UNIVERSIDAD
COMPLUTENSE
MADRID

Proyecto de Innovación

Convocatoria 2017/2017

Nº de proyecto 211

Diseño, desarrollo y puesta en marcha de un "Laboratorio Virtual de Geomatemática en lenguaje Python"

Rafael Lahoz-Beltrá, Pilar López González-Nieto, Mariángeles Gómez Flechoso, María Eugenia Arribas Mocoroa, Alfonso Muñoz Martín, María de la Luz García Lorenzo, Gloria Cabrera Gómez, Jose Antonio Alvarez Gómez, Andrea Caso Fraile, Jefferson Mark Orosco Dagan, Raul Merinero Palomar

Facultad de Ciencias Biológicas – Facultad de Ciencias Geológicas

1. Introducción

En la década de los 50 la Geología se transforma en una disciplina cuantitativa, y desde entonces las Matemáticas y Estadística son materias imprescindibles para los geólogos. La incorporación en Geología de las técnicas de análisis químico y físico en rocas y minerales, la experimentación en laboratorio y la exploración y explotación del subsuelo, condujeron junto con la popularización del ordenador, a un proceso de cuantificación de los conceptos. La Geología dejaba así de ser una disciplina cualitativa, tal y como había ocurrido desde sus orígenes en el siglo XVIII con los trabajos de John Hutton. El resultado de esta evolución es que a día de hoy los procedimientos matemáticos y estadísticos se aplican ampliamente en Geología, ya sea en petrología y sedimentología, paleomagnetismo, topografía, en el estudio de las formaciones geológicas o de la evolución de la erosión en un terreno, la exploración de recursos naturales (petróleo y gas), el estudio de la distribución de guijarros, en la medición de las presiones litostáticas e hisdrostáticas, el cálculo de la probabilidad de un terremoto la probabilidad del impacto de un asteroide en nuestro planeta.

2. Objetivos

El objetivo general del proyecto ha sido la creación de un “Laboratorio Virtual de Geomatemática en Python”, al que originariamente denominamos con el acrónimo LVGPy. Sin embargo, este nombre ha sido sustituido por ***GeoPy***, en abreviatura de “Geología en Python”. Se trata de una colección de *scripts*, es decir de programas ejecutados en un intérprete de lenguaje Python, con los que los estudiantes de Grado en Geología aprenden a aplicar modelos y métodos de Geomatemática (Matemáticas y Estadística) a situaciones experimentales y fenómenos geológicos reales. La finalidad del laboratorio virtual es contribuir a remediar en la medida de lo posible la actual escasez de obras sobre estas técnicas y su aplicación (Ferguson, 1988; Waltham, 200).

Los objetivos específicos de este Proyecto de Innovación fueron mejorar tanto (a) la formación de los egresados de la Facultad de Ciencias Geológicas de la UCM como (b) su empleabilidad.

Con el fin de lograr estos objetivos, el proyecto considera que los estudiantes deben mejorar su formación, lográndolo a partir de los siguientes objetivos docentes o hitos:

a) A partir de una colección de situaciones experimentales y fenómenos geológicos realistas los estudiantes ponen en práctica los conocimientos adquiridos en las asignaturas de Matemáticas (Matemáticas I en el Grado de Geología de la UCM) y Estadística (Matemáticas II en el Grado de Geología de la UCM).

b) Desarrollo de habilidades orientadas a la aplicación de los modelos matemáticos (por ejemplo, test estadísticos) y de la utilización del cálculo simbólico (por ejemplo, cálculo integral) en experimentos de campo o laboratorio.

c) Adquirir un grado de madurez que les permita participar en actividades de carácter multidisciplinar. Los estudiantes adquieren destrezas de razonamiento en un entorno de

trabajo en el que la Matemática es aplicada a la Geología, y la Geología condiciona el razonamiento matemático. Es decir, el alumno debe ser capaz de manejar tanto las operaciones básicas, como los símbolos y las expresiones formales, que conducen tanto a un razonamiento deductivo (por ejemplo, en Matemáticas) como inductivo (por ejemplo, en Estadística). En resumen, el estudiante aprende a razonar manejando un discurso en el que ambas disciplinas se confunden entre sí, preparándolo para el trabajo en un entorno multidisciplinar.

d) Introducir el uso del ordenador como elemento de integración de las Matemáticas y Estadística en la Geología.

e) Uno los hitos que otorga valor añadido a este Proyecto de Innovación es que los alumnos adquieren conocimientos de programación en lenguaje Python, uno de los lenguajes que por sus características está imponiéndose en el mundo científico, y por tanto en Geología (Wei-Bing, 2012). Puesto que Python es gratuito los alumnos pueden disponer de todo el material docente de ***GeoPy*** y usarlo libremente en su ordenador una vez se haya instalado cualquiera de las distribuciones orientadas al ámbito científico (por ejemplo, Anaconda, WinPython etc.). De esta forma se accede a coste cero a un *software* cuyas prestaciones son similares a MATLAB, y que permite por ejemplo realizar análisis estadísticos con una fiabilidad igual o incluso superior a algunos paquetes estadísticos comerciales.

El proyecto ***GeoPy***, tiene sus orígenes en una ponencia presentada en el "IV Congreso de Docentes de Ciencias" celebrado en abril de 2016 (González-Nieto y Lahoz-Beltra, 2017). En dicha ponencia se expusieron algunos de los *scripts* en lenguaje Python con los que abordar las siguientes preguntas ¿Cómo calcular el volumen del monte Fuji? ¿Cómo mediríamos el radio de la Tierra 255 a. C? En una ciudad como Madrid ¿cómo han variado los niveles de ozono máximo por mes? entre otros muchos ejemplos que fueron desarrollados posteriormente en este proyecto.

Los *scripts* tratan sobre (a) métodos matemáticos, tal es el caso de las aplicaciones del cálculo integral, y (b) análisis de datos estadísticos. En este último apartado se han incluido *scripts* de estadística descriptiva (univariante y bivariante) y de diseño experimental con una y dos poblaciones (inferencia estadística paramétrica y no paramétrica). Además, ha sido incluido un *script* sobre el análisis de datos enumerativos (frecuencias y proporciones) así como una introducción a los modelos lineales elementales, esto es el análisis de la regresión lineal simple y el modelo de ANOVA. Estas técnicas son aplicadas a situaciones reales familiarizando al estudiante con los (i) conceptos geomatemáticos, (ii) su aplicación, y (iii) el manejo del ordenador bajo lenguaje Python. De esta forma el ordenador se convierte en una herramienta más del geólogo junto con la "brújula y martillo".

3. Metodología

El proyecto fue realizado en varias etapas, que pueden resumirse tal y como se describe a continuación:

3.1. Creación de una colección de casos. Se plantean casos de estudio acerca de situaciones experimentales y fenómenos geológicos que requieran tratamiento matemático o estadístico. Se organizan en dos grupos según se utilicen métodos matemáticos o análisis de datos estadísticos. El resultado es una colección de "casos de estudio".

3.2. Aplicación de las Matemáticas y Estadística a la colección de casos de estudio. Los casos de estudio son resueltos "a mano" (por ejemplo, cálculo de integrales) o la solución es planteada sin resolver (por ejemplo, análisis estadístico descriptivo). Se incluye información en la que se explica cómo ha de resolverse un caso, conceptos teóricos, el razonamiento matemático que conduce a la solución, expresiones, supuestos, lectura e interpretación del *output*, y cualquier otra información relevante para su resolución. Los casos que requieren análisis estadístico son explicados por medio de videoclips en formato mp4 que pueden ser descargados de la página web de ***GeoPy***. El resultado es la solución al problema planteado en los "casos de estudio", y su explicación.

3.3. Creación, depuración y puesta a punto de ***GeoPy***. La solución o algoritmo al problema matemático/estadístico planteado en los "casos de estudio" fue programado como un *script* en Python versión 3.5.1. El conjunto de *scripts* junto con otros elementos es organizado como páginas web. El resultado es ***GeoPy***, el Laboratorio Virtual: las páginas web reúnen a los *scripts* con documentos electrónicos (en PDF descargables) con texto y fórmulas matemáticas de los modelos matemáticos, nociones de programación en Python, gráficos, etc. También se incluyen videos en formato mp4 con explicaciones sobre los *outputs* estadísticos y enlaces a otras páginas web sobre el tema geológico que es objeto de estudio y experimentación. El resultado es la publicación de una primera versión del Laboratorio Virtual ***GeoPy***.

3.4. Medición de indicadores clave de desempeño. Una vez disponible esta primera versión del Laboratorio Virtual ***GeoPy***, tenemos previsto realizar talleres piloto durante el curso 2017-2018 con los estudiantes dentro de los días dedicados a impartir seminarios/prácticas. Seguidamente se procederá a la medición del impacto del Proyecto de Innovación en relación con los objetivos docentes o hitos. El resultado es la evaluación del impacto del Proyecto de Innovación en los objetivos docentes.

4. Resultados

La versión 1.0 de ***GeoPy*** se encuentra hospedada en:

http://geopy.ga

dando acceso la página principal a las siguientes opciones (Figura 1):

Métodos Estadísticos

- Estadística descriptiva univariante.

- Estadística descriptiva bivariante.
- Estadística inferencial.
- Inferencia estadística en una población.
- Inferencia estadística en dos poblaciones.
- Inferencia estadística con proporciones y frecuencias.
- Modelos generales lineales: ANOVA.
- Modelos generales lineales: regresión lineal simple.

Modelos Geomatemáticos

- Ley de Gutenberg-Richter.
- Análisis de un perfil topográfico.
- Cálculo de la energía liberada por un terremoto.
- Cálculo del volumen de una elevación del terreno (colina, montaña, etc.).

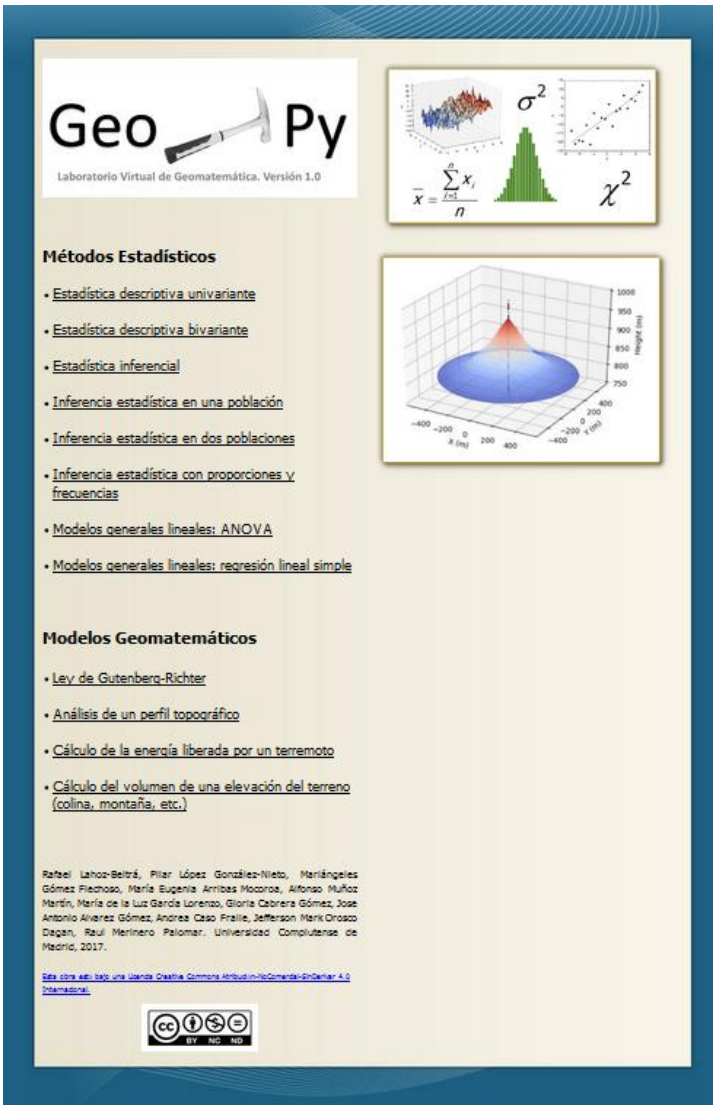

Figura 1.- Página principal de ***GeoPy***.

