules (pure python and compiled)

```python
In [28]:
class Blockmodel:
    '''This is a docstring for class Blockmodel
    Example
    >>> Blockmodel(nx,ny,nz)
    ...'
    def __init__(self,nx,ny,nz):
        self.the_nx = nx
        self.the_ny = ny
        self.the_nz = nz

a = Blockmodel
b = Blockmodel(nx = 2, ny = 3, nz = 4)

print(a)
print(b)
print('nx : ', b.the_nx)
```

```
In [28]:
import blocks

a = blocks.Blockmodel
b = blocks.Blockmodel(nx = 2, ny = 3, nz = 4)

print(a)
print(b)
print('nx : ', b.the_nx)
```

Save class, functions, and any object in file block.py to create a pure python module.
Introduction to python (the language)

- Built-in data types and data structures (list, tuples, and dictionaries)

```python
In [31]:
a = [1,2,3,4] # this is a list
b = [[1,2],
     [3,4]] # this is a list of lists
c = (1,2,3,4) # this is a tuple
d = {
     # this is a dictionary
     'I am a Key': 'I am data',
     'X': [1,2,3,4],
     'Y': [1,2,3,4],
     }

a[0]= 1
b[1]= [3, 4]
b[1][1]= 4
d['X']= [1, 2, 3, 4]
```
Introduction to python (the language)

- Modules for efficient data structures: numpy and pandas

  Numpy defines arrays for efficient numerical operations (similar to lists)

```python
import numpy as np
a = np.array ([[1,2,3], [4,5,6]], dtype = float)
print (a)
print (a.shape)
print (a.ndim)
print (a + 10)
```

```
[[ 1.  2.  3.]
 [ 4.  5.  6.]]
2
[[11. 12. 13.]
 [14. 15. 16.]]
```

Pandas DataFrames are to handle tables, import csv, do SQL style queries, etc.
It behaves +/- like dictionaries.

```python
In [46]:
import pandas as pd
collar = pd.DataFrame ( { 'BHID': [1,2,3,4,5],
                         'XCOLLAR': [10,45,20,28,43],
                         'YCOLLAR': [43,47,91,53,76],
                         'ZCOLLAR': [0,0,0,0,0] })
collar[['BHID', 'XCOLLAR']]
```

```
   BHID  XCOLLAR
0      1.0  10.0
1      2.0  45.0
2      3.0  20.0
3      4.0  28.0
4      5.0  43.0
```

Columns are numpy arrays
try:
```python
>>> collar['XCOLLAR'].values
```

Index
Exercise 1: Hello world in Jupyter Lab

Familiarizing yourself with Jupyter
- Open jupyter notebook
- Rename the file as MRE using python
- Add Markdown cell at the top and add some comments

Exploring Jupyter functionality
- Import pandas as pd, pygslib, numpy as np and matplotlib.pyplot as plt
- Import drillhole tables into pandas.DataFrame objects using function pd.read_csv()
- Explore data using pandas.DataFrame functions head(), describe() and plot collar location using pandas.DataFrame plot() function
- Create a column of log(Au) in table assays
Introduction to Paraview
Introduction to Paraview

Paraview is based on VTK (visualization toolkit)

We will use it to visualize

▪ Points: vtkPolydata (*.vtp, *.vtk)
▪ Surfaces: vtkPolydata
▪ Full block models: vtkImageData (*.vti)
▪ Partial block models: vtkUnstructuredGrid (*.vtu)
▪ Variogram grids and some models: vtkStructuredGrid (*.vts)
Introduction to Paraview

Paraview uses Pipelines with:

- **Sources**: No input signal, only output. Example readers (stl, csv, etc)
- **Filters**: Have input and output signal. Examples, contour filter, tube filter.

Use CTR + Space key to see list of commands available for the object selected. Non-valid filters are in grey.

