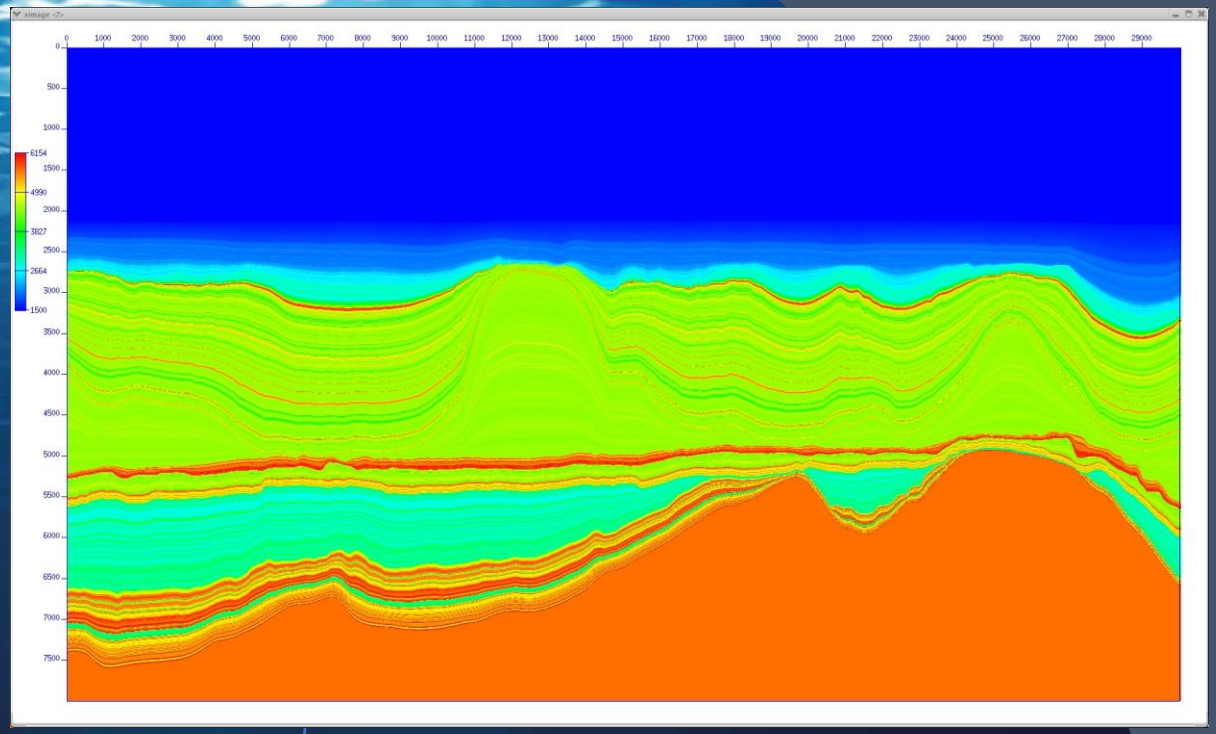
DISCIPLINA

Métodos para análise de grande volume de dados e Astroinformática - 2021/2

Professor: Clécio de Bom

Análise dos dados Sísmicos

(Projeto 12)



Aluno: Eldues Martins – Doutorando (Computação Quântica)

Orientador: Prof. Ivan Santos Oliveira - COMAN

Data:
16/dezembro/2021

Centro Brasileiro de Pesquisas Físicas

CBPF

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÕES

What we did ?

1. Introdução à Python e bibliotecas de manipulação de dados (PANDAS);
2. Estatística descritiva e distribuições de probabilidades em catálogos;
3. Acesso a base de dados Astronômicas; queries
4. Estrutura de dados; fits; hdf; npy, json
5. Análise e visualização Catálogos e imagens
6. Introdução a análise e processamento de imagens
7. Matching de dados astronômicos
8. O método da máxima verossimilhança;
9. Inferência Bayesiana;
10. Simulações de Monte Carlo e MCMC;
11. Introdução ao Aprendizado de Máquina (ML);
12. Regressão e Classificação com ML;

1 . Métodos específicos sobre series temporais temporais (LSTM)

2. Gaussian Process;

3. Data Lakes;

4.Hadoop;