Una vez elegida una opción, la página web incluye una explicación general de las técnicas estadísticas o de cálculo (Figura 2). Además, las páginas incluyen elementos multimedia (Figura 3) y enlaces a otras páginas web.

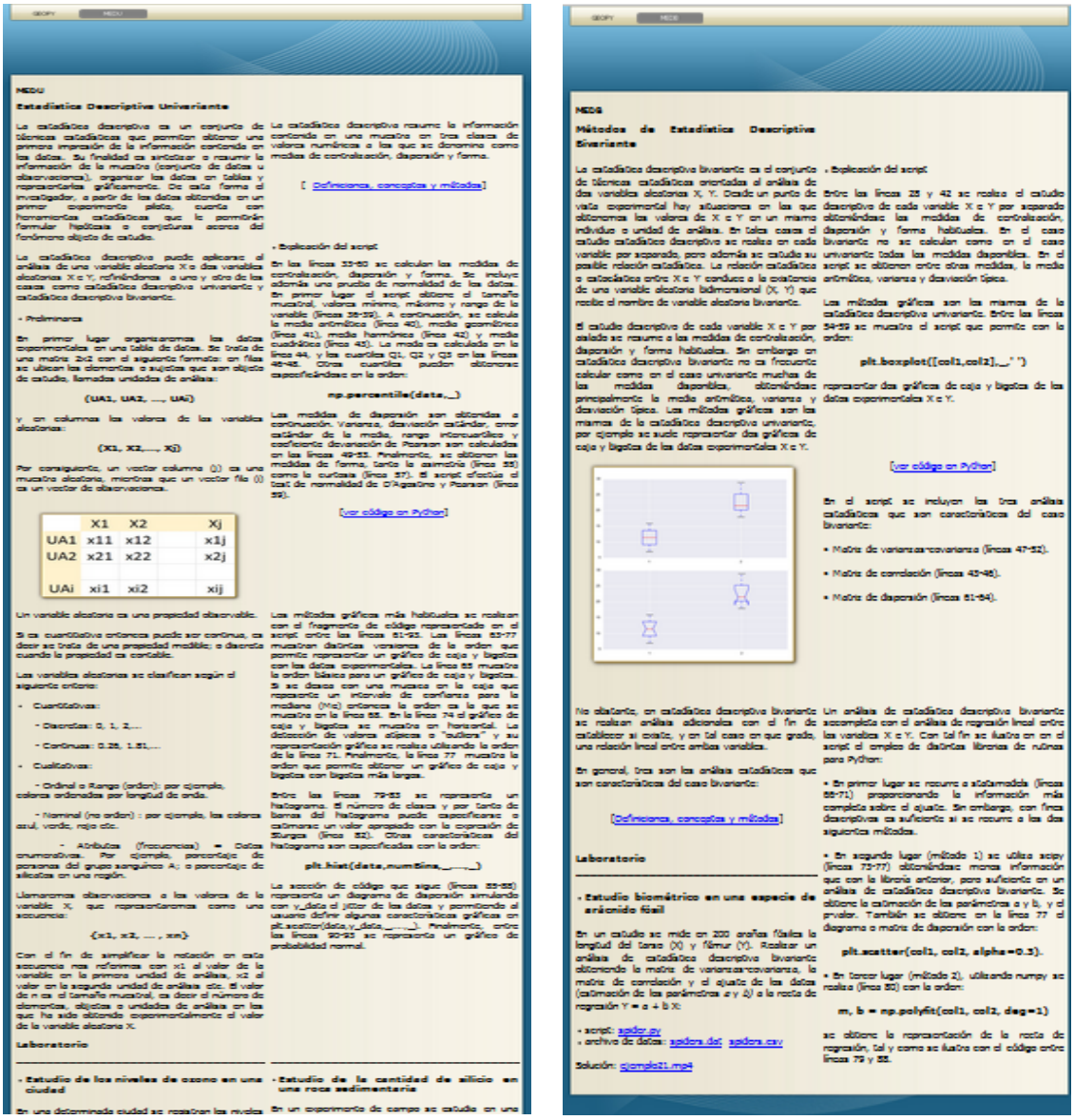

Figura 2.- Aspecto general de las páginas web.

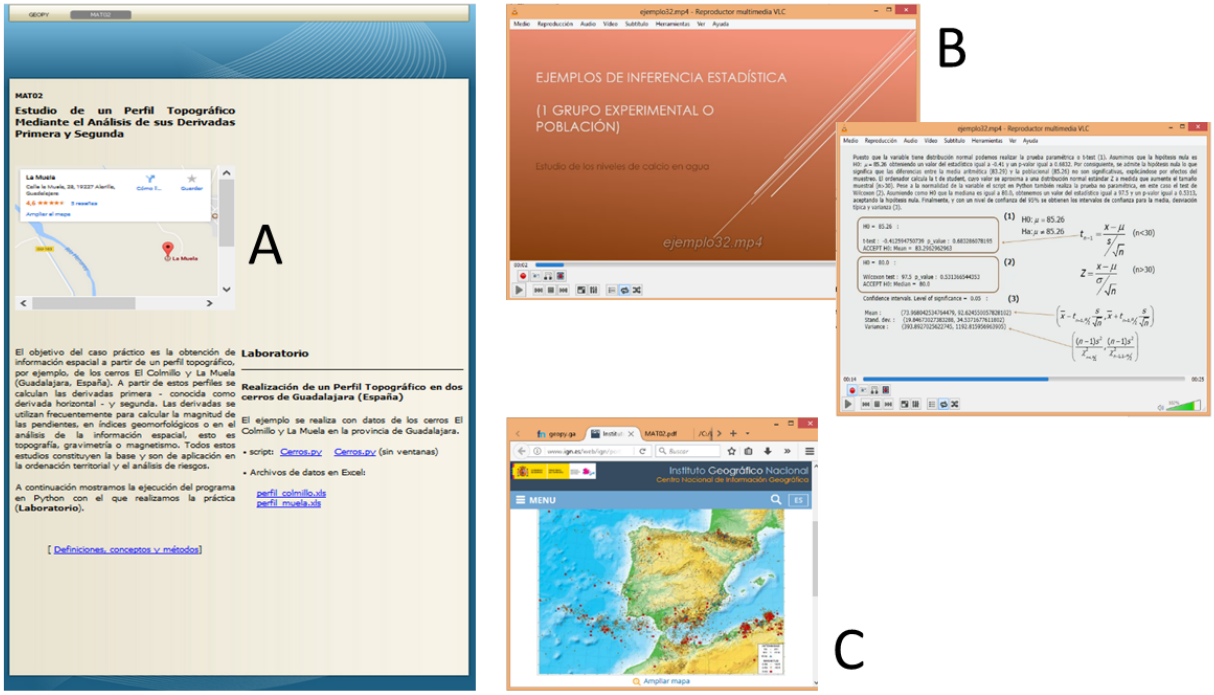

Figura 3. Elementos multimedia en una página web de ***GeoPy***: mapas de Google incrustados (**A**), videoclips con explicaciones detalladas e interpretación de los *outputs* (**B**). Algunas páginas incluyen enlaces a otras páginas web de instituciones u organizaciones que son relevantes en relación con el caso geológico que es objeto de estudio (**C**).

5. Referencias

Ferguson. J. 1988. *Mathematics in Geology*. Springer Netherlands.

Lopez González-Nieto, P., Lahoz-Beltra. R. 2017. Rocas, minerales y matemáticas o como aplicar la matemática en el estudio de la Tierra. En: *Actas del IV Congreso de Docentes de Ciencias de la Naturaleza* (M. Gónzalez Montero de Espinosa y A. Baratas Díaz, Eds.), Santillana Educación: 249-256.

Waltham, D. 2000. *Mathematics: A simple tool for geologists*. Wiley-Blackwell.

Wei-Bing, J. 2012. Why Python is the next wave in earth sciences computing. American Meteorological Society: 1823-1824.

6. Anexos

Los problemas planteados en el Laboratori o Virtual y los scripts en lenguaje Python fueron los que se muestran a continuación:

- **Métodos Estadísticos**
- **Estadística Descriptiva Univariante**

Estudio de los niveles de ozono en una ciudad

En una determinada ciudad se registran los niveles de ozono máximos durante 60 días. Sea X la variable aleatoria "nivel de ozono máximo por día", efectúese el correspondiente análisis estadístico descriptivo de la variable objeto de estudio ¿Qué podemos concluir? ¿hay días con valores atípicos o extremos de ozono?

- script: Ozone.py
- archivo de datos: Ozone.dat

Solución: ejemplo11.mp4

```
#############################################################
#                                                           #
#       Virtual Laboratory of Geomathematics in Python      #
#                                                           #
#       Univariate descriptive statistics  (01.06.2017)     #
#                                                           #
#    VLG Team, Complutense University of Madrid, Spain      #
#                                                           #
#   THIS SCRIPT IS PROVIDED BY THE AUTHORS "AS IS" AND      #
#   CAN BE USED BY ANYONE FOR THE PURPOSES OF EDUCATION     #
#   AND RESEARCH.                                           #
#                                                           #
#############################################################
import math
import numpy as np # importing numpy
import scipy.stats as s # importing scipy.stats
import statistics as ss # importing statistics

# Embedded graphics
import matplotlib.pyplot as plt # importing matplotlib
import seaborn as sns # importing seaborn

import pylab

# Aesthetic parameters of seaborn
sns.set_palette("deep", desat=.6)
sns.set_context(rc={"figure.figsize": (8, 4)})

# Example measurements of ozone in the atmosphere
data=np.loadtxt('Ozone.dat', skiprows=0)
print(data)

# Measures of centralization, dispersion and form
def quadratic_mean(num):
       return math.sqrt(sum(n*n for n in num)/len(num))
print('n = ',len(data))
print('Minimum = ',min(data))
print('Maximum = ',max(data))
print('Rank = ',max(data)-min(data))
print('Average = ',ss.mean(data))
print('Geometric mean= ',s.gmean(data))
print('Harmonic mean= ','s.hmean(data)')
print('Quadratic mean=',quadratic_mean(data))
```

```
print('Mode = ',s.mode(data))
print('Median = ',np.median(data))
print('Q1 = ',np.percentile(data,25))
print('Q2 = ',np.percentile(data,50))
print('Q3 = ',np.percentile(data,75))
print('Variance  =',ss.variance(data))
print('Standard deviation = ',ss.stdev(data))
print('Standard error of the mean  =',ss.stdev(data)/math.sqrt(len(data)))
print('Interquatile rank = ',np.percentile(data,75)-np.percentile(data,25))
print('Coefficient of variation = ',(ss.stdev(data)/ss.mean(data))*100)
print()
print('Skewness =',s.skewtest(data))
print()
print('Kurtosis = ',s.kurtosistest(data))
print()
print('Normality test (D'Agostino and Pearson's ) = ',s.normaltest(data))
print()
# Descriptive statistics

# Box-and-Whisker plot
# basic plot
plt.boxplot(data,0,' ')
# notched plot
plt.figure()
plt.boxplot(data, 1,' ')
# change outliers symbols
plt.figure()
plt.boxplot(data, 1, 'gD')
# horizontal plot
plt.figure()
plt.boxplot(data, 0, 'rs', 0)
# whisker length change
plt.figure()
plt.boxplot(data, 0, 'rs', 0, 0.75)

# Histogram
# histtype= normed=0 1 'bar' 'step', cumulative=0 1
plt.figure()
numBins = round(1+3.222*math.log10(len(data)))
plt.hist(data,numBins,normed=0,color='green',alpha=0.8, histtype='bar', cumulative=0)

# Scatter plot
y_data=[np.random.random() for x in range (0, len(data))]
plt.figure()
plt.scatter(data, y_data, color="red", marker="^")

# Normality probability plot
plt.figure()
s.probplot(data, dist="norm", plot=pylab)
pylab.show()
```

Estudio de la cantidad de silicio en una roca sedimentaria

En un experimento de campo se estudia en una roca sedimentaria la variable aleatoria X "cantidad de silicio por roca". Si realizamos un análisis estadístico descriptivo de la variable ¿Qué anomalía hemos detectado en este estudio? ¿cómo se distribuye el silicio en esta roca sedimentaria?