Use key v to activate query/selection
Introduction to Paraview

Paraview has many types of visualizations, we will mostly use:

- Render View
- SpreadSheet View

Visualizations share data and selections

Important: vtk geometries are composed by points and cells. You can have:

- Point data
- Cell data
Exercise 1b: Plot collars in Paraview
Mineral resource estimation with PyGSLIB

THE BIG EXERCISE EXPLAINED
PyGSLIB

Python package for mineral resource estimation consisting of:

- drillhole module: it creates drillholes, do desurvey, and composites
- blockmodel module: to creates block models, fill wireframes with blocks, etc.
- gslib: wraps modified Fortran code from GSLIB, it implements kriging, declustering,…
- vtktools: to handle and generate wireframes and do exports/imports from/to vtk (and Paraview)
- nonlinear: to do nonlinear geostat with discrete gaussian model
- sandbox: area to test new code before it goes to a module
- Sandbox2: Only available as source code
- Html plot: Plot using bokeh (the plan is using Altair(?)

Drillhole objects are composed by tables (Pandas Dataframes) collar and survey. In addition you may add any number of interval tables:

Collar: may have fields
- BHID: any type
- XCOLLAR, YCOLLAR, ZCOLLAR: type float
- (Optional) LENGTH: type float

Survey: may have fields
- BHID: any type
- AT, AZ, DIP: type float (AT at zero is compulsory)

Interval: may have fields
- BHID: any type
- FROM, TO: type float

Drillholes contains functions to:
- Add or remove interval tables (assay, lithology)
- Merge interval tables
- Validate drillhole and interval tables
- Downhole composite, and key composite
- Desurvey
- Add gaps, Split long intervals
- Export drillholes to VTK
- Utility functions: 'fix_survey_one_interval_err', 'fix_zero_interval', 'txt2intID', collar2table
- Properties: collar, survey, tables, table_names
PyGSLIB (drillholes)

Common operations with drillhole data

Create Drillholes
- Define drillholes with tables
  - Collar
  - Survey
  - IntervalAssay
    - Assay
    - Lithology
    - ...
  - Validate drillhole
  - Validate tables

Visualize and modeling
- Combine tables Assay and Litho
- Desurvey (calculate x,y,z)
- Export to VTK
- Modeling
  - Composite per litho
  - Extract contacts (manual)

Prepare for interpolation
- Tag interval with estimation domains using surfaces and solids
- Composite downhole
Exercise 2: Drillholes

Exploratory data analysis
• Create a drillhole object
• Add drillhole intervals
• Validate
• Desurvey
• Export drillhole as vtk file and identify mineralization

Modeling
• Label drillhole intervals with domain
• Optionally composite with key
• Extract contact points to model surfaces

Composite
• Composite downhole (only samples in mineralized domain)
The mineral resource estimation (MRE)

### Tables
- Desurvey
- Wireframing
- Drill holes

### Solids and surfaces
- Tagging
- Offsets

### Sample/domains Block/domain
- Composites
- Compositing

### New data?
- Testing different approaches?
- Validation issues?

### GCOS/Swath plots/sections
- Validation
- Estimation parameters and estimated values

### Interpolation
- Variogram/stats/plots

### Variography Stats
- Re-estimate

MINERAL RESOURCE ESTIMATION WITH PYTHON
PyGSLIB (vtktools and modeling)

Vtktools is a collection of functions to work with spatial data, to import spatial data (wireframes) and to do some computational geometry.

We use this module to define domains (vtkPolyData) and to save block models (vtkImageData, vtkRectilinearGrid, vtkUnstructuredGrid).

We use this module to select points and blocks inside domains defined by vtkPolyData.

Vtktools contains functions to:

- 'SaveImageData', 'SaveRectilinearGrid', 'SaveUnstructuredGrid', 'define_region_grid', 'grid2vtkImageData', 'partialgrid2vtkfile',
- 'GetPointsInPolydata', 'PolyData2dxf', 'SavePolydata', 'dxf2PolyData', 'delaunay2D', 'SetPointsInPolydata', 'clip_with_surface', 'dmtable2wireframe', 'loadSTL', 'loadVTP', 'rbfinterpolate',
- 'getbounds',
- 'evaluate_implicit_grid', 'implicit_surface', 'evaluate_implicit_points', 'vtk_raycasting', 'pointinsolid', 'pointquering',

Vtk cells
PyGSLIB (vtktools and modeling)

VTK uses implicit functions to do some computational geometry operations, for example:

- Find points inside/outside solids
- Extracting an isosurface
- Cutting with planes or surfaces
- Generating blocks in solid

Surfaces (PolyData) need to be converted to implicit functions.