5. Data Lakes

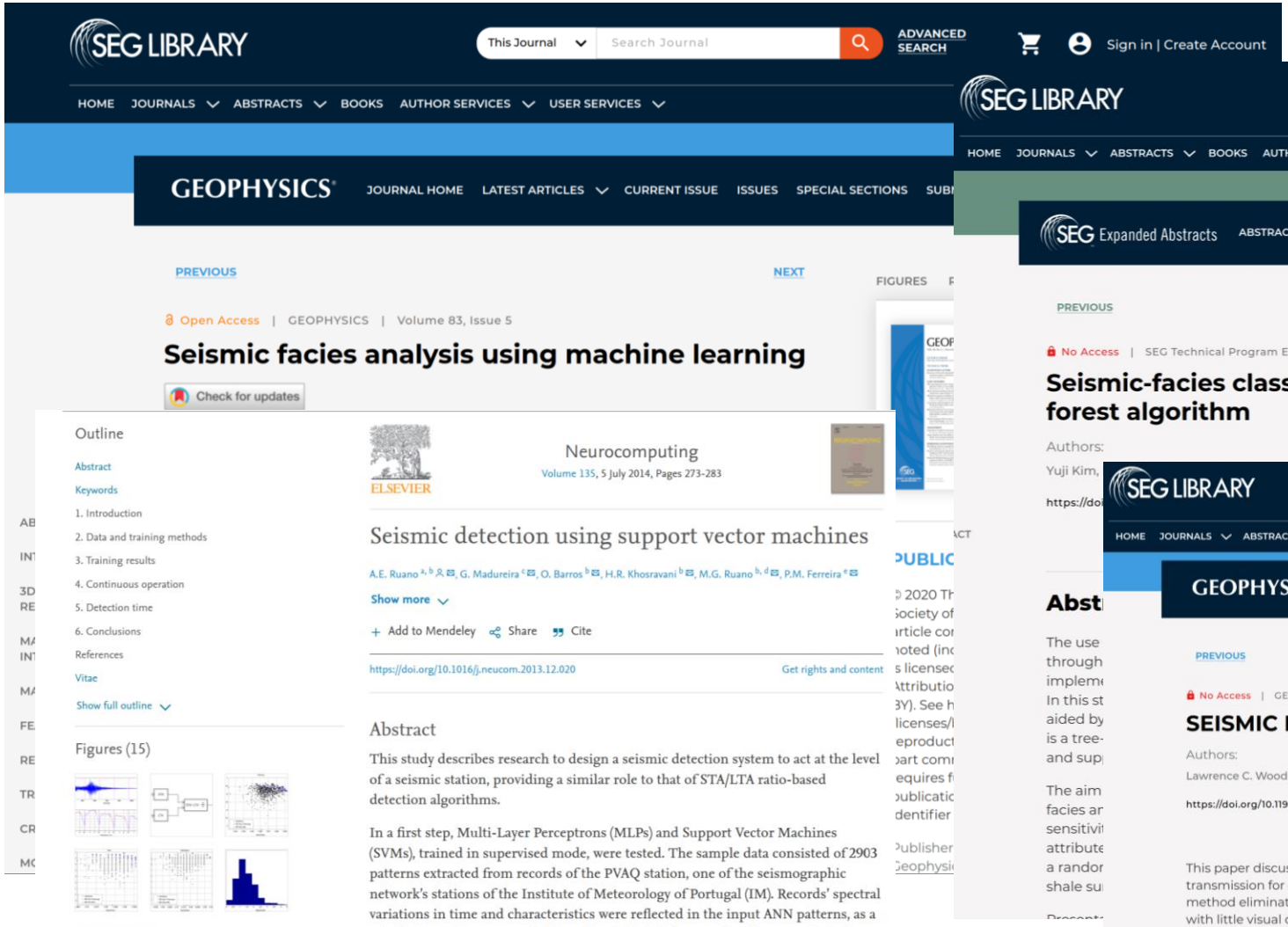
6. + SQL

7. Deep Learning (curso de 1 semestre, espaço para dois)

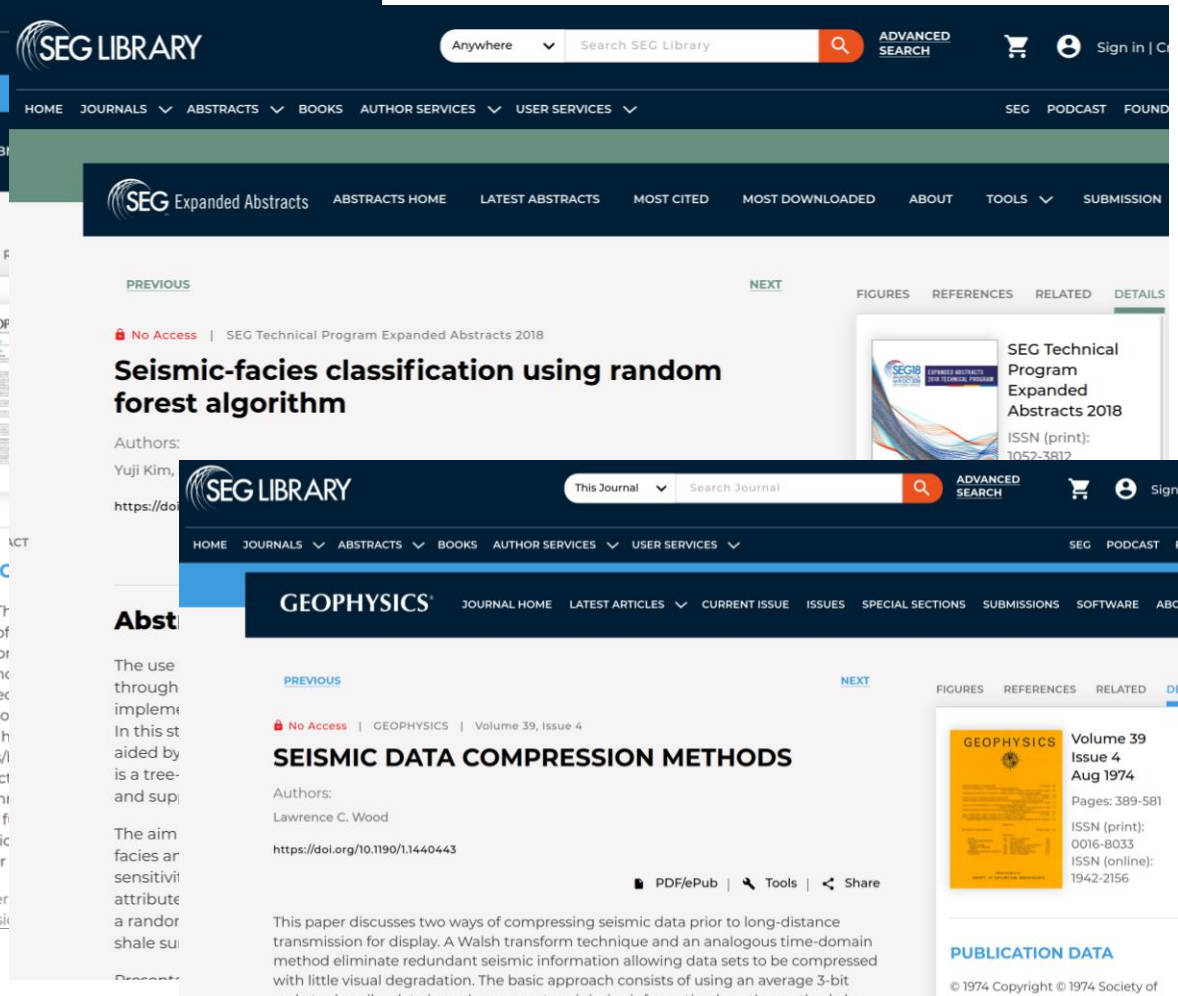
8. Missing data inputation

9. Processamento de Sinais e Imagens (cu

Você


SEG LIBRARY
This Journal
Search Journal
ADVANCED SEARCH
Sign in | Create Account
HOME JOURNALS ABSTRACTS BOOKS AUTHOR SERVICES USER SERVICES
GEOPHYSICS
JOURNAL HOME LATEST ARTICLES CURRENT ISSUE ISSUES SPECIAL SECTIONS
PREVIOUS
NEXT
Open Access | GEOPHYSICS | Volume 83, Issue 5
Seismic facies analysis using machine learning
Check for updates
FIGURES

Outline
Abstract
Keywords
1. Introduction
2. Data and training methods
3. Training results
4. Continuous operation
5. Detection time
6. Conclusions
References
Vitae
Show full outline
Figures (15)
Show all figures
Tables (5)
Table 1
Table 2
Table 3
Table 4
Table 5
ELSEVIER
Neurocomputing
Volume 135, 5 July 2014, Pages 273-283
Seismic detection using support vector machines
A.E. Ruano, G. Madureira, O. Barros, H.R. Khosravani, M.G. Ruano, P.M. Ferreira
Show more
Add to Mendeley
Share
Cite
https://doi.org/10.1016/j.neucom.2013.12.020
Get rights and content
Abstract
This study describes research to design a seismic detection system to act at the level of a seismic station, providing a similar role to that of STA/LTA ratio-based detection algorithms.
In a first step, Multi-Layer Perceptrons (MLPs) and Support Vector Machines (SVMs), trained in supervised mode, were tested. The sample data consisted of 2903 patterns extracted from records of the PVAQ station, one of the seismographic network's stations of the Institute of Meteorology of Portugal (IM). Records' spectral variations in time and characteristics were reflected in the input ANN patterns, as a set of values of power spectral density at selected frequencies. To ensure that all patterns of the sample data were within the range of variation of the training set, we used an algorithm to separate the universe of data by hyper-convex polyhedrons, determining in this manner a set of patterns that have a mandatory part of the training set. Additionally, an active learning strategy was conducted, by iteratively incorporating poorly classified cases in the training set. The proposed system best results, in terms of sensitivity and selectivity in the whole data ranged between 98% and 100%. These results compare very favourably with the ones obtained by the existing detection system, 50%, and with other approaches found in the literature.
Subsequently, the system was tested in continuous operation for unseen (out of sample) data, and the SVM detector obtained 97.7% and 98.7% of sensitivity and selectivity, respectively. The classifier presented 88.4% and 99.4% of sensitivity and selectivity when applied to data of a different seismic station of IM.
Due to the input features used, the average time taken for detection with this


SEG LIBRARY
Anywhere
Search SEG Library
ADVANCED SEARCH
Sign in | C
HOME JOURNALS ABSTRACTS BOOKS AUTHOR SERVICES USER SERVICES
SEG PODCAST FOUND
SEG Expanded Abstracts
ABSTRACTS HOME LATEST ABSTRACTS MOST CITED MOST DOWNLOADED ABOUT TOOLS SUBMISSION
PREVIOUS
NEXT
No Access | SEG Technical Program Expanded Abstracts 2018
Seismic-facies classification using random forest algorithm
Authors:
Yuji Kim,
https://do
FIGURES REFERENCES RELATED DETAILS
SEG Technical Program Expanded Abstracts 2018
ISSN (print):
1052-3812
Abst
The use
through
implem
In this st
aided by
is a tree-
and sup
The aim
facies ar
sensitivit
attribute
a randor
shale su

SEG LIBRARY
This Journal
Search Journal
ADVANCED SEARCH
Sign in | Create Account
HOME JOURNALS ABSTRACTS BOOKS AUTHOR SERVICES USER SERVICES
SEG PODCAST FOUNDATION WIKI
GEOPHYSICS
JOURNAL HOME LATEST ARTICLES CURRENT ISSUE ISSUES SPECIAL SECTIONS SUBMISSIONS SOFTWARE ABOUT
PREVIOUS
NEXT
No Access | GEOPHYSICS | Volume 39, Issue 4
SEISMIC DATA COMPRESSION METHODS
Authors:
Lawrence C. Wood
https://doi.org/10.1190/1.1440443
PDF/ePub | Tools | Share
This paper discusses two ways of compressing seismic data prior to long-distance transmission for display. A Walsh transform technique and an analogous time-domain method eliminate redundant seismic information allowing data sets to be compressed with little visual degradation. The basic approach consists of using an average 3-bit code to describe data in such a way as to minimize information loss; the method also uses the Walsh transform to achieve further compaction through sequency bandlimiting. A second technique is entirely a time-domain operation and does not use transforms. The Walsh method, however, produces larger compression ratios than the time technique before serious image degradation occurs. Both schemes have six basic parts: bandlimiting, quantization, encoding, decoding, interpolation, and band-pass filtering; they differ only in band limiting and interpolation. Band limiting sequencies in the Walsh domain is very similar to, but not the same as, alias filtering and resampling in time. Reducing Walsh bandwidths by some power of two has a time-domain implementation consisting of an averaging procedure with subsequent resampling, while the inverse Walsh transform step can be viewed as a means of interpolating in the time domain. The convergence properties of three Rademacher derived transforms—Hadamard, Paley, and Walsh—are studied with regard to exploration seismic data. Hadamard energy has been found to be uniformly distributed over its entire spectrum, whereas Walsh and Paley transforms concentrate about 80 percent of the total energy into a major lobe occupying about 15 percent of the total bandwidth (2 msec sampling). Smaller minor lobes containing the remaining 20 percent are discarded while bandlimiting. The major lobe energy suffices for many
FIGURES REFERENCES RELATED DETAILS
GEOPHYSICS
Volume 39
Issue 4
Aug 1974
Pages: 389-581
ISSN (print):
0016-8033
ISSN (online):
1942-2156
PUBLICATION DATA
© 1974 Copyright © 1974 Society of Exploration Geophysicists
Publisher:Society of Exploration Geophysicists
History
Received: 02 May 1973
Published: 07 Feb 2012
Published in print: 01 Aug 1974
CITATION INFORMATION
Lawrence C. Wood, (1974), "SEISMIC DATA COMPRESSION METHODS," GEOPHYSICS 39: 499-525.
https://doi.org/10.1190/1.1440443
PLAIN-LANGUAGE SUMMARY

Install User Guide API Examples More Go

scikit-learn

Machine Learning in Python

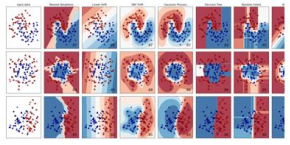
- Simple and efficient tools for predictive d
- Accessible to everybody, and reusable in
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSI

Getting Started | Release Highlights for 1.0 | GitHub

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.
Algorithms: SVM, nearest neighbors, random forest, and more...