- script: Silicon.py
- archivo de datos: Silicon.dat

Solución: ejemplo12.mp4

```
#############################################################
#                                                           #
#       Virtual Laboratory of Geomathematics in Python      #
#                                                           #
#       Univariate descriptive statistics  (01.06.2017)     #
#                                                           #
#    VLG Team, Complutense University of Madrid, Spain      #
#                                                           #
#   THIS SCRIPT IS PROVIDED BY THE AUTHORS "AS IS" AND      #
#   CAN BE USED BY ANYONE FOR THE PURPOSES OF EDUCATION     #
#   AND RESEARCH.                                           #
#                                                           #
#############################################################
import math
import numpy as np # importing numpy
import scipy.stats as s # importing scipy.stats
import statistics as ss # importing statistics

# Embedded graphics
import matplotlib.pyplot as plt # importing matplotlib
import seaborn as sns # importing seaborn

import pylab

# Aesthetic parameters of seaborn
sns.set_palette("deep", desat=.6)
sns.set_context(rc={"figure.figsize": (8, 4)})

# Example amount of silicon in sedimentary rocks
data=np.loadtxt('Silicon.dat', skiprows=0)
print(data)

# Measures of centralization, dispersion and form
def quadratic_mean(num):
        return math.sqrt(sum(n*n for n in num)/len(num))
print('n = ',len(data))
print('Minimum = ',min(data))
print('Maximum = ',max(data))
print('Rank = ',max(data)-min(data))
print('Average = ',ss.mean(data))
print('Geometric mean= ',s.gmean(data))
print('Harmonic mean= ','s.hmean(data)')
print('Quadratic mean=',quadratic_mean(data))
print('Mode = ',s.mode(data))
print('Median = ',np.median(data))
print('Q1 = ',np.percentile(data,25))
print('Q2 = ',np.percentile(data,50))
print('Q3 = ',np.percentile(data,75))
print('Variance  =',ss.variance(data))
print('Standard deviation = ',ss.stdev(data))
print('Standard error of the mean  =',ss.stdev(data)/math.sqrt(len(data)))
print('Interquatile rank = ',np.percentile(data,75)-np.percentile(data,25))
print('Coefficient of variation = ',(ss.stdev(data)/ss.mean(data))*100)
print()
print('Skewness =',s.skewtest(data))
print()
print('Kurtosis = ',s.kurtosistest(data))
print()
print('Normality test (D'Agostino and Pearson's ) = ',s.normaltest(data))
print()
# Descriptive statistics

# Box-and-Whisker plot
# basic plot
plt.boxplot(data,0,' ')
# notched plot
plt.figure()
plt.boxplot(data, 1,' ')
# change outliers symbols
plt.figure()
plt.boxplot(data, 1, 'gD')
# horizontal plot
plt.figure()
plt.boxplot(data, 0, 'rs', 0)
# whisker length change
plt.figure()
plt.boxplot(data, 0, 'rs', 0, 0.75)
```

```
# Histogram
# histtype= normed=0 1 'bar' 'step', cumulative=0 1
plt.figure()
numBins = round(1+3.222*math.log10(len(data)))
plt.hist(data,numBins,normed=0,color='green',alpha=0.8, histtype='bar', cumulative=0)

# Scatter plot
y_data=[np.random.random() for x in range (0, len(data))]
plt.figure()
plt.scatter(data, y_data, color="red", marker="^")

# Normality probability plot
plt.figure()
s.probplot(data, dist="norm", plot=pylab)
pylab.show()
```

- **Métodos de Estadística Descriptiva Bivariante**

Estudio biométrico en una especie de arácnido fósil

En un estudio se mide en 200 arañas fósiles la longitud del tarso (X) y fémur (Y). Realizar un análisis de estadística descriptiva bivariante obteniendo la matriz de varianzas-covarianza, la matriz de correlación y el ajuste de los datos (estimación de los parámetros *a* y *b)* a la recta de regresión Y = a + b X:

- script: spider.py
- archivo de datos: spiders.dat spiders.csv

Solución: ejemplo21.mp4

```
#############################################################
#                                                           #
#       Virtual Laboratory of Geomathematics in Python      #
#                                                           #
#       Bivariate descriptive statistics  (01.06.2017)      #
#                                                           #
#    VLG Team, Complutense University of Madrid, Spain      #
#                                                           #
#   THIS SCRIPT IS PROVIDED BY THE AUTHORS "AS IS" AND      #
#   CAN BE USED BY ANYONE FOR THE PURPOSES OF EDUCATION     #
#   AND RESEARCH.                                           #
#                                                           #
#############################################################
import numpy as np
import scipy.stats as s
import statistics as ss
import matplotlib.pyplot as plt

# importing pandas
import pandas
from pandas.tools import plotting

import pylab

# Example of measurements of the tarsus and femur in fossil spiders
# Read data with numpy
col1, col2 = np.loadtxt('spiders.dat', unpack=True)
# Statistical summary
print('                  X         Y           ')
print('================================')
print('n            =',len(col1),'     ',len(col2))
print('Minimum      = %.2f' % min(col1),'     %.2f' %min(col2))
print('Maximum      = %.2f' % max(col1),'     %.2f' % max(col2))
print('Rank         = %.2f' % (max(col1)-min(col1)),'     %.2f' % (max(col2)-min(col2)))
print('Average      = %.2f' % ss.mean(col1),'     %.2f' % ss.mean(col2))
print('Median       = %.2f' % ss.median(col1),'     %.2f' % ss.median(col2))
print('Q1          = %.2f' % np.percentile(col1,25),'     %.2f' % np.percentile(col2,25))
```

```
print('Q2          = %.2f' % np.percentile(col1,50),'     %.2f' % np.percentile(col2,50))
print('Q3          = %.2f' % np.percentile(col1,75),'     %.2f' % np.percentile(col2,75))
print('Variance     = %.2f' % ss.variance(col1),'     %.2f' % ss.variance(col2))
print('Stand. dev.  = %.2f' % ss.stdev(col1),'     %.2f' % ss.stdev(col2))
print()
print('================================')
print("Correlation matrix: ")
print(np.corrcoef(col1,col2))
print()
print('================================')
print("Matrix of variance-covariance: ")
print(np.cov(col1,col2))
print()
print('================================')
print()

# Box-and-Whisker plot
# basic plot
plt.boxplot([col1,col2],0,' ')
# notched plot
plt.figure()
plt.boxplot([col1,col2], 1,' ')

# Scatter diagram with pandas
# Read data with pandas
data = pandas.read_csv('spiders.csv')
plotting.scatter_matrix(data, marker='o')

import statsmodels.api as sm
COL1=sm.add_constant(col1)
mod = sm.OLS(col2, COL1)
res = mod.fit()
print (res.summary())
print()

# Method 1. Regression table and scatter diagram
regresionlineal=s.linregress(col1,col2)
print('Linear regression = ',regresionlineal)
plt.figure()
plt.scatter(col1, col2, alpha=0.3)

# Method 2. Draw regression line
m, b = np.polyfit(col1, col2, deg=1)
plt.plot(col1, col2, '.')
plt.plot(col1, m*col1 + b, 'blue')

plt.ylim(0,12)  # Axis limits y
plt.xlabel("X") # X axis legend
plt.ylabel("Y") # Y axis legend
plt.title("Regression adjustment") # Title of the graph
pylab.show()
```

- **Métodos de Estadística Inferencial**

Descargar: Z-Table.py

```
#############################################################
#                                                           #
#         Virtual Laboratory of Statistics in Python        #
#                                                           #
#       Inferential statistics introduction (12.06.2017)    #
#                                                           #
#         Complutense University of Madrid, Spain           #
#                                                           #
#   THIS SCRIPT IS PROVIDED BY THE AUTHORS "AS IS" AND      #
#   CAN BE USED BY ANYONE FOR THE PURPOSES OF EDUCATION     #
#   AND RESEARCH.                                           #
#                                                           #
#############################################################
import scipy.stats as s

# Obtaining critical values in standard normal table Z:N(0,1)
```

```
# CASE A: For example, for a level of significance of 5%
alpha=0.05;
# area is the confidence level
area=1-alpha;
critical_value=s.norm.ppf(area);
print('Case A, -z_alpha = ', -critical_value);

# CASE B: For example, for a level of significance of 5%
alpha=0.05;
half_alpha=alpha/2;
# area is the confidence level and half of the significance level
area=(1-alpha)+half_alpha;
critical_value=s.norm.ppf(area);
print('Case B, -z_alpha/2 = ',-critical_value," z_alpha/2 = ",critical_value);


# CASE C: For example, for a level of significance of 5%
alpha=0.05;
# area is the confidence level
area=1-alpha;
critical_value=s.norm.ppf(area);
print('Case C, z_alpha = ',critical_value);
```

- **Métodos de Inferencia Estadística con una población**

Estudio de los niveles de sodio en agua

En un estudio geoquímico obtenemos el nivel de Na+ en 27 muestras de agua procedente de un pozo de EE.UU. ¿Tiene la variable "nivel de sodio" distribución normal? En caso afirmativo obtener los parámetros media y desviación típica de la distribución:

- script: sodium.py
- archivo de datos: Na.dat

Solución: ejemplo31.mp4

```
#############################################################
#                                                           #
#       Virtual Laboratory of Geomathematics in Python      #
#                                                           #
# Inferential statistics with one population  (18.07.2017) #
#                                                           #
#    VLG Team, Complutense University of Madrid, Spain      #
#                                                           #
#   THIS SCRIPT IS PROVIDED BY THE AUTHORS "AS IS" AND      #
#   CAN BE USED BY ANYONE FOR THE PURPOSES OF EDUCATION     #
#   AND RESEARCH.                                           #
#                                                           #
#############################################################
import math
import numpy as np
import scipy.stats as s
import matplotlib.pyplot as plt
import pylab

# ONE POPULATION

# Example sodium levels in water from a well located in the United States
data=np.loadtxt('Na.dat', skiprows=1)
print(data)

# Statistical summary
n, min_max, mean, var, skew, kurt = s.describe(data)
std=math.sqrt(var)
print()
print("============================================================")
```

```
print("n = ",n)
print('Average = ',mean)
print('Median = ',np.median(data))
print('Variance  =',var)
print('Stand. dev. = ',std)
print('Stand. error of the mean  =',std/math.sqrt(n))
print("================================================================")
print()

# Box-and-Whisker plot
plt.figure()
plt.boxplot(data, 1, 'rs', 0)

# Scatter plot
y_data=[np.random.random() for x in range (0, len(data))]
plt.figure()
plt.scatter(data, y_data, color="red", marker="^")

# Normal probability plot
plt.figure()
s.probplot(data, dist="norm", plot=pylab)
pylab.show()

# Gaussian histogram
plt.hist(data)
plt.title("Gaussian Histogram")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.show()

# Histogram
# histtype= normed=0 1 'bar' 'step', cumulative=0 1
bins=round(1+3.222*math.log10(len(data)))
plt.figure()
plt.hist(data,bins,normed=0,color='green',alpha=0.8, histtype='bar', cumulative=0)
pylab.show()

# Normality test
print()
print("Normality tests: ")
print()
normed_data=(data-mean)/std
kolmogorov_results=s.kstest(normed_data,'norm')
print("Kolmogorov = ",kolmogorov_results)
shapiro_results=s.shapiro(data)
print("Shapiro-Wilks = ",shapiro_results)
agostino_results=s.mstats.normaltest(data)
print("D'Agostino = ",agostino_results)
anderson_results=s.anderson(normed_data,'norm')
print("Anderson-Darling = ",anderson_results)
print()

# t-test one population
# Set up H0: Mean
H0=2713
print("H0 = ",H0,"  :")
print()
# test
t_stat, p_value_t = s.ttest_1samp(data,H0)
print("t-test : ",t_stat," p_value : ",p_value_t)
if p_value_t>=0.05:
    print("ACCEPT H0: Mean = ",mean)
else:
    print("REJECT H0: Mean = ",mean)
print()

# Wilcoxon Signed-Rank Test
# H0: Median
print("H0 = ",np.median(data),"  :")
print()
# test
w_stat, p_value_w = s.wilcoxon(data-np.median(data))
print("Wilcoxon test : ",w_stat," p_value : ",p_value_w)
if p_value_w>=0.05:
    print("ACCEPT H0: Median = ",np.median(data))
else:
    print("REJECT H0: Median = :",np.median(data))
```

```
print()

# Confidence Intervals
def sdev_interval(alpha, std, data):
    df  = n-1
    chi2upper = math.sqrt(s.chi2.ppf(1-alpha/2.0, df))
    chi2lower = math.sqrt(s.chi2.ppf(alpha/2.0, df))
    LRL = std * math.sqrt(n)/ chi2upper
    URL = std * math.sqrt(n)/ chi2lower
    return (LRL, URL)

def var_interval(alpha, var, data):
    LRL, URL = sdev_interval(alpha, math.sqrt(var), data)
    return (LRL*LRL, URL*URL)

# Choose level of significance
alpha=0.05
ci_mean = s.norm.interval(1-alpha,loc=mean,scale=std/math.sqrt(n))
print("Confidence intervals. Level of significance = ",alpha,"  :")
print()
print(" Mean :           ",ci_mean)

ci_sdev = sdev_interval(alpha, std, data)
print(" Stand. dev. :    ",ci_sdev)
ci_var = var_interval(alpha,var,data)
print(" Variance :       ",ci_var)
print()
```

Estudio de los niveles de calcio en agua

En un estudio geoquímico medimos el nivel del Ca2+ en 27 muestras de agua procedente de un pozo de EE.UU. (a) Obtener el intervalo de confianza para la media y la desviación típica poblacionales del nivel de calcio en agua. (b) Supóngase que el laboratorio de una empresa de prospección sostiene que para ese pozo el contenido medio de calcio es 85.26 y la desviación típica 25. Según sus datos ¿está de acuerdo con esta afirmación?

- script: calcium.py
- archivo de datos: Ca.dat

Solución: ejemplo32.mp4

```
#############################################################
#                                                           #
#       Virtual Laboratory of Geomathematics in Python      #
#                                                           #
# Inferential statistics with one population  (18.07.2017) #
#                                                           #
#    VLG Team, Complutense University of Madrid, Spain      #
#                                                           #
#   THIS SCRIPT IS PROVIDED BY THE AUTHORS "AS IS" AND      #
#   CAN BE USED BY ANYONE FOR THE PURPOSES OF EDUCATION     #
#   AND RESEARCH.                                           #
#                                                           #
#############################################################
import math
import numpy as np
import scipy.stats as s
import matplotlib.pyplot as plt
import pylab

# ONE POPULATION

# Example calcium levels in water from a well located in the United States
data=np.loadtxt('Ca.dat', skiprows=1)
print(data)

# Statistical summary
n, min_max, mean, var, skew, kurt = s.describe(data)
std=math.sqrt(var)
print()
```

```
print("================================================================")
print("n = ",n)
print('Average = ',mean)
print('Median = ',np.median(data))
print('Variance  =',var)
print('Stand. dev. = ',std)
print('Stand. error of the mean  =',std/math.sqrt(n))
print("================================================================")
print()

# Box-and-Whisker plot
plt.figure()
plt.boxplot(data, 1, 'rs', 0)

# Scatter plot
y_data=[np.random.random() for x in range (0, len(data))]
plt.figure()
plt.scatter(data, y_data, color="red", marker="^")

# Normal probability plot
plt.figure()
s.probplot(data, dist="norm", plot=pylab)
pylab.show()

# Gaussian histogram
plt.hist(data)
plt.title("Gaussian Histogram")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.show()

# Histogram
# histtype= normed=0 1 'bar' 'step', cumulative=0 1
bins=round(1+3.222*math.log10(len(data)))
plt.figure()
plt.hist(data,bins,normed=0,color='green',alpha=0.8, histtype='bar', cumulative=0)
pylab.show()

# Normality test
print()
print("Normality tests: ")
print()
normed_data=(data-mean)/std
kolmogorov_results=s.kstest(normed_data,'norm')
print("Kolmogorov = ",kolmogorov_results)
shapiro_results=s.shapiro(data)
print("Shapiro-Wilks = ",shapiro_results)
agostino_results=s.mstats.normaltest(data)
print("D'Agostino = ",agostino_results)
anderson_results=s.anderson(normed_data,'norm')
print("Anderson-Darling = ",anderson_results)
print()

# t-test one population
# Set up H0: Mean
H0=85.26
print("H0 = ",H0,"  :")
print()
# test
t_stat, p_value_t = s.ttest_1samp(data,H0)
print("t-test : ",t_stat," p_value : ",p_value_t)
if p_value_t>=0.05:
    print("ACCEPT H0: Mean = ",mean)
else:
    print("REJECT H0: Mean = ",mean)
print()

# Wilcoxon Signed-Rank Test
# H0: Median
print("H0 = ",np.median(data),"  :")
print()
# test
w_stat, p_value_w = s.wilcoxon(data-np.median(data))
print("Wilcoxon test : ",w_stat," p_value : ",p_value_w)
if p_value_w>=0.05:
    print("ACCEPT H0: Median = ",np.median(data))
else:
```

```
    print("REJECT H0: Median = :",np.median(data))
print()

# Confidence Intervals
def sdev_interval(alpha, std, data):
    df  = n-1
    chi2upper = math.sqrt(s.chi2.ppf(1-alpha/2.0, df))
    chi2lower = math.sqrt(s.chi2.ppf(alpha/2.0, df))
    LRL = std * math.sqrt(n)/ chi2upper
    URL = std * math.sqrt(n)/ chi2lower
    return (LRL, URL)

def var_interval(alpha, var, data):
    LRL, URL = sdev_interval(alpha, math.sqrt(var), data)
    return (LRL*LRL, URL*URL)

# Choose level of significance
alpha=0.05
ci_mean = s.norm.interval(1-alpha,loc=mean,scale=std/math.sqrt(n))
print("Confidence intervals. Level of significance = ",alpha,"  :")
print()
print(" Mean :           ",ci_mean)

ci_sdev = sdev_interval(alpha, std, data)
print(" Stand. dev. :    ",ci_sdev)
ci_var = var_interval(alpha,var,data)
print(" Variance :       ",ci_var)
print()
```

- Métodos de Inferencia Estadística con dos poblaciones

Comparación del caudal de un rio en dos estaciones

En un estudio se desea comparar el caudal máximo anual del Rio James (Virginia, EE.UU.) medido en dos épocas diferentes. ¿Qué estadístico utilizaremos? ¿por qué? ¿Qué concluimos en este estudio?

- script: River.py
- archivo de datos: River.dat

Solución: ejemplo41.mp4

```
#################################################################
#                                                               #
#       Virtual Laboratory of Geomathematics in Python          #
#                                                               #
# Inferencial statistics with two populations (25.06.2017)      #
#                                                               #
#    VLG Team, Complutense University of Madrid, Spain          #
#                                                               #
#   THIS SCRIPT IS PROVIDED BY THE AUTHORS "AS IS" AND          #
#   CAN BE USED BY ANYONE FOR THE PURPOSES OF EDUCATION         #
#   AND RESEARCH.                                               #
#                                                               #
#################################################################
import math
import numpy as np
import scipy.stats as s
import statistics as ss
import matplotlib.pyplot as plt
import pylab

# TWO POPULATIONS

# Example James River flow comparison in two stations
col1, col2 = np.loadtxt('River.dat', unpack=True)

# Summary statistics
print('           Group 1       Group 2           ')
```

```
print('================================')
print('n              =',len(col1),'      ',len(col2))
print('Minimum        = %.2f' % min(col1),'     %.2f' %min(col2))
print('Maximum        = %.2f' % max(col1),'     %.2f' % max(col2))
print('Rank           = %.2f' % (max(col1)-min(col1)),'     %.2f' % (max(col2)-min(col2)))
print('Average        = %.2f' % ss.mean(col1),'     %.2f' % ss.mean(col2))
print('Median         = %.2f' % ss.median(col1),'     %.2f' % ss.median(col2))
print('Q1            = %.2f' % np.percentile(col1,25),'     %.2f' % np.percentile(col2,25))
print('Q2            = %.2f' % np.percentile(col1,50),'     %.2f' % np.percentile(col2,50))
print('Q3            = %.2f' % np.percentile(col1,75),'     %.2f' % np.percentile(col2,75))
print('Variance       = %.2f' % ss.variance(col1),'     %.2f' % ss.variance(col2))
print('Stand. dev.    = %.2f' % ss.stdev(col1),'     %.2f' % ss.stdev(col2))
print('Stand. error of the mean = %.2f' % s.sem(col1),'     %.2f' % s.sem(col2))

# Box-and-Whisker plot
plt.figure()
plt.boxplot([col1,col2], 1,' ')

# Scatter plot
y_data=[np.random.random() for x in range (0, len(col1))]
plt.figure()
plt.scatter(col1, y_data, color="red", marker="^")

y_data=[np.random.random() for x in range (0, len(col2))]
plt.figure()
plt.scatter(col2, y_data, color="red", marker="+")


# Normal probability plot
plt.figure()
s.probplot(col1, dist="norm", plot=pylab)
pylab.show()

plt.figure()
s.probplot(col2, dist="norm", plot=pylab)
pylab.show()

# Gaussian histogram
plt.hist(col1)
plt.title("Gaussian Histogram")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.show()

plt.hist(col2)
plt.title("Gaussian Histogram")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.show()

# Histogram
# histtype= normed=0 1 'bar' 'step', cumulative=0 1
bins1=round(1+3.222*math.log10(len(col1)))
bins2=round(1+3.222*math.log10(len(col2)))
plt.figure()
plt.hist(col1,bins1,normed=0,color='green',alpha=0.8, histtype='bar', cumulative=0)
plt.hist(col2,bins2,normed=0,color='yellow',alpha=0.8, histtype='bar', cumulative=0)
pylab.show()


# Normality test
print()
print("Normality tests (Sample 1): ")
print()
normed_datos=(col1-ss.mean(col1))/ss.stdev(col1)
kolmogorov_results=s.kstest(normed_datos,'norm')
print("Kolmogorov = ",kolmogorov_results)
shapiro_results=s.shapiro(col1)
print("Shapiro-Wilks = ",shapiro_results)
agostino_results=s.mstats.normaltest(col1)
print("D'Agostino = ",agostino_results)
anderson_results=s.anderson(normed_datos,'norm')
print("Anderson-Darling = ",anderson_results)
print()
print()
print("Normality tests (Sample 2): ")
print()
```

```
normed_datos=(col1-ss.mean(col2))/ss.stdev(col2)
kolmogorov_results=s.kstest(normed_datos,'norm')
print("Kolmogorov = ",kolmogorov_results)
shapiro_results=s.shapiro(col2)
print("Shapiro-Wilks = ",shapiro_results)
agostino_results=s.mstats.normaltest(col2)
print("D'Agostino = ",agostino_results)
anderson_results=s.anderson(normed_datos,'norm')
print("Anderson-Darling = ",anderson_results)
print()

print()
print("Test for homogeneity of variance: ")
print()
# F-test (Fisher test)
alpha = 0.05
F = ss.variance(col1) / ss.variance(col2)
df1 = len(col1) - 1; df2 = len(col2) - 1
p_value1 = s.f.cdf(F, df1, df2)
# p_value2=s.f(df1, df2).cdf(F)  Equivalent expression
print('F-test:',F,'  p-value = ',2*p_value1);
print()
bart_result,p_value=s.bartlett(col1,col2)
print('Bartlett test: ',bart_result,' p-value = ',p_value)
print()
levene_result,p_value=s.levene(col1,col2)
print('Levene test: ',levene_result,' p-value = ',p_value)


# Two samples t-test
# test assumes the two groups have the same variance...
t_stat, p_value = s.ttest_ind(col1, col2)
# test assumes the two groups have different variances...
t_stat_u, p_value_u = s.ttest_ind(col1, col2, equal_var=False)
print()
print('Test for means:')
print("============================================================")
print("t-test (equal variance) : ",t_stat," ",p_value)
if p_value>=0.05:
    print("ACCEPT H0")
else:
    print("REJECT H0")
print()
print("t-test (different variance) : ",t_stat_u," ",p_value_u)
if p_value_u>=0.05:
    print("ACCEPT H0")
else:
    print("REJECT H0")
print()
# Two samples Mann Whitney U test
u_stat, p_value = s.mannwhitneyu(col1,col2)
print("Wilcoxon test (Mann Whitney U test): ",u_stat," ",p_value)
if p_value>=0.05:
    print("ACCEPT H0")
else:
    print("REJECT H0")
print()
```