In, out, and distance directions are calculated with surface normals.

Implicit functions

- Distance to surface (sign tells you if you are inside or outside)
- Normals
- Resolution is important
PyGSLIB (vtktools and modeling)

There are different ways to obtain similar models:

- Cut a VTK Grid (Structured or Unstructured)
- Extracting an isosurface and implicit boolean
- Using open implicit surfaces
- Intrusion (not implemented)

We need closed surfaces to calculate block volume
PyGSLIB (vtktools and modeling)

Implicit Boolean = max(hw,fw)

Hanging wall

Footwall
PyGSLIB (vtktools and modeling)

Raycasting is a non-implicit tool based on rays and intersections with solids.

Is used to find points above/below or within surfaces.

Works with open surfaces

Intersection points

Points below surface
Exercise 3: Model domains

- Define working region (regular grid or vtkImage)
- Create contact surfaces and topo interpolating with Rbf (vtkPolyData with open surface triangulations)

Create solids
- Convert vtkPolyData surfaces to implicit functions
- Model solids:
  - Use cutting tool (not recommended)
  - Or evaluate distance in region and extract isosurface

Tag drillhole data
- Tag drillhole data is not necessary in this case, but you will have to do this if using solids created in a different way. There are many ways of tagging, example:
  - `pygslib.vtktools.evaluate_implicit_points()`
  - `pygslib.vtktools.pointinsolid()`
  - `pygslib.vtktools.vtk_raycasting()`
The mineral resource estimation (MRE)

- Tables
- Drill holes
  - Desurvey
  - Wireframing
- Solids and surfaces
  - Tagging
- Sample/domains
  - Blocks/domain
  - Compositing
  - Composites

- New data?
- Testing different approach?
- Validation issues?

- GCOS/Swath
  - plots/sections
  - Validation
- Estimation parameters and estimated values
  - Interpolation
  - Variogram/stats/plots
- Variography Stats

© OpenGeostat/CSA Global all rights reserved
Blockmodel class creates and handle block models. Blocks models are defined by origin of coordinates (lower left corner), block size, and number of blocks. Block values are stored in Pandas DataFrames.

We use this module to define a block model object, calculate percentage of blocks inside domain, and to export blocks to vtk.

Other interesting applications are: migrating block data to point data (useful for drillhole spacing studies)

Blockmodel contains functions to:

- 'block2point', 'point2block', 'blockinsurface',
- 'blocks2vtkImageData', 'blocks2vtkUnstructuredGrid', 'blocks2vtkUnstructuredGrid_p',
- 'reblock', 'set_block_size', 'set_blocks', 'set_origin', 'set_rcl',
- 'calc_ijk', 'calc_ixyz_fromijk', 'calc_ixyz_fromxyz', 'calc_xyz_fromixyz', 'fillwireframe', 'create_3Dgrid', 'create_IJK', 'delete_blocks',
- Properties: 'dx', 'dy', 'dz', 'xorg', 'yorg', 'zorg', 'bmtable', 'nx', 'ny', 'nz',
PyGSLIB (blockmodel)

- In Pygslib a block model can be full or partial.
- It is fully defined by coordinates of the lower left corner, number of blocks and block size.
- Block positions can be defined with an IJK index that grows in X, then Y and finally Z direction. This is similar to GSLIB and different to Datamine IJK value.
- The class includes functions to calculate block row, col, and level index and coordinates from ijk and vice versa.
- The function `fillwireframe()` generates a full model and calculates percent inside closed surface (or horizontal section polylines).
- Subcells are not implemented yet but can be handled. You could use functions in vtktools to inspect subcell centroids position relative to surfaces and solids.
Exercise 4: Block models

- Generate a block model with percentage inside solid and validate the results in Paraview
The mineral resource estimation (MRE)

Tables → Drill holes → Solids and surfaces → Sample/domains → Blocks/domain

Desurvey → Wireframing → Tagging → Compositing

New data? Testing different approaches? Validation issues?