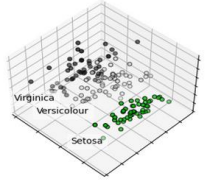


Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency
Algorithms: k-Means, feature selection, non-negative matrix factorization, and more...



Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: SVR, nearest neighbors, random for-

Clustering

Automatic groupin

Applications: Cu
experiment outcor
Algorithms: k-Me

Powered by Dask

These software projects are well-integrated with Dask, or use Dask to power components of their infrastructure.

pandas, NumPy, scikit-learn, Scikit-image, XGBoost, RAPIDS, XArray, Iris, Pangeo, Prefect, Napari, Snorkel, Datashader, Intake, TPOT, MDAnalysis, Stumpy, Featuretools, Cesium-ML, SkyPortal, Conda Forge, DataPrep, LightGBM, Xarray-Spatial, Karthotek, SatPy, Streamz, Scikit-allel, tsfresh

Supported By

ANACONDA, Capital One, Coiled, DARPA, D.E. Shaw & Co, Moore Foundation, NSF, NVIDIA, PREFECT, QuanSight, SaturnCloud

NUMFOCUS

HOME ABOUT GET STARTED COMMUNITY BLOG DOCS GITHUB

RAPIDS

Open GPU Data Science

GET STARTED

GPU DATA SCIENCE

ACCELERATED DATA SCIENCE
The RAPIDS suite of open source software libraries gives you the freedom to execute end-to-end data science and analytics pipelines entirely on GPUs.
Learn about RAPIDS »

SCALE OUT ON GPUS
Seamlessly scale from GPU workstations to multi-GPU servers and multi-node clusters with Dask.
Learn about Dask »

PYTHON INTEGRATION
Accelerate your Python data science toolchain with minimal code changes and no new tools to learn.
Learn about our libraries »

TOP MODEL ACCURACY
Increase machine learning model accuracy by iterating on models faster and deploying them more frequently.
Learn about RAPIDS for model optimization »

REDUCED TRAINING TIME
Drastically improve your productivity with more interactive data science tools like XGBoost.
Learn about XGBoost »
Learn about accelerated ML with cuML »

OPEN SOURCE
RAPIDS is an open source project. Supported by NVIDIA, it also relies on Numba, Apache Arrow, and many more open source projects.
Learn about our projects »

GETTING STARTED

The RAPIDS data science framework is designed to have a familiar look and feel to data scientist working in Python. Here's a code snippet where we read in a CSV file and output some descriptive statistics:

```
import cudf

gdf = cudf.read_csv('path/to/file.csv')
for column in gdf.columns:
    print(gdf[column].mean())
```

Find more details on our Get Started Page »

TRY NOW ONLINE
Jump right into a GPU powered RAPIDS notebook online.
Try with Colaboratory »

10 MINUTES TO CUDF
Modeled after 10 Minutes to Pandas, this is a short introduction to cuDF that is geared mainly for new users.
Go to guide »

MULTI-GPU WITH DASK-CUDF
A short introduction to multi-GPU solutions with a distributed DataFrame via Dask-cuDF.
Go to guide »

EXAMPLE NOTEBOOKS
A Github repository with our introductory examples of XGBoost, cuML demos, cuGraph demos, and more.
Go to repo »

EXAMPLE COMMUNITY NOTEBOOKS
A second Github repository with our extended collection of community contributed notebook examples.
Go to repo »

RAPIDS NEWS

ACCELERATING TF-IDF FOR

The Bokeh Visualization Library

DASK

Dask natively scales Python

Dask provides advanced parallelism for analytics, enabling performance at scale for the tools you love

Learn More | Try Now »

Star

Integrates with existing projects

BUILT WITH THE BROADER COMMUNITY

Dask is open source and freely available. It is developed in coordination with other community projects like NumPy, pandas, and scikit-learn.

NumPy

Dask arrays scale NumPy workflows, enabling multi-dimensional data analysis in earth science, satellite imagery, genomics, biomedical applications, and machine learning algorithms.

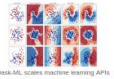
Learn More » | Try Now »

pandas

Dask dataframes scale pandas workflows, enabling applications in time series, business intelligence, and general data munging on big data.

Learn More » | Try Now »

scikit-learn



Dask-ML scales machine learning APIs like scikit-learn and XGBoost to enable scalable training and prediction on large models and large datasets.

Learn More » | Try Now »

Familiar for Python users

AND EASY TO GET STARTED

Dask uses existing Python APIs and data structures to make it easy to switch between NumPy, pandas, scikit-learn to their Dask-powered equivalents.

You don't have to completely rewrite your code or retrain to scale up.

Learn About Dask APIs »

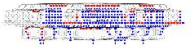
Scale up to clusters

OR JUST USE IT ON YOUR LAPTOP

Dask's schedulers scale to thousand-node clusters and its algorithms have been tested on some of the largest supercomputers in the world.

But you don't need a massive cluster to get started. Dask ships with schedulers designed for use on personal machines. Many people use Dask today to scale computations on their laptop, using multiple cores for computation and their disk for excess storage.

Learn About Dask Schedulers »



Customizable

ENABLING YOU TO PARALLELIZE INTERNAL SYSTEMS

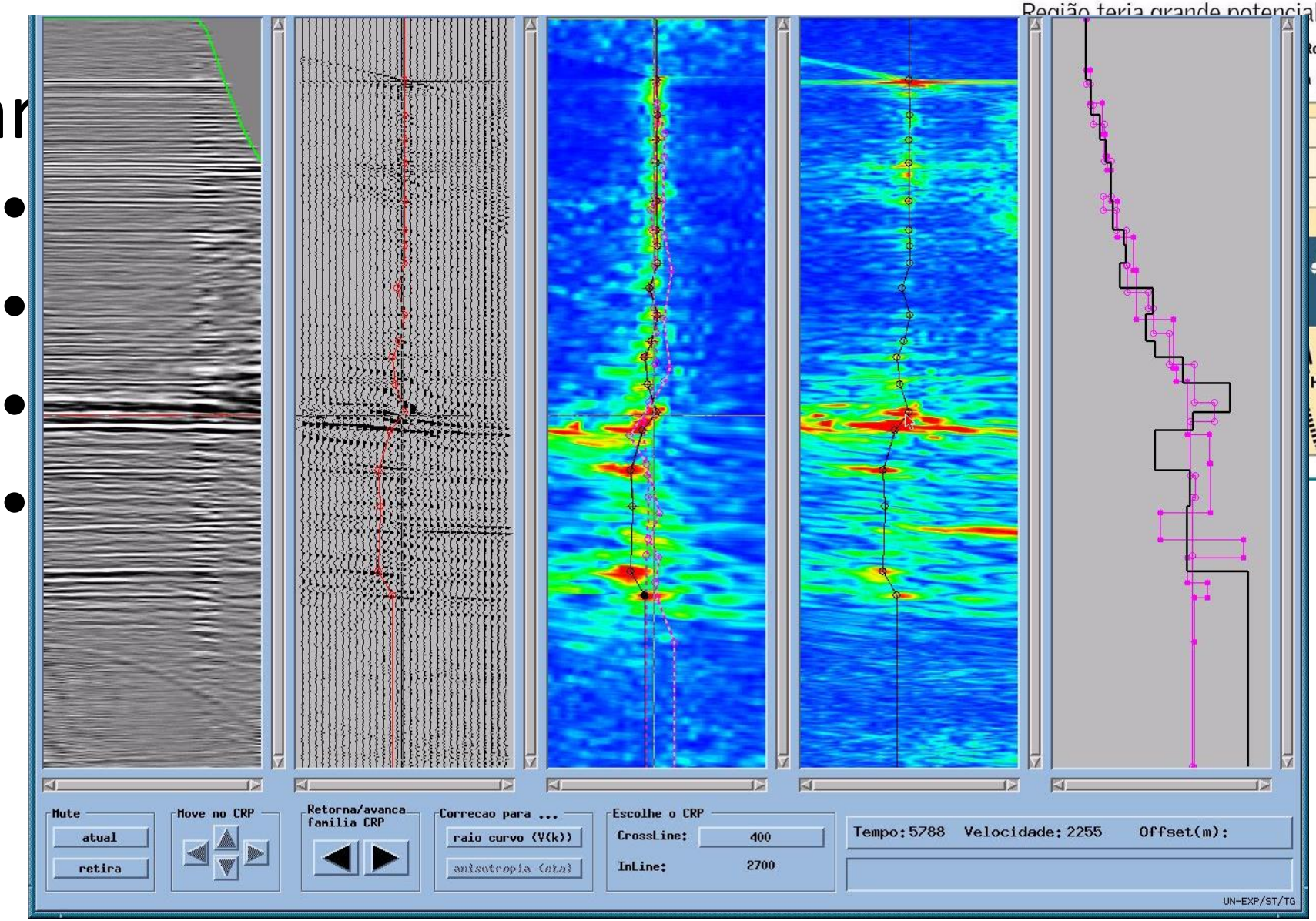
Not all computations fit into a big dataframe.