- **Métodos de Inferencia Estadística con Proporciones y Frecuencias**

Análisis granulométrico del suelo ¿es el suelo apto para la agricultura?

En un terreno se obtiene una muestra de tierra, realizándose un análisis de su granulometría. El resultado del análisis fue que de un total de 42 fracciones de tierra 15 son arenas, el suelo ¿es apto para el cultivo? ¿Qué supuesto debe verificarse para poder utilizar este estadístico? ¿Qué distribución tiene la variable "proporción muestral"?

Utilizar la siguiente tabla. Identificación del tipo de suelo según su granulometría:

Tabla.- Clases de suelo

Arcilloso: 25% de arenas, 75% limo y arcillas. Muy porosos, poca aireación. Retienen mucha agua. No son aptos para la agricultura.
Arenoso: 75% de arenas, 25% limo y arcillas. Gran aireación, no retienen agua. No son aptos para la agricultura.
Francos: 45% de arenas, 55% limo y arcillas. Texturas media. Si son aptos para la agricultura.

- script: Soil.py

Solución: ejemplo51.mp4

```
############################################################
#                                                          #
#       Virtual Laboratory of Geomathematics in Python     #
#                                                          #
#            Enumerative Data Models (18.08.2017)          #
#                                                          #
#    VLG Team, Complutense University of Madrid, Spain     #
#                                                          #
#   THIS SCRIPT IS PROVIDED BY THE AUTHORS "AS IS" AND     #
#   CAN BE USED BY ANYONE FOR THE PURPOSES OF EDUCATION    #
#   AND RESEARCH.                                          #
#                                                          #
############################################################

from statsmodels.stats.proportion import proportions_ztest
# Example. Granulometric analysis of soil, Is the soil suitable for agriculture?
# ONE PROPORTION
x = 15
n = 42
H0 = 0.45
# alternative : string in [‘two-sided’, ‘smaller’, ‘larger’]
HA = 'smaller'
z,p = proportions_ztest(x, n, H0, HA)
print()
print(' z-stat. one proportion = {z} \n p-value = {p}'.format(z=z,p=p))
print()
```

Incidencia del cáncer en un pueblo de la provincia de Orellana (Ecuador)

En un estudio realizado en un pueblo de la provincia de Orellana (Ecuador) se obtuvieron los porcentajes de enfermos de cáncer (E) y personas sanas (S) en dos ubicaciones distintas tomando como referencia la presencia de un pozo de petróleo. En las personas allí residentes que bebían agua obtenida de una fuente a menos de 50 m del pozo de petróleo, el porcentaje de personas S y E era igual al 43% y 57% respectivamente. Por el contrario, en los sujetos que bebían agua de una fuente situada a más de 250 m del pozo de petróleo, el porcentaje de personas S y E era del 94.3% y 5.7% respectivamente ¿Qué podemos concluir?

- script: cancer.py

Solución: ejemplo52.mp4

```
#############################################################
#                                                           #
#       Virtual Laboratory of Geomathematics in Python      #
#                                                           #
#            Enumerative Data Models (18.08.2017)           #
#                                                           #
#    VLG Team, Complutense University of Madrid, Spain      #
#                                                           #
#   THIS SCRIPT IS PROVIDED BY THE AUTHORS "AS IS" AND      #
#   CAN BE USED BY ANYONE FOR THE PURPOSES OF EDUCATION     #
#   AND RESEARCH.                                           #
#                                                           #
#############################################################

import scipy.stats as s
import numpy as np
from statsmodels.stats.proportion import proportions_ztest
#Example. Incidence of cancer in a town in the province of Orellana (Ecuador)
# TWO PROPORTIONS
n1 = 100
x1 = 57
n2 = 100
x2 = 5.7
H0 = 0.0
# alternative : string in [‘two-sided’, ‘smaller’, ‘larger’]
HA='larger'
x12 = np.array([x1,x2])
n12 = np.array([n1, n2])
z,p = proportions_ztest(x12, n12, H0, HA)
print()
print(' z-stat. two proportions = {z} \n p-value = {p}'.format(z=z,p=p))
print()

# CHI-SQUARE TEST

# 2x2 - CONTINGENCE TABLE
data_observed = np.array([[43,94.3], [57,5.7]])
chi2_result,p_value,df_value,table_expected=s.chi2_contingency(data_observed)
print()
confidence_level = 0.95
crit_value = s.chi2.ppf(q = confidence_level, df=df_value)
print()
oddsratio, p_value_f = s.fisher_exact(data_observed)
print ('Contingency table 2x2')
print ('====================================')
print ('chi2:', chi2_result)
print ('p-value:', p_value)
print ('df = ', df_value)
print('Critical value: ',crit_value)
print()
print()
print('        Fisher exact test:')
print()
print(' Oddsratio: ',oddsratio,' p-value: ',p_value_f)
print()
print ('              Table')
print ('Obs.: ')
print(data_observed)
print ('Exp.: ')
print(table_expected)
print ('====================================')
print()
print()
```

- **Modelos Generales Lineales: ANOVA**

Análisis del contenido en cobre en plantas cultivadas en tres clases de suelos ácidos

Un laboratorio analiza el contenido en cobre en 33 ejemplares de una especie de planta procedente de tres terrenos diferentes (k=3) caracterizados por su diferente acidez. De cada terreno se obtuvieron 11

plantas aleatoriamente, tras el análisis ¿afecta la clase de suelo (su acidez) a la presencia de cobre en la planta?

- script: Copper.py
- archivo de datos: copper.dat, copper2.dat

Solución: ejemplo71.mp4

```
#############################################################
#                                                           #
#       Virtual Laboratory of Geomathematics in Python      #
#                                                           #
#     General Linear Models: One way ANOVA  (27.07.2017)    #
#                                                           #
#    VLG Team, Complutense University of Madrid, Spain      #
#                                                           #
#   THIS SCRIPT IS PROVIDED BY THE AUTHORS "AS IS" AND      #
#   CAN BE USED BY ANYONE FOR THE PURPOSES OF EDUCATION     #
#   AND RESEARCH.                                           #
#                                                           #
#############################################################
import math
import numpy as np
import scipy.stats as s
import statistics as ss
import matplotlib.pyplot as plt

import pylab

#from statsmodels.stats.multicomp import pairwise_tukeyhsd
from statsmodels.stats.multicomp import MultiComparison

# ANOVA ONE-WAY

# Example Presence of copper in a species of plant cultivated in three
# different soils
col1, col2, col3 = np.loadtxt('copper.dat', unpack=True)
col_Cu, col_Group = np.loadtxt('copper2.dat', unpack=True)

# Box-and-Whisker plot
plt.figure()
plt.boxplot([col1,col2, col3], 1,' ')

# Scatter plot
y_data=[np.random.random() for x in range (0, len(col1))]
plt.figure()
plt.scatter(col1, y_data, color="red", marker="^")

y_data=[np.random.random() for x in range (0, len(col2))]
plt.figure()
plt.scatter(col2, y_data, color="red", marker="+")

y_data=[np.random.random() for x in range (0, len(col3))]
plt.figure()
plt.scatter(col2, y_data, color="red", marker="+")

# Normal probability plot
plt.figure()
s.probplot(col1, dist="norm", plot=pylab)
pylab.show()

plt.figure()
s.probplot(col2, dist="norm", plot=pylab)
pylab.show()

plt.figure()
s.probplot(col3, dist="norm", plot=pylab)
pylab.show()

# Histogram
# histtype= normed=0 1 'bar' 'step', cumulative=0 1
bins1=round(1+3.222*math.log10(len(col1)))
```

```
bins2=round(1+3.222*math.log10(len(col2)))
bins3=round(1+3.222*math.log10(len(col3)))
plt.figure()
plt.hist(col1,bins1,normed=0,color='green',alpha=0.8, histtype='bar', cumulative=0)
plt.hist(col2,bins2,normed=0,color='yellow',alpha=0.8, histtype='bar', cumulative=0)
plt.hist(col3,bins3,normed=0,color='blue',alpha=0.8, histtype='bar', cumulative=0)
pylab.show()


# Normality tests
print()
print("Normality tests (Sample 1): ")
print()
normed_datos=(col1-ss.mean(col1))/ss.stdev(col1)
kolmogorov_results=s.kstest(normed_datos,'norm')
print("Kolmogorov = ",kolmogorov_results)
shapiro_results=s.shapiro(col1)
print("Shapiro-Wilks = ",shapiro_results)
agostino_results=s.mstats.normaltest(col1)
print("D'Agostino = ",agostino_results)
anderson_results=s.anderson(normed_datos,'norm')
print("Anderson-Darling = ",anderson_results)
print()
print()
print("Normality tests (Sample 2): ")
print()
normed_datos=(col2-ss.mean(col2))/ss.stdev(col2)
kolmogorov_results=s.kstest(normed_datos,'norm')
print("Kolmogorov = ",kolmogorov_results)
shapiro_results=s.shapiro(col2)
print("Shapiro-Wilks = ",shapiro_results)
agostino_results=s.mstats.normaltest(col2)
print("D'Agostino = ",agostino_results)
anderson_results=s.anderson(normed_datos,'norm')
print("Anderson-Darling = ",anderson_results)
print()
print()
print("Normality tests (Sample 3): ")
print()
normed_datos=(col3-ss.mean(col3))/ss.stdev(col3)
kolmogorov_results=s.kstest(normed_datos,'norm')
print("Kolmogorov = ",kolmogorov_results)
shapiro_results=s.shapiro(col3)
print("Shapiro-Wilks = ",shapiro_results)
agostino_results=s.mstats.normaltest(col2)
print("D'Agostino = ",agostino_results)
anderson_results=s.anderson(normed_datos,'norm')
print("Anderson-Darling = ",anderson_results)
print()


print()
print("Test for homogeneity of variance: ")
print()
bart_result,p_value=s.bartlett(col1,col2,col3)
print('Bartlett test: ',bart_result,' p-value = ',p_value)
if p_value<0.05:
    print("homoscedasticity is not met in the ANOVA")
print()
levene_result,p_value=s.levene(col1,col2,col3)
print('Levene test: ',levene_result,' p-value = ',p_value)
if p_value<0.05:
    print("homoscedasticity is not met in the ANOVA")
print()

# ANOVA one-way
print()
k=3 # Define the number of treated groups
N=33 # Total data or individuals
F_result, p_value=s.f_oneway(col1,col2,col3)
print ('One-way ANOVA')
print ('==============================')
print ('F value:', F_result)
print ('p-value:', p_value, '\n')
print()
print('Between Groups df = ', k-1)
print('Within Groups df = ',N-k)
```

```
    print()
    print('Total df = ', N - 1)
    print ('==============================')
    print()
    mc = MultiComparison(col_Cu,col_Group)
    result = mc.tukeyhsd()
    print(result)
    print(mc.groupsunique)
    print()
    print()
    # Kruskal-Wallis
    print()
    H_result, p_value=s.kruskal(col1,col2,col3)
    print ('Kruskal-Wallis H-test')
    print ('==============================')
    print ('H value:', H_result)
    print ('p-value:', p_value, '\n')
```