GCOS/Swath plots/sections → Validation → Estimation parameters and estimated values → Variogram/stats/plots

Re-estimate → Interpolation → Variography Stats

Composites
This module interfaces Fortran code:

- modified from gslib source files (gslib77_ls.tar.gz)
- new Fortran code, inspired or not in gslib (example to read datamine files)

Two main changes were introduced in the code:

- GSLIB code was reorganized as safe functions
- Data is directly transferred from python through memory (for that reason we use numpy arrays)
- Data and parameters are defined as dictionaries.
- Improvements in the programs: variogram cloud and domain, etc.

Hidden internal functions with direct access to code in Fortran, for example '__bigaus' and '__block_covariance'.

Python interface to low level functions:

- 'kt3d', 'postik',
- 'addcoord', 'rotcoord', 'rotscale',
- 'dm2csv_ep', 'dm2csv_sp', 'read_gslib_file',
- 'block_covariance', 'check_gamv_par',
- 'cova3', 'gam', 'gamv'
- 'setrot', 'set_nan_value'
- 'cdfplt', 'histgplt',
- 'declus'

Some are under construction or need redesign:
- 'gamv3D', 'cova3', 'gam', 'gamv'

Not all the gslib programs are included
PyGSLIB (gslib & htmlplot)

Programs for exploratory data analysis:

- **declus**: Declustering
- **plothtml.histgplt**: plot histograms
- **plothtml.probplt**: plot cdf and pdf
- **plothtml.qqplt**: plot qq plot
- **plothtml.scatter2D**: plot scatterplot and allows selection (no gslib)

To use bokeh with Jupyter Lab you may install the extension (this may be difficult to do!):

```
c:\> jupyter labextension install jupyterlab_bokeh

To use bokeh with Jupyter Lab you may install the extension (this may be difficult to do!):
```

```
c:\> conda install -c conda-forge nodejs
```
PyGSLIB (gslib declustering)

Clustered

Clustered

Primary

0.2

4

True

Clustered

Non declustered

Declustered

True
PyGSLIB (gslib declustering)

Delustered
PyGSLIB (gslib declustering)

Optimizing cell size

Optimum cell size
PyGSLIB (gslib declustering)

Declustering parameter dictionary

```python
#declustering parameters
parameters_declus = {
    'x' : mydholedb.table['CMP'].loc[mydholedb.table['CMP']['Domain']==1, 'xm'],
    'y' : mydholedb.table['CMP'].loc[mydholedb.table['CMP']['Domain']==1, 'ym'],
    'z' : mydholedb.table['CMP'].loc[mydholedb.table['CMP']['Domain']==1, 'zm'],
    'vr' : mydholedb.table['CMP'].loc[mydholedb.table['CMP']['Domain']==1, 'CU'],
    'anisy' : 1.,
    'anisz' : 0.05,
    'minmax' : 0,
    'ncell' : 100,
    'cmin' : 100.,
    'cmax' : 5000.,
    'noff' : 8,
    'maxcel' : -1
}

# declustering
wtopt, vrop, wtmin, wtmax, error, \
xinc, yinc, zinc, rxcs, rycs, rzcs, rvrcr = pygslib.gslib.declus(parameters_declus)
```
Exercise 5: Stats and variography

- Composite
- Declustering
- Stats
- Variography (not implemented but explained)
The mineral resource estimation (MRE)

- Tables
- Drill holes
- Solids and surfaces
- Sample/domains
- Blocks/domain
- GCOS/Swath plots/sections
- Desurvey
- Wireframing
- Tagging
- Compositing
- Variography
- Stats
- Variogram/stats/plots
- Variography Stats
- Estimation parameters and estimated values
- Re-estimate
- New data?
- Testing different approaches?
- Validation issues?
- Validation
- Interpolation
PyGSLIB (gslib interpolation)