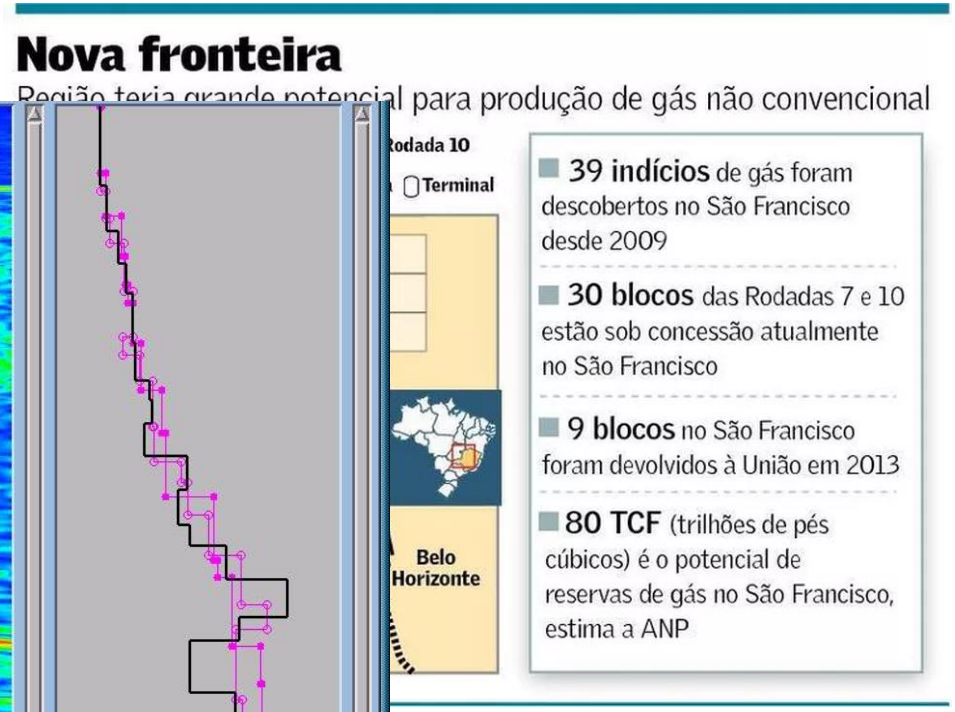
Dask exposes lower-level APIs letting you build custom systems for in-house applications. This helps open source leaders parallelize their own packages and helps business leaders scale custom business logic.

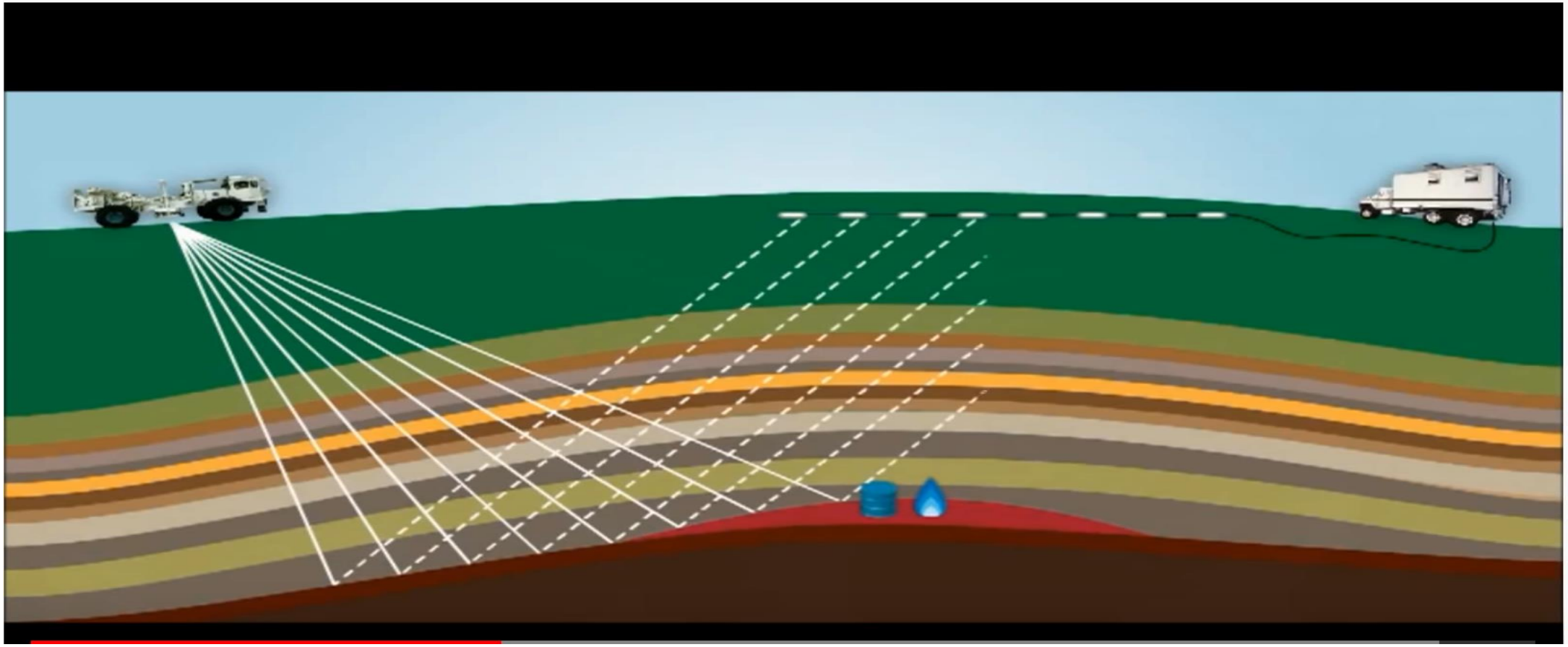
Learn More » | Try Now »

Car

-
-
-
-


Mute
atual
retira
Move no CRP
Retorna/avanca familia CRP
Correcao para ...
raio curvo (V(k))
anisotropia (eta)
Escolhe o CRP
CrossLine: 400
InLine: 2700
Tempo: 5788 Velocidade: 2255 Offset(m):
UN-EXP/ST/TG


Nova fronteira
Região teria grande potencial para produção de gás não convencional
Rodada 10
Terminal
Belo Horizonte
39 indícios de gás foram descobertos no São Francisco desde 2009
30 blocos das Rodadas 7 e 10 estão sob concessão atualmente no São Francisco
9 blocos no São Francisco foram devolvidos à União em 2013
80 TCF (trilhões de pés cúbicos) é o potencial de reservas de gás no São Francisco, estima a ANP



https://www.youtube.com/watch?v=wy68B476XVU

C 1 Client: SEAM Corp. Vendor: AGT Date: 14 Oct 2014
C 2 Project Description: SEAM Phase I Interpretation Challenge
C 3 SEGY Data Format: IBM Real XY Units: Meters
C 4 Data Description: 2793 Full Azimuth Shot Records with Free Surface
C 5 Data Type: RTM Migration with Perturbed Vp and Noise in depth Units: m
C 6 Sparse collection of 2D offset gathers in depth
C 7
C 8 Projection: UTM Zone 16 Spheroid: WGS84
C 9 Geodetic Datum: WGS84 Central Meridian: 87 degrees W
C10 Grid Origin XY: 401,674.66E 3,097,605.23N Scale Factor: .9996
C11 Grid Origin Lat Long: 28:00:00N 88:00:00W
C12
C13 Corner X Y Iline Xline Given in Local Coordinates K
C14 1 2940 24000 1589 5801
C15 2 31740 24000 7349 5801 Number of gathers: 33
C16
C17 Distance between gather X locations: 900 meters
C18
C19 Inline bin size: 5 m Max depth: 15000 m
C20 Crossline bin size: 5 m Sample interval: 20 m
C21 First Sample: 0 ms Samples/trace: 751
C22 Offsets: 0 - 11600 m Increment: 400 m Number of offsets:
C23
C24 Polarity: +RC peak Phase: Zero
C25 Datum: Sea Level
C26
C27 HEADER NAME POSITION LENGTH
C28 CDPX 181 4I
C29 CDPY 185 4I
C30 INLINE 189 4I
C31 CROSSLINE 193 4I
C32
C33 PROCESSING FLOW:
C34 Designature
C35 RTM with perturbed Vp
C36
C37
C38
C39 SEGY REV1
C40 END TEXTUAL HEADER