- **Modelos Generales Lineales: REGRESIÓN LINEAL**

Absorción de fosfato en una planta

Supóngase que en un grupo formado por 9 plantas analizamos la cantidad de fosfato X presente en la tierra de cada maceta (archivo phosphate, columna izquierda) y la cantidad de fosfato Y asimilado por la planta (archivo phosphate, columna derecha). Construir un modelo de regresión lineal entre X e Y.

- script: Phosphate.py
- archivo de datos: phosphate.dat

Solución: ejemplo61.mp4

```
#############################################################
#                                                           #
#       Virtual Laboratory of Geomathematics in Python      #
#                                                           #
#    General Linear Models: Linear Regression (01.08.2017)  #
#                                                           #
#    VLG Team, Complutense University of Madrid, Spain      #
#                                                           #
#   THIS SCRIPT IS PROVIDED BY THE AUTHORS "AS IS" AND      #
#   CAN BE USED BY ANYONE FOR THE PURPOSES OF EDUCATION     #
#   AND RESEARCH.                                           #
#                                                           #
#############################################################
import math
import numpy as np
import scipy.stats as s
import statistics as ss
import statsmodels.api as sm
import matplotlib.pyplot as plt

import pylab

# LINEAR REGRESSION
# Example absorption of phosphate in a plant: assimilated phosphate (col2) and
# phosphate in soil (col1)
col1, col2 = np.loadtxt('phosphate.dat', unpack=True)

# Box-and-Whisker plot
plt.figure()
plt.boxplot([col1,col2], 1,' ')

# Scatter plot
y_data=[np.random.random() for x in range (0, len(col1))]
plt.figure()
plt.scatter(col1, y_data, color="red", marker="^")
```

```
y_data=[np.random.random() for x in range (0, len(col2))]
plt.figure()
plt.scatter(col2, y_data, color="red", marker="+")

# Normal probability plot
plt.figure()
s.probplot(col1, dist="norm", plot=pylab)
pylab.show()

plt.figure()
s.probplot(col2, dist="norm", plot=pylab)
pylab.show()

# Gaussian histogram
plt.hist(col1)
plt.title("Gaussian Histogram")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.show()

plt.hist(col2)
plt.title("Gaussian Histogram")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.show()

# Histogram
# histtype= normed=0 1 'bar' 'step', cumulative=0 1
bins1=round(1+3.222*math.log10(len(col1)))
bins2=round(1+3.222*math.log10(len(col2)))
plt.figure()
plt.hist(col1,bins1,normed=0,color='green',alpha=0.8, histtype='bar', cumulative=0)
plt.hist(col2,bins2,normed=0,color='yellow',alpha=0.8, histtype='bar', cumulative=0)
pylab.show()


# Normality test
print()
print("Normality tests (Variable 1): ")
print()
normed_datos=(col1-ss.mean(col1))/ss.stdev(col1)
kolmogorov_results=s.kstest(normed_datos,'norm')
print("Kolmogorov = ",kolmogorov_results)
shapiro_results=s.shapiro(col1)
print("Shapiro-Wilks = ",shapiro_results)
agostino_results=s.mstats.normaltest(col1)
print("D'Agostino = ",agostino_results)
anderson_results=s.anderson(normed_datos,'norm')
print("Anderson-Darling = ",anderson_results)
print()
print()
print("Normality tests (Variable 2): ")
print()
normed_datos=(col2-ss.mean(col2))/ss.stdev(col2)
kolmogorov_results=s.kstest(normed_datos,'norm')
print("Kolmogorov = ",kolmogorov_results)
shapiro_results=s.shapiro(col2)
print("Shapiro-Wilks = ",shapiro_results)
agostino_results=s.mstats.normaltest(col2)
print("D'Agostino = ",agostino_results)
anderson_results=s.anderson(normed_datos,'norm')
print("Anderson-Darling = ",anderson_results)
print()
print()

# Simple linear regression
print()
# Method 1
slope_result, intercept_result, r_value, p_value, std_err_result=s.linregress(col1,col2)
print ('Linear regression ')
print ('==============================')
print ('slope:', slope_result)
print ('intercept:', intercept_result)
print ('correlation coefficient:', r_value)
print ('p-value:', p_value)
print ('Standard error of the estimate:',std_err_result , '\n')
print()
```

```
print ('================================')
print()
# Method 2: Note that X=col1 e Y=col2
print()
results = sm.OLS(col2,sm.add_constant(col1)).fit()
print (results.summary())
plt.scatter(col1,col2)
m, b = np.polyfit(col1, col2, deg=1)
plt.plot(col1, col2, '.')
plt.plot(col1, m*col1 + b, 'blue')
plt.show()
```

- **Modelos Geomatemáticos**

- **Análisis de la Distribución de Frecuencia de los Terremotos en Función de su Magnitud (Ley de Gutenberg-Richter)**

Aplicación de la Ley de Gutenberg-Richter en la Península Ibérica

- script: Earthquake.py Earthquake.py (sin ventanas)

- Archivos de datos en Excel:

  Catalogo_CMT_Iberia.xls
  Catalogo_EHB_Iberia.xls
  Catalogo_IGN_Iberia.xls

```
############################################################
#                                                          #
#     Virtual Laboratory of Geomathematics in Python       #
#                                                          #
#   Analysis of Frequency Distribution of Earthquakes:     #
#          Gutenbeg-Richter's Law   (26.09.2017)           #
#                                                          #
#         Complutense University of Madrid, Spain          #
#                                                          #
#   THIS SCRIPT IS PROVIDED BY THE AUTHORS "AS IS" AND     #
#   CAN BE USED BY ANYONE FOR THE PURPOSES OF EDUCATION    #
#   AND RESEARCH.                                          #
#                                                          #
############################################################
import sys

from matplotlib.widgets import Cursor
from matplotlib.gridspec import GridSpec
import matplotlib.pylab as plt
import numpy as np
from math import log10

import xlrd
import easygui as eg

# Ventana informacion
eg.msgbox("In this script:\n 1) We selected a file with earthquakes magnitude information\n
2) An histogram of frequency is plotted\n 3) The cumulative number (for >mag) is also
plotted in log(N) vs mag format\n 4) The limits for the Gutenberg-Richter fit are selected
using a cursor\n 5) The cumulative number and the line of fit are plotted
together",title='Problem 2', ok_button='Continue')


# Eleccion de fichero
seleccion = ["*.xls","*.xlsx","*.ods"]
fichero=eg.fileopenbox(msg="Please choose a file",title="Data
file",default="*",filetypes=seleccion)
print(fichero)

# Funcion de apertura y lectura de la columna n de un fichero excel
def read_excel(fichero,n):
        book = xlrd.open_workbook(fichero)
        sh = book.sheet_by_index(0)
```

```
        # Metemos la columna n en una lista
        coln=[]
        for rx in range(1,sh.nrows):
               coln.append(float(sh.cell_value(colx=n-1,rowx=rx)))
        return coln

# Si no se ha seleccionado el fichero se cierra el programa
if fichero == None:
     print("No file selected")
     sys.exit()

# Lee las magnitudes
num = eg.integerbox(msg='Column number for the magnitudes (1-100):',
                    title='Magnitudes',
                    default=13,
                    lowerbound=1,
                    upperbound=100,
                    image=None)
if num == None:
       print("No column selected")
       sys.exit()
mag=read_excel(fichero,num)

# Limites en el eje x de la grafica de magnitudes y numero de intervalos
magmin=int(min(mag))
magmax=int(max(mag))+1
etlim=['Mag min', 'Mag max', 'Number of bins']
xlimi=eg.multenterbox(msg='Data      for      the      histogram',      title='Magnitude
histogram',fields=etlim,values=(magmin,magmax,20))
xmin=float(xlimi[0])
xmax=float(xlimi[1])
nbin=int(xlimi[2])
tbin=(xmax-xmin)/float(nbin)

# Almacenamos los datos del histograma
histograma, bines=np.histogram(mag,bins=nbin,range=(xmin,xmax))
posb=[]
for i in range(1,len(bines)):
         posb.append((bines[i]+bines[i-1])*0.5)

# Frecuencia acumulada
fac=[]
freini=0
for i in range(len(posb)-1,-1,-1):
    freini=freini+histograma[i]
    fac.append(freini)
prev=posb[::-1]

# Dibujamos graficas
fig=plt.figure('Magnitude')
fig.subplots_adjust(wspace=0.3)

# Histograma de las magnitudes
gs = GridSpec(2, 2, height_ratios=[10,1])
ax1=fig.add_subplot(gs[:,0])
ax1.tick_params(direction='in')
ax1.yaxis.set_ticks_position('both')
ax1.xaxis.set_ticks_position('both')
ax1.hist(mag,bins=bines)
ax1.set_title("    "+fichero)
ax1.set_ylabel("Frequency")
ax1.set_xlabel("Magnitude")

# Grafica de log de la frecuencia acumulada
ax2=fig.add_subplot(gs[1])
ax2.tick_params(direction='in')
ax2.yaxis.set_ticks_position('both')
ax2.xaxis.set_ticks_position('both')
ax2.semilogy(prev,fac,'bs')
ax2.set_ylabel("Cumulative frequency (>mag)")
ax2.set_xlabel("Magnitude")

# Mensaje de  seleccion
textstring='Selected in the above cumulative frequency plot\n the two limits for the fit'
axm=fig.add_subplot(gs[3])
axm.text(0,0,textstring,fontsize=6,bbox={'facecolor':'blue', 'alpha':0.2, 'pad':1})
```

```
axm.axis('off')

# Seleccionamos rango de magnitudes para el ajuste
coords = []
cursor = Cursor(ax2,color='r',lw=1,horizOn=False,vertOn=True)

def onclick_x(event):
        if event.xdata != None and event.ydata != None:
              print(event.xdata)

        global coords
        coords.append(event.xdata)

        if len(coords)==2:
              fig.canvas.mpl_disconnect(cid)
              plt.close('Magnitude')
        return

cid = fig.canvas.mpl_connect('button_press_event', onclick_x)


plt.show()
plt.close()

## Limites para el ajuste
amin=min(coords)
amax=max(coords)
majus=[]
fajus=[]
for i in range(len(prev)):
        if prev[i]>amin and prev[i]<amax:
              majus.append(prev[i])
              fajus.append(log10(fac[i]))

fit=np.polyfit(majus,fajus,1)
m=fit[0]
b=fit[1]
ajuste=np.poly1d(fit)
la="log(y)="+str(m)+"x+"+str(b)
co=np.corrcoef(majus,fajus)[0,1]
print(la)
print('r=',co)

# Dibuja los datos y el ajuste
fig2=plt.figure('Fit')
ax3=fig2.add_subplot(111)
ax3.tick_params(direction='in')
ax3.yaxis.set_ticks_position('both')
ax3.xaxis.set_ticks_position('both')
# Datos
ax3.semilogy(prev,fac,'bs',label='data')
# Ajuste
ax3.semilogy(majus,10**ajuste(majus),'r-',label='fit: log(y)=%5.3fx+%5.3f\n    r=%6.4f' %
(m, b, co))
ax3.legend(loc=1)
ax3.set_ylabel("Cumulative frequency (>mag)")
ax3.set_xlabel("Magnitude")
ax3.set_title(fichero)

plt.show()
```