Interpolation overview

- Uses the function kt3d from GSLIB : pygslib.gslib.kt3d
- It support:
  - Ordinary/universal kriging
  - Inverse of the distance
  - Median indicator kriging
  - Multiple indicator kriging is possible using function pygslib.gslib.postik
- support for maximum number of samples per drillhole
- The output is a dictionary with:
  - output : dict, estimation results at target points/blocks
  - debug : debug output for the last block estimated
  - estimate : estimation summary

Type `help(pygslib.gslib.kt3d)` to see full help

Kriging, inverse of the distance and nearest neighbor can be written as:

$$ Au(v) = \sum_{i=1}^{n} \lambda_i Au(x_i) $$

The only difference is how you calculate the weights $\lambda_i$
Exercise 6: Interpolation

- Interpolate
The mineral resource estimation (MRE)

Tables

Drill holes

Desurvey

Wireframing

Solids and surfaces

Tagging

Sample/domains

Blocks/domain

Composites

Compositing

GCOS/Swath plots/sections

Estimation parameters and estimated values

Validation

Interpolation

Variogram/stats/plots

Variography Stats

Re-estimate

New data?
Testing different approaches?
Validation issues?

New data?
Testing different approaches?
Validation issues?
PyGSLIB (nonlinear)

The nonlinear module implements the discrete gaussian model with hermite polynomials. The main applications are:

- **Global Change Of Support (GCOS).** A tool for global estimation of mineral resources and for estimation validation.
- Normal score variogram transformation. Not implemented yet.
- Uniform conditioning: not implemented yet.
- Normal score transformation (also available in pygslib.gslib)

'ana_options', 'anamor',
'anamor_blk', 'anamor_raw',
'backtr', 'brentq',
'calauthorized',
'calauthorized_blk',
'expand_anamor',
'f_covar_ZvZv', 'f_var_Zv',
'findcontrolpoints', 'fit_PCI',
'get_r', 'get_ro',
'gtcurve', 'norm',
'nscore', 'plotgt',
'recurrentH',
'ttable',
'var_PCI'

Gaussian Anamorphosis in point and block support
PyGSLIB (nonlinear) GCOS

Global estimation in point support

Ore: 20% of total tonees

Waste: 80% of total tonees

Cumulative distribution function

Grade: \[ g = c + \sum_i (g_i \times P_i) \]
PyGSLIB (nonlinear) GCOS

Global estimation in block support

Ore

Waste

Ore: 10% of total tones

Waste: 90% of total tones

Block CDF
PyGSLIB (nonlinear) GCOS

Global estimation in block support

a) Get Point Anamorphosis

b) Fit the block Anamorphosis

c) Use block Anamorphosis To deduce G/T
PyGSLIB Validations

- Global change of support
- Mean (declust.) comparisons
- Visual validations
- Swath plots not implemented but there is a work around...

Visual validations

('Mean in model OK', 0.21166003)
('Mean in model ID2', 0.20810474)
('Mean in model NN', 0.20687895)
('Mean in data', 0.24149141734342264)
('Declustered mean', 0.2251903672467954)
Exercise 7: Validation

- Validate the estimate
Exercise 8: Reporting

• Reporting (Homework). Hint, use pandas dataframes
PyGSLIB what is next?

Pygslib TODO list:

- (documentation) Better documentation
- (nonlinear) Implement Uniform conditioning
- (nonlinear) Add upper and lower tail CDF model fitting tool
- (gslib) complete variogram3d and python interface and create an Ipython interface with widgets for friendly variogram modeling
- (blockmodel) implement grade tonnage report and block cdf
End of presentation
Thank you for participating in our training courses!

Kind regards,

Adrian Martinez Vargas
adrian.martinez@opengeostat.com
Opengeostat and CSA Global
M. +1 647 773 2921

Questions?
• ?????