Seismic_Analysis_upload4.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text Connect Editing

Seismic Analysis with Python

1) Importing Libraries

```
# Importing Libraries
import numpy as np
from scipy import ndimage as ndi
from shutil import copyfile
from skimage import exposure
!pip install segyio
import segyio
import matplotlib
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.offsetbox import AnchoredText
```

```
Collecting segyio
  Downloading segyio-1.9.7-cp37-cp37m-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (83 kB)
     |████████████████████████████████| 83 kB 2.0 MB/s
Requirement already satisfied: numpy>=1.10 in /usr/local/lib/python3.7/dist-packages (from segyio) (1.19.5)
Installing collected packages: segyio
Successfully installed segyio-1.9.7
```

```
[ ] #Data no Google Drive
    #from google.colab import drive
    #drive.mount('/content/drive')
```

```
[ ] #import segyio ## JA FORAM IMPORTADOS NO INICIO
    #import numpy as np ## JA FORAM IMPORTADOS NO INICIO
    #Data uploaded
    with segyio.open('SEAM_Interpretation_Challenge_1_2DSparseGathers_Depth.sgy','r+') as f:
        for trace in f.trace:
            filtered = trace[np.where(trace < 1e-2)]
```

2) Loading Data

```
[ ] import sys
```

Seismic_Analysis_upload4.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text Connect Editing

2) Loading Data

```
[ ] import sys
    import os
```

```
[ ] # Set the default plot size for matplotlib figures
    matplotlib.rcParams['figure.figsize'] = (11.75,8.5)
```

```
[ ] base_segy = 'SEAM_Interpretation_Challenge_1_2DSparseGathers_Depth.sgy'
```

```
    f = segyio.open(base_segy,ignore_geometry=True)
    segyio.tools.wrap(f.text[0])
```

```
'C 1  Client: SEAM Corp.     Vendor: AGT                    Date: 14 Oct 2014\nC 2  Project Description: SEAM Phase I Interpretation Challenge\nC 3  SEGY Data Format: IBM Real                          XY Un
its: Meters\nC 4  Data Description: 2793 Full Azimuth Shot Records with Free Surface\nC 5  Data Type: RTM Migration with Perturbed Vp and Noise in depth   Units: m\nC 6             Sparse collection of 2D offs
et gathers in depth\nC 7\nC 8  Projection: UTM Zone 16               Spheroid: WGS84\nC 9  Geodetic Datum: WGS84                 Central Meridian: 87 degrees W\nC10  Grid Origin XY: 401,674.66E 3,097,605.23N
Scale Factor: .9996\nC11  Grid Origin Lat Long: 28:00:00N  88:00:00W\nC12\nC13  Corner     X        Y     Iline  Xline   Given in Local Coordinates     K\nC14    1       2940    24000   1589   5801\nC15    2
31740    24000   7349   5801    Number of gathers: 33\nC16\nC17  Distance between gather X locations:  900 meters\nC18\nC19  Inline bin size:      5 m        ...'
```

```
[ ] f.trace[10]
    f.trace[-2]
```

```
array([-2.01366043e+00, -6.05375385e+00, -3.78104305e+00, -8.63110006e-01,
        6.11716866e-01, -1.63687706e+00, -2.72039413e+00, -1.14763451e+00,
        3.35989475e+00,  4.76012993e+00,  3.01636696e+00,  1.83521557e+00,
        2.65846252e+00, -1.27267742e+00, -3.69801140e+00,  8.05239201e-01,
        4.53335667e+00, -1.65886497e+00, -6.33541203e+00,  8.70620906e-01,
        7.55232525e+00,  4.12557602e+00,  5.89431465e-01,  5.17095290e-02,
       -4.96521854e+00, -1.07247038e+01, -7.67143345e+00,  2.50645256e+00,
        1.10698195e+01,  1.00203018e+01,  9.17518139e-01, -8.90377998e+00,
       -9.49874973e+00,  1.43924522e+00,  8.84746647e+00,  2.81741047e+00,
       -8.42773628e+00, -6.76858616e+00,  3.82758808e+00,  1.22012606e+01,
        9.46120262e+00, -8.98574948e-01, -7.73301506e+00, -7.93793106e+00,
       -1.43096924e+00,  4.90565586e+00,  3.15968513e+00,  9.93162572e-01,
        1.53293324e+00,  2.92832184e+00,  2.60410786e+00, -7.50936151e-01,
       -4.47587109e+00, -6.36281204e+00, -7.13270426e-01,  4.24686813e+00,
       -6.18524432e-01, -7.36450863e+00, -2.28927326e+00,  6.71272373e+00,
        3.75258255e+00, -9.49347782e+00, -8.34703350e+00,  7.37300205e+00,
        1.78732300e+01,  1.31640291e+01, -5.32160282e-01, -8.42532444e+00,
       -1.10773792e+01, -6.01653004e+00,  1.80954838e+00,  7.43215275e+00,
        3.20165825e+00, -6.73680019e+00, -6.42342377e+00,  8.22889090e-01,
        6.02881241e+00,  2.97925282e+00, -7.87215412e-01, -5.12140691e-01,
        2.53087616e+00,  9.92177129e-02, -3.70834732e+00,  2.38673878e+00,
        2.21116352e+00,  2.37938023e+00,  2.30319118e+00, -1.29623127e+00
```

```
        -8.96327496e+00, -6.63597775e+00, -5.75602341e+00, -5.19660282e+00,
        -5.01639080e+00, -4.17887974e+00, -2.04370785e+00,  9.70532596e-01,
         4.06124210e+00.  5.93174458e+00.  5.84596157e+00.  4.41881561e+00.
```

```
f.samples[:5]
```

```
array([ 0., 20., 40., 60., 80.])
```

```
with segyio.open(base_segy) as segyf:
    n_traces = segyf.tracecount
    sample_rate = segyio.tools.dt(segyf)
    n_samples = segyf.samples.size
    n_il = len(segyf.iline)
```

```
f = segyio.open(base_segy, ignore_geometry = True)
ntraces    = len(f.trace)
inlines    = []
crosslines = []

for i in range(ntraces):
    headeri = f.header[i]
    inlines.append(headeri[segyio.su.iline])
    crosslines.append(headeri[segyio.su.xline])

print(f'{ntraces} traces')
print(f'first 10 inlines: {inlines[:10]}')
print(f'first 10 crosslines: {crosslines[:10]}')
```

```
990 traces
first 10 inlines: [1589, 1589, 1589, 1589, 1589, 1589, 1589, 1589, 1589, 1589]
first 10 crosslines: [5801, 5801, 5801, 5801, 5801, 5801, 5801, 5801, 5801, 5801]
```

```
# loading the seismic data as cube
seismic_data = segyio.tools.cube(base_segy)
```

```
# lets check the inline, xline spread of the seismic data
print('Survey IL/XL shape:' +str(np.shape(seismic_data)[0])+ ' / ' +str(np.shape(seismic_data)[1]))
```

```
Survey IL/XL shape:1 / 33
```

```
#Check the inline e cross line range for the seismic data directly from segy file is below
print('Inline Range:' +str(np.amin(segyf.ilines))+'-'+str(np.amax(segyf.ilines)))
print('Xnline Range:' +str(np.amin(segyf.xlines))+'-'+str(np.amax(segyf.xlines)))
```

```
[ ] print('Xnline Range:' +str(np.amin(segyf.xlines))+'-'+str(np.amax(segyf.xlines)))
```

```
Inline Range:1589-7349
Xnline Range:5801-5801
```

```
[ ] # Plot the inline and crossline as a scatter plot
    #plt.scatter(crosslines, inlines, marker="s", s=1)
```