- **Estudio de un Perfil Topográfico Mediante el Análisis de sus Derivadas Primera y Segunda**

Realización de un Perfil Topográfico en dos cerros de Guadalajara (España)

El ejemplo se realiza con datos de los cerros El Colmillo y La Muela en la provincia de Guadalajara.

- script: Cerros.py Cerros.py (sin ventanas)
- Archivos de datos en Excel:

perfil_colmillo.xls
perfil_muela.xls

```
############################################################
#                                                          #
#       Virtual Laboratory of Geomathematics in Python     #
#                                                          #
#   Study of the topographic profile of a hill(26.09.2017) #
#                                                          #
#         Complutense University of Madrid, Spain          #
#                                                          #
#   THIS SCRIPT IS PROVIDED BY THE AUTHORS "AS IS" AND     #
#   CAN BE USED BY ANYONE FOR THE PURPOSES OF EDUCATION    #
#   AND RESEARCH.                                          #
#                                                          #
############################################################
import sys

from pylab import *

import xlrd
import easygui as eg
import numpy as np
from scipy.signal import savgol_filter

# Ventana informacion
eg.msgbox("In this script:\n 1) We selected a file with a height profile\n 2) The height
profile is plotted\n 3) First derivative (slope) and second derivative are calculated\n
4) Both of them are plotted",title='Problem 1', ok_button='Continue')

# Eleccion de fichero
seleccion = ["*.xls","*.xlsx","*.ods"]
fichero=eg.fileopenbox(msg="Please choose a file",title="Data
file",default="*",filetypes=seleccion)
print(fichero)

# Funcion de apertura y lectura de la columna n de un fichero excel
def read_excel(fichero,n):
        book = xlrd.open_workbook(fichero)
        sh = book.sheet_by_index(0)

        # Metemos la columna n en una lista
        coln=[]
        for rx in range(1,sh.nrows):
               coln.append(float(sh.cell_value(colx=n-1,rowx=rx)))
        return coln

# Si no se ha seleccionado el fichero se cierra el programa
if fichero == None:
     print("No file selected")
     sys.exit()

# Lee las posiciones en x
numx = eg.integerbox(msg='Column number for the position (1-100):',
                     title='Position',
                     default=4,
                     lowerbound=1,
                     upperbound=100,
                     image=None)
if numx == None:
       print("No column selected")
       sys.exit()
posx=read_excel(fichero,numx)

# Lee las posiciones en y
numy = eg.integerbox(msg='Column number for the height (1-100):',
                     title='Height',
                     default=3,
                     lowerbound=1,
                     upperbound=100,
                     image=None)
if numy == None:
       print("No column selected")
       sys.exit()
altura=read_excel(fichero,numy)
```

```
## Lee las derivadas
#numz = eg.integerbox(msg='Column number for the slope (1-100):',
#                     title='Slope',
#                     default='',
#                     lowerbound=1,
#                     upperbound=100,
#                     image=None)
#if numz == None:
#       print("No column selected")
#       sys.exit()
#derv=read_excel(fichero,numz)

# Suavizado de la altura con un filtro Savitzky-Golay
smaltura=savgol_filter(altura, 11, 4)

# Calculamos la primera derivada
pos1=[]
der1=[]  # Primera derivada
for i in range(0,len(posx)-2):
       pos1.append((posx[i+1]+posx[i])*0.5)
       der1.append((altura[i+1]-altura[i])/(posx[i+1]-posx[i]))

# Suavizado de la primera derivada con un filtro Savitzky-Golay
smder1=savgol_filter(der1, 21, 4, mode='nearest')

# Calculamos la segunda derivada
pos2=[]
der2=[]  # Segunda derivada
der2s=[] # Segunda derivada a partir de la primera suaviazada
for i in range(0,len(pos1)-2):
       pos2.append((pos1[i+1]+pos1[i])*0.5)
       der2.append((der1[i+1]-der1[i])/(pos1[i+1]-pos1[i]))
       der2s.append((smder1[i+1]-smder1[i])/(pos1[i+1]-pos1[i]))

# Suavizado de la segunda derivada con un filtro Savitzky-Golay
smder2=savgol_filter(der2, 21, 4, mode='nearest')
# Suavizado de la segunda derivada obtenida a partir de la primera suavizada con un filtro
Savitzky-Golay
smder2s=savgol_filter(der2s, 21, 4, mode='nearest')

# Limites en el eje x de la grafica
etlim=['X min', 'X max']
minx=int(min(posx))
maxx=int(max(posx))
xlimi=eg.multenterbox(msg='Limits in position', title='X
limits',fields=etlim,values=(minx,maxx))

# GRAFICAS
fig=figure()
fig.subplots_adjust(hspace=0.0)

# Grafica de la altura
ax1=fig.add_subplot(311)                          # Posicion de la grafica de altura
ax1.tick_params(direction='in')                      # Ticks en la parte interior de la
grafica
ax1.yaxis.set_ticks_position('both')              # Ticks en ambos ejes verticales
ax1.xaxis.set_ticks_position('both')              # Ticks en ambos ejes horizontales
ax1.plot(posx,altura,color="blue")                # Dibuja la altura en azul
ax1.set_title(fichero)                            # Nombre del fichero como titulo
ax1.set_ylabel("Height (m)")                      # Etiqueta vertical
ax1.set_xticklabels([])                           # No pone numeros en el eje horizontal
ax1.set_xlim(float(xlimi[0]),float(xlimi[1]))     # Limites de la grafica

# Grafica de la pendiente
ax2=fig.add_subplot(312)                          # Posicion de la grafica de pendientes
ax2.set_xlim(float(xlimi[0]),float(xlimi[1]))     # Limites de la grafica
ax2.tick_params(direction='in')                      # Ticks en la parte interior de la
grafica
ax2.yaxis.set_ticks_position('both')              # Ticks en ambos ejes verticales
ax2.xaxis.set_ticks_position('both')              # Ticks en ambos ejes horizontales
ax2.plot(pos1,der1,color="blue")                  # Dibuja la pendiente en azul
ax2.plot(pos1,smder1,'r--',lw=0.6)                # Dibuja la pendiente suavizada en rojo
con linea de trazos fina
#ax2.plot(posx,derv,color="red")                  # Dibuja la derivada leida del fichero
ax2.set_ylabel("Slope\n (First derivative)")      # Etiqueta vertical
ax2.set_xticklabels([])                           # No pone numeros en el eje horizontal
```

```
# Grafica de la segunda derivada
ax3=fig.add_subplot(313)                                # Posicion de la grafica de segundas
derivadas
ax3.set_xlim(float(xlimi[0]),float(xlimi[1]))     # Limites de la grafica
ax3.tick_params(direction='in')                          # Ticks en la parte interior de la
grafica
ax3.yaxis.set_ticks_position('both')              # Ticks en ambos ejes verticales
ax3.xaxis.set_ticks_position('both')              # Ticks en ambos ejes horizontales
ax3.plot(pos2,der2,color="blue")                  # Dibuja la segunda derivada en azul
ax3.plot(pos2,smder2s,'r--',lw=0.6)               # Dibuja la segunda derivada suavizada
a partir de datos suavizados en rojo
ax3.set_xlabel("Position (m)")                    # Etiqueta del eje horizontal
ax3.set_ylabel("Curvature\n (Second derivative)") # Etiqueta vertical

plt.show()
```

- **Cálculo de la Energía Liberada por los Terremotos**

  Cálculo de la Energía Liberada por los Terremotos

  - script: Earthquake_Energy.py
    Earthquake_Energy.py (sin ventanas)

  - Archivos de datos en Excel:

    frecuencias_mag.xls

```
############################################################
#                                                          #
#       Virtual Laboratory of Geomathematics in Python     #
#                                                          #
#             Analysis of Earthquake Energy                #
#                      (27.09.2017)                        #
#                                                          #
#         Complutense University of Madrid, Spain          #
#                                                          #
#   THIS SCRIPT IS PROVIDED BY THE AUTHORS "AS IS" AND     #
#   CAN BE USED BY ANYONE FOR THE PURPOSES OF EDUCATION    #
#   AND RESEARCH.                                          #
#                                                          #
############################################################
import sys

from matplotlib.widgets import Cursor
from matplotlib.gridspec import GridSpec
import matplotlib.pylab as plt
import numpy as np

import xlrd
import easygui as eg

# Ventana informacion
eg.msgbox("Within this script:\n 1) We selected a file with earthquakes frequency and
magnitude information\n 2) The frequency for each magnitude is plotted\n 3) The limits
for the magnitude intervale are selected using a cursor\n 4) The cumulative energy is
plotted",title='Problem 3', ok_button='Continue')


# Eleccion de fichero
seleccion = ["*.xls","*.xlsx","*.ods"]
fichero=eg.fileopenbox(msg="Please choose a file",title="Data
file",default="*",filetypes=seleccion)
print(fichero)

# Funcion de apertura y lectura de la columna n de un fichero excel
def read_excel(fichero,n):
        book = xlrd.open_workbook(fichero)
        sh = book.sheet_by_index(0)

        # Metemos la columna n en una lista
        coln=[]
        for rx in range(1,sh.nrows):
```

```
            coln.append(float(sh.cell_value(colx=n-1,rowx=rx)))
        return coln

# Si no se ha seleccionado el fichero se cierra el programa
if fichero == None:
     print("No file selected")
     sys.exit()

# Lee las magnitudes
num = eg.integerbox(msg='Column number for the minimum magnitudes (1-100):',
                    title='Minimum Magnitud',
                    default=1,
                    lowerbound=1,
                    upperbound=100)
if num == None:
       print("No column selected")
       sys.exit()
magmin=read_excel(fichero,num)

numm = eg.integerbox(msg='Column number for the maximum magnitudes (1-100):',
                    title='Maximum Magnitud',
                    default=2,
                    lowerbound=1,
                    upperbound=100)
if numm == None:
       print("No column selected")
       sys.exit()
magmax=read_excel(fichero,numm)

numf = eg.integerbox(msg='Column number for the frequency (1-100):',
                    title='Frequency',
                    default=3,
                    lowerbound=1,
                    upperbound=100)
if numm == None:
       print("No column selected")
       sys.exit()

frequ=read_excel(fichero,numf)

magmed=[]
for i in range(len(magmax)):
      magmed.append((magmax[i]+magmin[i])*0.5)

# Limites en el eje x de la grafica de magnitudes y numero de intervalos
etlim=['Mag min', 'Mag max', 'Frequencys']
xmin=min(magmin)
xmax=max(magmax)
fmax=max(frequ)+100

# Frecuencia acumulada
fac=[]
faca=0
for i in range(len(frequ)):
    fac.append(faca+frequ[i])
    faca=fac[i]

# Dibujamos graficas
fig=plt.figure('Magnitude')
fig.subplots_adjust(wspace=0.3)

# Histograma de las magnitudes
gs = GridSpec(2, 1, height_ratios=[10,1])
ax1=fig.add_subplot(gs[0])
ax1.tick_params(direction='in')
ax1.yaxis.set_ticks_position('both')
ax1.xaxis.set_ticks_position('both')
ax1.semilogy(magmed,frequ,'bs')
ax1.set_title("    "+fichero)
ax1.set_ylabel("Frequency")
ax1.set_xlabel("Magnitude")

# Mensaje de  seleccion
textstring='Selected in the above frequency plot the two limits for the energy calculation'
axm=fig.add_subplot(gs[1])
axm.text(0,0,textstring,fontsize=6,bbox={'facecolor':'blue', 'alpha':0.2, 'pad':1})
axm.axis('off')
```

```
# Seleccionamos rango de magnitudes para el ajuste
coords = []
cursor = Cursor(ax1,color='r',lw=1,horizOn=False,vertOn=True)

def onclick_x(event):
        if event.xdata != None and event.ydata != None:
              print(event.xdata)

        global coords
        coords.append(event.xdata)

        if len(coords)==2:
              fig.canvas.mpl_disconnect(cid)
              plt.close('Magnitude')
        return

cid = fig.canvas.mpl_connect('button_press_event', onclick_x)


plt.show()
plt.close()


# Limites para el ajuste
amin=min(coords)
amax=max(coords)
msum=[]
esum=[]
etot=[]
eant=0
j=0
for i in range(len(magmed)):
        if magmed[i]>amin and magmed[i]<amax:
              msum.append(magmed[i])
              esum.append(frequ[i]*(10**(5.24+1.44*magmed[i])))
              etot.append(eant+esum[j])
              eant=etot[j]
              j=j+1

# Dibuja los datos y el ajuste
fig2=plt.figure('Energy')
ax2=fig2.add_subplot(111)
ax2.tick_params(direction='in')
ax2.yaxis.set_ticks_position('both')
ax2.xaxis.set_ticks_position('both')
# Datos
ax2.semilogy(msum,esum,'bs',label='Energy (mag)')
# Total
ax2.semilogy(msum,etot,'r-',label='Total Energy (<mag)')
ax2.legend(loc=2)
ax2.set_ylabel("Energy (J)")
ax2.set_xlabel("Magnitude")
ax2.set_title(fichero)

plt.show()
```

- **Introducción al Cálculo Integral: Cálculo del Volumen del Sólido de Revolución**

A continuación, dos programas en Python ilustran el uso del cálculo simbólico, calculándose el volumen del Monte Fuji:

```
#############################################################
#                                                           #
#       Virtual Laboratory of Geomathematics in Python      #
#                                                           #
#             Calculating the volume of Mount Fuji          #
#                      (30.09.2017)                         #
#                                                           #
#         Complutense University of Madrid, Spain           #
#                                                           #
#   THIS SCRIPT IS PROVIDED BY THE AUTHORS "AS IS" AND      #
#   CAN BE USED BY ANYONE FOR THE PURPOSES OF EDUCATION     #
#   AND RESEARCH.                                           #
#                                                           #
#############################################################
from sympy import integrate, Symbol, pi, sqrt
# Definicion de variables simbolicas
z = Symbol('z')
# Modelo matematico
f=pi*((400*z)/3 - (800*sqrt(z))/sqrt(3)+400)
# Calculo de la integral
v=integrate(f,(z,0,3))
print(v.evalf())
```

```
#############################################################
#                                                           #
#       Virtual Laboratory of Geomathematics in Python      #
#                                                           #
#             Calculating the volume of Mount Fuji          #
#                      (30.09.2017)                         #
#                                                           #
#         Complutense University of Madrid, Spain           #
#                                                           #
#   THIS SCRIPT IS PROVIDED BY THE AUTHORS "AS IS" AND      #
#   CAN BE USED BY ANYONE FOR THE PURPOSES OF EDUCATION     #
#   AND RESEARCH.                                           #
#                                                           #
#############################################################
from sympy import *
# Definicion de variables simbolicas
y,z = symbols('y z')
# Modelo matematico
y=pi*((400*z)/3 - (800*sqrt(z))/sqrt(3)+400)
# Calculo de la integral
c=integrate(y,(z,0,3))
print(c.evalf())
```

Cálculo del Volumen de los cerros El Colmillo y La Muela (Guadalajara, España)

- script: Volumen_monte.py

  Volumen_monte.py (sin ventanas)

- Archivos de datos en Excel:

  perfil_colmillo_mod.xls
  perfil_muela_mod.xls

```
#############################################################
#                                                           #
#       Virtual Laboratory of Geomathematics in Python      #
#                                                           #
#             Calculating the volume of a hill              #
#                      (30.09.2017)                         #
#                                                           #
#         Complutense University of Madrid, Spain           #
#                                                           #
#   THIS SCRIPT IS PROVIDED BY THE AUTHORS "AS IS" AND      #
#   CAN BE USED BY ANYONE FOR THE PURPOSES OF EDUCATION     #
#   AND RESEARCH.                                           #
#                                                           #
#############################################################
import sys

from pylab import *

import xlrd
import easygui as eg
import numpy as np
from mpl_toolkits.mplot3d import Axes3D

import matplotlib.pyplot as plt
from matplotlib import animation

# Ventana informacion
eg.msgbox("In this script:\n 1) We selected a file with a height profile\n 2) The height
profile is plotted\n 3) The solid obtained by rotation of the height profile is also
plotted",title='Problem 4', ok_button='Continue')

# Eleccion de fichero
seleccion = ["*.xls","*.xlsx","*.ods"]
fichero=eg.fileopenbox(msg="Please          choose          a          file",title="Data
file",default="*",filetypes=seleccion)
print(fichero)

# Funcion de apertura y lectura de la columna n de un fichero excel
def read_excel(fichero,n):
        book = xlrd.open_workbook(fichero)
        sh = book.sheet_by_index(0)

        # Metemos la columna n en una lista
        coln=[]
        for rx in range(1,sh.nrows):
               coln.append(float(sh.cell_value(colx=n-1,rowx=rx)))
        return coln

# Si no se ha seleccionado el fichero se cierra el programa
if fichero == None:
     print("No file selected")
     sys.exit()

global posx,altura
# Lee las posiciones en x
numx = eg.integerbox(msg='Column number for the position (1-100):',
                     title='Position',
                     default=4,
                     lowerbound=1,
                     upperbound=100,
                     image=None)
if numx == None:
       print("No column selected")
       sys.exit()

posx=read_excel(fichero,numx)
posxmax=max(posx)

for i in range(len(posx)):
       posx[i]=posxmax-posx[i]

# Lee las posiciones en y
numy = eg.integerbox(msg='Column number for the height (1-100):',
                     title='Height',
                     default=3,
                     lowerbound=1,
                     upperbound=100,
```

```
                    image=None)
if numy == None:
       print("No column selected")
       sys.exit()
altura=read_excel(fichero,numy)


def alt3d(anmax):

   # Precision en los angulos
   n_angles=anmax
   n_radii=len(posx)
   angles=np.linspace(0,anmax*np.pi/180, n_angles, endpoint=True)
   angles= np.repeat(angles[...,np.newaxis], n_radii,axis=1)
   altura3d=[]
   x=(posx*np.cos(angles)).flatten()
   y=(posx*np.sin(angles)).flatten()
   z=np.tile(altura,n_angles)
   z=z.flatten()
   return x,y,z

def alt3dinter(anmax):

   # Precision en los angulos
   n_angles=3
   n_radii=len(posx)
   angles=np.linspace((anmax-5)*np.pi/180,anmax*np.pi/180, n_angles, endpoint=True)
   angles= np.repeat(angles[...,np.newaxis], n_radii,axis=1)
   altura3d=[]
   x=(posx*np.cos(angles)).flatten()
   y=(posx*np.sin(angles)).flatten()
   z=np.tile(altura,n_angles)
   z=z.flatten()
   return x,y,z

# Limites en el eje x de la grafica
minx=int(min(posx))
maxx=int(max(posx))

global altmi,altma
# Limites en altura
altmi=int(min(altura)-50)
altma=int(max(altura)+50)

# Superficie
x,y,altura3d = alt3d(360)

# GRAFICAS
fig1 = plt.figure('Profiles',figsize=plt.figaspect(2.))
fig2 = plt.figure('Animation')

# Grafica de la altura
ax1=fig1.add_subplot(2,1,1)
ax1.tick_params(direction='in')                         # Ticks en la parte interior de la
grafica
ax1.yaxis.set_ticks_position('both')              # Ticks en ambos ejes verticales
ax1.xaxis.set_ticks_position('both')              # Ticks en ambos ejes horizontales
ax1.plot(posx,altura,color="blue")                # Dibuja la altura en azul
ax1.set_title(fichero)                            # Nombre del fichero como titulo
ax1.set_ylabel("Height (m)")                      # Etiqueta vertical
ax1.set_xlabel("Radius (m)")                      # Etiqueta vertical
ax1.set_xlim(minx,maxx)                           # Limites de la grafica
ax1.set_ylim(altmi,altma)

ax2=fig1.add_subplot(2,1,2,projection='3d')
ax2.tick_params(direction='in')                         # Ticks en la parte interior de la
grafica
ax2.yaxis.set_ticks_position('both')              # Ticks en ambos ejes verticales
ax2.xaxis.set_ticks_position('both')              # Ticks en ambos ejes horizontales
ax2.set_zlabel("Height (m)")
ax2.set_xlabel("X (m)")
ax2.set_ylabel("Y (m)")
ax2.plot([0,0],[0,0],[altmi,altma],'-.',color='k')
ax2.set_zlim(altmi,altma)
ax2.plot_trisurf(x,y,altura3d,cmap=cm.coolwarm,linewidth=0.5)

# Grafica de la animacion
```

```
ax3=fig2.add_subplot(1,1,1,projection='3d')
ax3.set_zlim(altmi,altma)
ax3.set_xlim(-maxx,maxx)
ax3.set_ylim(-maxx,maxx)
ax3.set_xlabel('X (m)')
ax3.set_ylabel('Y (m)')
ax3.set_zlabel('Height (m)')

#surf,=ax3.plot([],[],[])

def init():
    surf = ax3.plot([0,0],[0,0],[altmi,altma],'-.',color='k')
    return surf,

# funcion de animacion, llamada secuencialmente
def animate(frame):
    xdata,ydata,zdata = alt3dinter(frame)
    surf = ax3.plot_trisurf(xdata,ydata,zdata,cmap=cm.coolwarm,linewidth=0.5)
    return surf,

nframes = np.arange(0,360,5)

# animador:
anim = animation.FuncAnimation(fig2,animate,init_func=init,frames=nframes,repeat=False)

plt.show()
```