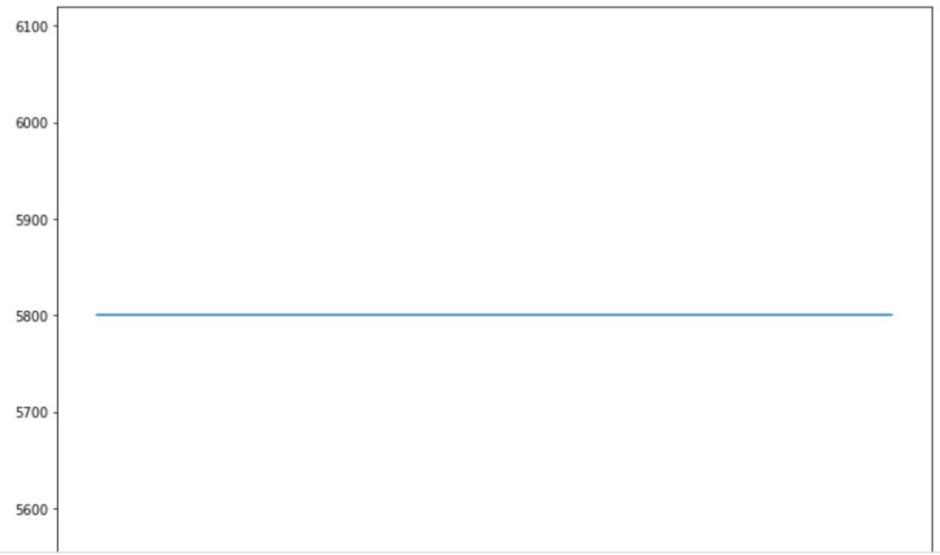
```
[ ] import itertools
    uniqil = set(inlines)
    uniqxl = set(crosslines)
    real = set(zip(inlines, crosslines))
    grid = set(itertools.product(uniqil, uniqxl))
    missing = grid - real
    missing
```

```
set()
```

```
[ ] #plt.plot(inlines)
```

```
[ ] plt.plot(crosslines[:5000])
```

```
[<matplotlib.lines.Line2D at 0x7f82f7df7210>]
```



Activities Firefox Web Browser dez 16 09:25

Seismic_Analysis_upload4.ipynb

https://colab.research.google.com/drive/1gHgFaYU8v63hCWXwcjFMyEIfZ2ZI_Rde

Seismic_Analysis_upload4.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text Connect Editing

```
[ ]  #import numpy as np
     ils = np.unique(inlines)
     xls = np.unique(crosslines)
     inline_interval    = ils[1:] - ils[:-1]
     crossline_interval = xls[1:] - xls[:-1]
     print(inline_interval)
     print(crossline_interval)
```

```
[180 180 180 180 180 180 180 180 180 180 180 180 180 180 180 180 180 180
 180 180 180 180 180 180 180 180 180 180 180 180 180 180 180]
[]
```

```
ils = sorted(uniqil)
xls = sorted(uniqxl)
lineindex = {
    (il, xl): i
    for i, (il, xl) in enumerate(sorted(grid))

}
lineindex
```

```
{(1589, 5801): 0,
 (1769, 5801): 1,
 (1949, 5801): 2,
 (2129, 5801): 3,
 (2309, 5801): 4,
 (2489, 5801): 5,
 (2669, 5801): 6,
 (2849, 5801): 7,
 (3029, 5801): 8,
 (3209, 5801): 9,
 (3389, 5801): 10,
 (3569, 5801): 11,
 (3749, 5801): 12,
 (3929, 5801): 13,
 (4109, 5801): 14,
 (4289, 5801): 15,
 (4469, 5801): 16,
 (4649, 5801): 17,
 (4829, 5801): 18,
 (5009, 5801): 19,
 (5189, 5801): 20,
 (5369, 5801): 21,
 (5549, 5801): 22,
 (5729, 5801): 23,
 (5909, 5801): 24,
 (6089, 5801): 25,
 (6269, 5801): 26,
 (6449, 5801): 27,
 (6629, 5801): 28
```

Activities Firefox Web Browser dez 16 09:25

Seismic_Analysis_upload4.ipynb

https://colab.research.google.com/drive/1gHgFaYU8v63hCWXwcjFMyEIfZ2ZI_Rde

Seismic_Analysis_upload4.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text Connect Editing

```
[ ] d = np.zeros((len(ils), len(xls), len(f.samples)))
    lineard = d.reshape(d.shape[0] * d.shape[1], d.shape[2])
    for il, xl, trace in zip(inlines, crosslines, f.trace[:]):
        lineard[lineindex[il, xl]][:] = trace[:]
```

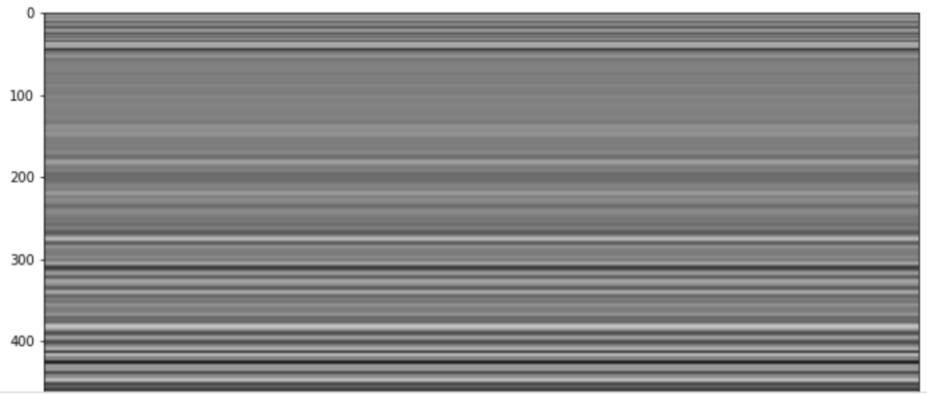
```
[ ] d.shape
```

```
(33, 1, 751)
```

```
[ ] # Set up some aliases
    ilines = np.array(sorted(uniqil))
    xlines = np.array(sorted(uniqxl))
    t = np.array(f.samples)

    # Estimate the amplitude range to use for the plots by taking the 95th percentile

    vm = np.percentile(d, 95)
    print(vm)

    # Define the central line
    central = len(ilines) // 2
```

```
19.057722473144516
```

```
[ ] # Plot

    plt.imshow(d[central,:,:].T, cmap='gray', aspect='auto', vmin=-vm, vmax=vm)
    plt.show()
```



Activities Firefox Web Browser dez 16 09:26

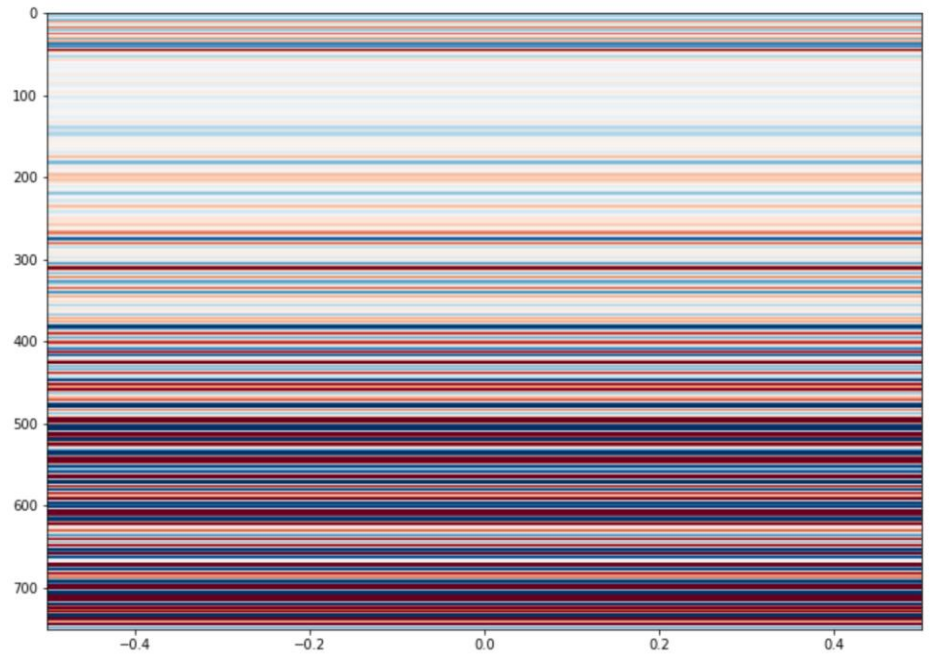
Seismic_Analysis_upload4.ipynb

File Edit View Insert Runtime Tools Help Saving...

Comment Share

+ Code + Text Connect Editing

```
#plt.imshow(d[central,:,:].T, cmap='RdBu',aspect='auto',vmin=-vm, vmax=vm)
plt.imshow(d[central,:,:].T, cmap='RdBu',aspect='auto', vmin=-10, vmax=10)
plt.show()
```




```
fig,ax = plt.subplots()

# Some example data
y = np.linspace(700.,900.,401)
offset = 94.
x = offset+10*(np.sin(y/2.)*
        1/(10. * np.sqrt(2 * np.pi)) *
        np.exp( - (y - 800)**2 / (2 * 10.**2))
        ) # This function just gives a wave that looks something like a seismic arrival

ax.plot(x,y,'k-')
ax.fill_betweenx(y,offset,x,where=(x>offset),color='k')

ax.set_xlim(93,95)

plt.show()
```

Activities Firefox Web Browser dez 16 09:26

Seismic_Analysis_upload4.ipynb

https://colab.research.google.com/drive/1gHgFaYU8v63hCWXwcjFMyEIfZ2ZI_Rde#scrollTo=mOwcMFsyIyjz

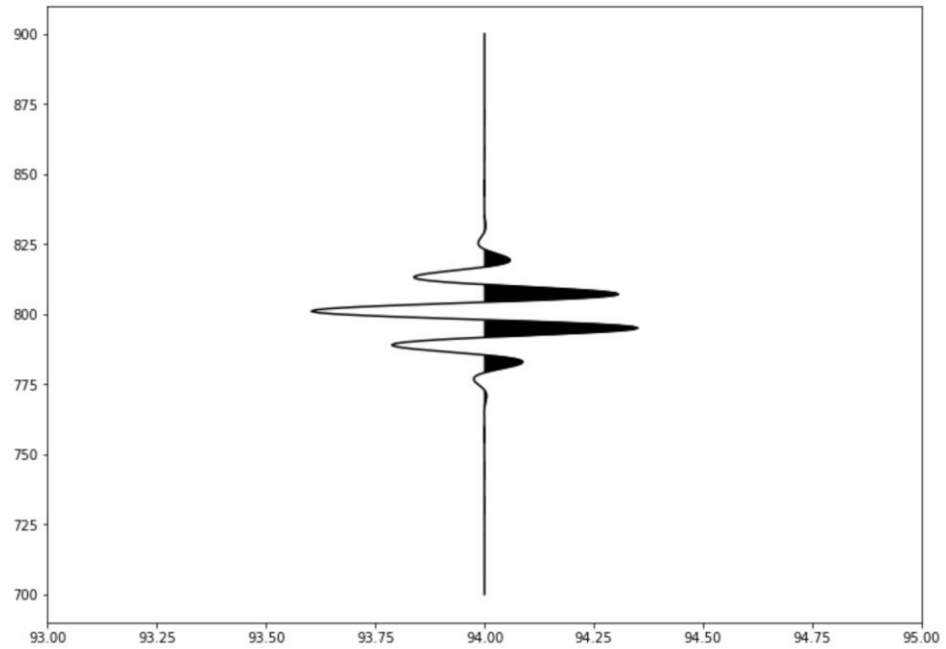
Seismic_Analysis_upload4.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text Connect Editing

```
plt.show()
```



[ ]

```
fig,ax = plt.subplots()

# Some example data
y = np.linspace(700.,900.,401)
offsets = [94., 95., 96., 97.]
times = [800., 790., 780., 770.]

for offset, time in zip(offsets,times):
    x = offset+10*(np.sin(y/2.)*
        1/(10. * np.sqrt(2 * np.pi)) *
        np.exp( - (y - time)**2 / (2 * 10.**2))
        )

    ax.plot(x,y,'k-')
    ax.fill_betweenx(y,offset,x,where=(x>offset),color='k')

ax.set_xlim(93,98)
```

Activities Firefox Web Browser dez 16 09:26

Seismic_Analysis_upload4.ipynb

https://colab.research.google.com/drive/1gHgFaYU8v63hCWXwcjFMyEIfZ2ZI_Rde#scrollTo=mOwcMFsyIyjz

Seismic_Analysis_upload4.ipynb

File Edit View Insert Runtime Tools Help All changes saved

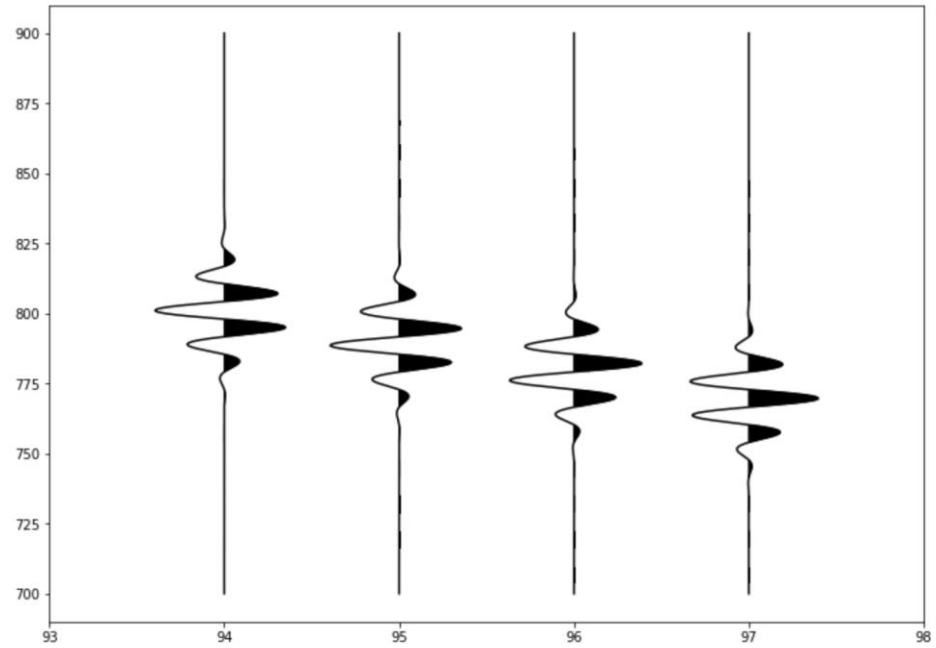
Comment Share

+ Code + Text

Connect Editing

```
    plt.show()
```

[ ]



[ ]

```
extent = [xlines[0], xlines[-1], t[-1], t[0]]

# Plot the central inline in the usual orientation by transposing the data

plt.imshow(d[central,:,:].T, cmap='gray', aspect='auto', vmin=-vm, vmax=vm, extent=extent)

plt.xlabel('Xline')
plt.ylabel('TWT (ms)')
plt.title('SEAM Phase I Interpretation Challenge'+ str(ilines[central]))
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:5: UserWarning: Attempting to set identical left == right == 5801 results in singular transformations; automatically expanding.
  """
```

```
import xarray as xr
!pip install hvplot
import panel as pn
```

```
Collecting hvplot
  Downloading hvplot-0.7.3-py2.py3-none-any.whl (3.1 MB)
     |████████████████████████████████| 3.1 MB 8.6 MB/s
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (from hvplot) (1.1.5)
Requirement already satisfied: holoviews>=1.11.0 in /usr/local/lib/python3.7/dist-packages (from hvplot) (1.14.6)
Requirement already satisfied: bokeh>=1.0.0 in /usr/local/lib/python3.7/dist-packages (from hvplot) (2.3.3)
Requirement already satisfied: numpy>=1.15 in /usr/local/lib/python3.7/dist-packages (from hvplot) (1.19.5)
Requirement already satisfied: colorcet>=2 in /usr/local/lib/python3.7/dist-packages (from hvplot) (2.0.6)
Requirement already satisfied: packaging>=16.8 in /usr/local/lib/python3.7/dist-packages (from bokeh>=1.0.0->hvplot) (21.3)
Requirement already satisfied: tornado>=5.1 in /usr/local/lib/python3.7/dist-packages (from bokeh>=1.0.0->hvplot) (5.1.1)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from bokeh>=1.0.0->hvplot) (2.8.2)
Requirement already satisfied: pillow>=7.1.0 in /usr/local/lib/python3.7/dist-packages (from bokeh>=1.0.0->hvplot) (7.1.2)
Requirement already satisfied: typing-extensions>=3.7.4 in /usr/local/lib/python3.7/dist-packages (from bokeh>=1.0.0->hvplot) (3.10.0.2)
Requirement already satisfied: Jinja2>=2.9 in /usr/local/lib/python3.7/dist-packages (from bokeh>=1.0.0->hvplot) (2.11.3)
Requirement already satisfied: PyYAML>=3.10 in /usr/local/lib/python3.7/dist-packages (from bokeh>=1.0.0->hvplot) (3.13)
Requirement already satisfied: param>=1.7.0 in /usr/local/lib/python3.7/dist-packages (from colorcet>=2->hvplot) (1.12.0)
Requirement already satisfied: pyct>=0.4.4 in /usr/local/lib/python3.7/dist-packages (from colorcet>=2->hvplot) (0.4.8)
Requirement already satisfied: panel>=0.8.0 in /usr/local/lib/python3.7/dist-packages (from holoviews>=1.11.0->hvplot) (0.12.1)
Requirement already satisfied: pyviz-comms>=0.7.4 in /usr/local/lib/python3.7/dist-packages (from holoviews>=1.11.0->hvplot) (2.1.0)
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.7/dist-packages (from Jinja2>=2.9->bokeh>=1.0.0->hvplot) (2.0.1)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.7/dist-packages (from packaging>=16.8->bokeh>=1.0.0->hvplot) (3.0.6)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/dist-packages (from pandas->hvplot) (2018.9)
Requirement already satisfied: tqdm>=4.48.0 in /usr/local/lib/python3.7/dist-packages (from panel>=0.8.0->holoviews>=1.11.0->hvplot) (4.62.3)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from panel>=0.8.0->holoviews>=1.11.0->hvplot) (2.23.0)
Requirement already satisfied: bleach in /usr/local/lib/python3.7/dist-packages (from panel>=0.8.0->holoviews>=1.11.0->hvplot) (4.1.0)
Requirement already satisfied: markdown in /usr/local/lib/python3.7/dist-packages (from panel>=0.8.0->holoviews>=1.11.0->hvplot) (3.3.6)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.1->bokeh>=1.0.0->hvplot) (1.15.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.7/dist-packages (from bleach->panel>=0.8.0->holoviews>=1.11.0->hvplot) (0.5.1)
Requirement already satisfied: importlib-metadata>=4.4 in /usr/local/lib/python3.7/dist-packages (from markdown->panel>=0.8.0->holoviews>=1.11.0->hvplot) (4.8.2)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from importlib-metadata>=4.4->markdown->panel>=0.8.0->holoviews>=1.11.0->hvplot) (3.6.0)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests->panel>=0.8.0->holoviews>=1.11.0->hvplot) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests->panel>=0.8.0->holoviews>=1.11.0->hvplot) (2.10)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests->panel>=0.8.0->holoviews>=1.11.0->hvplot) (2021.10.8)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests->panel>=0.8.0->holoviews>=1.11.0->hvplot) (1.24.3)
Installing collected packages: hvplot
Successfully installed hvplot-0.7.3
```

```
[ ] # Define a plotting function, the input will be the inline

    def plot_inl(inl):
        """
        Plot a single inline using hvplot
        """
```

Activities Firefox Web Browser dez 16 09:26

Seismic_Analysis_upload4.ipynb

https://colab.research.google.com/drive/1gHgFaYU8v63hCWXwcjFMyEIfZ2ZI_Rde#scrollTo=mOwcMFsyiyjz

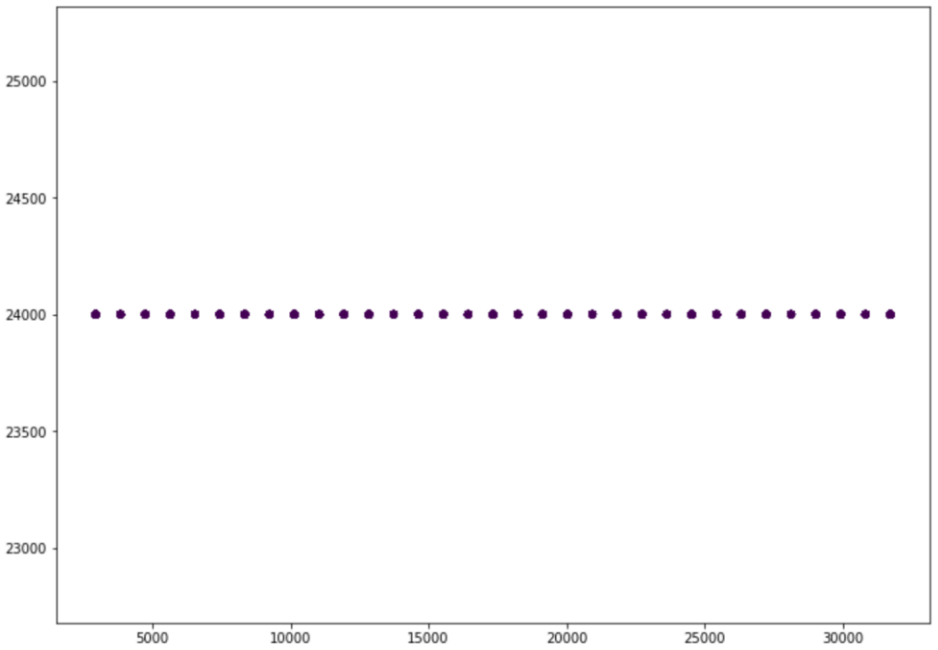
Seismic_Analysis_upload4.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text Connect Editing

```
filename = 'SEAM_Interpretation_Challenge_1_2DSparseGathers_Depth.sgy'
with segyio.open(filename, ignore_geometry=True) as segyfile:
    segyfile.mmap()

    # Extract header word for all traces
    sourceX = segyfile.attributes(segyio.TraceField.SourceX)[:]

    # Scatter plot sources and receivers color-coded on their number
    plt.figure()
    sourceY = segyfile.attributes(segyio.TraceField.SourceY)[:]
    nsum = segyfile.attributes(segyio.TraceField.NSummedTraces)[:]
    plt.scatter(sourceX, sourceY, c=nsum, edgecolor='none')

    groupX = segyfile.attributes(segyio.TraceField.GroupX)[:]
    groupY = segyfile.attributes(segyio.TraceField.GroupY)[:]
    nstack = segyfile.attributes(segyio.TraceField.NStackedTraces)[:]
    plt.scatter(groupX, groupY, c=nstack, edgecolor='none')
```



2) Data Preprocessing

```
[ ] def main():
        filename = 'SEAM_Interpretation_Challenge_1_2DSparseGathers_Depth.sgy'

        # Open file
        with segyio.open(filename, ignore_geometry=True) as f:
        # Get all header keys:
            header_keys = segyio.tracefield.keys
            # Initialize df with trace id as index and headers as columns
            trace_headers = pd.DataFrame(index=range(1, f.tracecount+1),
                                columns=header_keys.keys())
            # Fill dataframe with all trace headers values
            for k, v in header_keys.items():
                trace_headers[k] = f.attributes(v)[:]
        print(trace_headers.head())
        trace_headers.to_csv('out.csv')
    if __name__ == '__main__':
        main()
```

```
   TRACE_SEQUENCE_LINE  TRACE_SEQUENCE_FILE  ...  UnassignedInt1  UnassignedInt2
1                    0                    0  ...               0               0
2                    0                    0  ...               0               0
3                    0                    0  ...               0               0
4                    0                    0  ...               0               0
5                    0                    0  ...               0               0

[5 rows x 91 columns]
```

```
[ ] pd.read_csv('out.csv')
```

	Unnamed: 0	TRACE_SEQUENCE_LINE	TRACE_SEQUENCE_FILE	FieldRecord	TraceNumber	EnergySourcePoint	CDP	CDP_TRACE	TraceIdentificationCode	NSummedTraces	NStackedTraces	DataUse	offset	ReceiverGroupElevatio
0	1	0	0	0	0	1589	5801	0	1	0	0	0	0	
1	2	0	0	0	0	1589	5801	0	1	0	0	0	400	
2	3	0	0	0	0	1589	5801	0	1	0	0	0	800	
3	4	0	0	0	0	1589	5801	0	1	0	0	0	1200	
4	5	0	0	0	0	1589	5801	0	1	0	0	0	1600	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
985	986	0	0	0	0	7349	5801	0	1	0	0	0	10000	

rapids-colab-0.8.ipynb

File Edit View Insert Runtime Tools Help

Share

+ Code + Text Copy to Drive

Connect Editing

RAPIDS Examples

Now you can run code!

What follows are basic examples where all processing takes place on the GPU.

cuDF

Load a dataset into a GPU memory resident DataFrame and perform a basic calculation.

Everything from CSV parsing to calculating tip percentage and computing a grouped average is done on the GPU.

```
[ ] import cudf
    import io, requests

    # download CSV file from GitHub
    url="https://github.com/plotly/datasets/raw/master/tips.csv"
    content = requests.get(url).content.decode('utf-8')

    # read CSV from memory
    tips_df = cudf.read_csv(io.StringIO(content))
    tips_df['tip_percentage'] = tips_df['tip']/tips_df['total_bill']*100

    # display average tip by dining party size
    print(tips_df.groupby('size').tip_percentage.mean())
```

```
size
1     21.72920154872781
2    16.571919173482886
3    15.215685473711835
4    14.594900639351332
5    14.149548965142023
6    15.622920072028379
Name: tip_percentage, dtype: float64
```

cuML

This snippet does label and one-hot encoding of the tips dataset's categorical features and applies standard scaling to all columns. All operations run on the GPU.

Conclusão

1. Desenvolvemos um *notebook* para dado sísmico utilizando como base o *notebook* da empresa petrolífera Equinor;
2. Adaptamos outras rotinas para o nosso problema;
3. Rotina para converter dados .sgy em .csv. Permitindo realizar novas implementações com o ferramental apresentado em sala;
4. Metodologia de QC do dado sísmico através do acesso a várias chaves do *header*;
5. Adquirirmos o conhecimento de novas formas de utilizar e manipular o dado sísmico em open *source*, ou seja, não utilizamos *software* de geofísica;
6. Temos como passos futuros a implementação das bibliotecas: Bokeh, Dask e Rapids;
7. Este notebook será ferramental para o desenvolvimento do doutorado.

Conclusão

Muito Obrigado pela oportunidade!
Saio do curso um profissional mais completo