

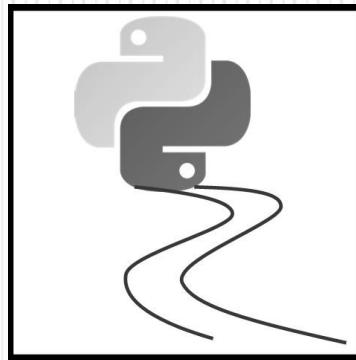
Taller corto de invierno
Winter short workshop

Lima, junio 2021

Hidrología en Python

Hydrology in Python

Pedro Rau, PhD
Hidrólogo



CENTRO DE
INVESTIGACIÓN
Y TECNOLOGÍA
DEL AGUA - UTEC



ESCUELA DE
POSGRADO
UTEC



✓ **Instructor:**

Pedro Rau

- PhD en Hidrología - Université de Toulouse - Francia
- Profesor full-time e investigador principal Universidad de Ingeniería y Tecnología UTEC - Dpto. Ingeniería Ambiental. Centro de Investigación y Tecnología del Agua CITA. Escuela de Posgrado.
- Profesor visitante UNI - Posgrado FIC.
- Hidrólogo investigador en redes internacionales: fr, ec, uk, us, cl
- Consultor en Hidrología y Recursos Hídricos

✓ **E.mail:** pedro.rau.ing@gmail.com

✓ **Sitio web:** <http://pedrorau.blogspot.com>

✓ **Repositorio del taller:** <https://github.com/hydrocodes/py.hidrologia>

✓ **Requisitos:** Conocimientos básicos en Hidrología y Estadística

Objetivos del curso

- Brindar las bases teóricas del estudio de las series de tiempo y aplicaciones hidrológicas.
- Punto de inicio para los interesados en el lenguaje Python aplicado al data science.

Temáticas

1. Introducción a las series de tiempo hidrológicas
2. Entorno Python y Spyder
3. Análisis exploratorio de datos
4. Aplicaciones hidrológicas

Evaluación

Clases teóricas + prácticas

- Asistencia y desarrollo de ejercicios

Metodología « Hands-on »

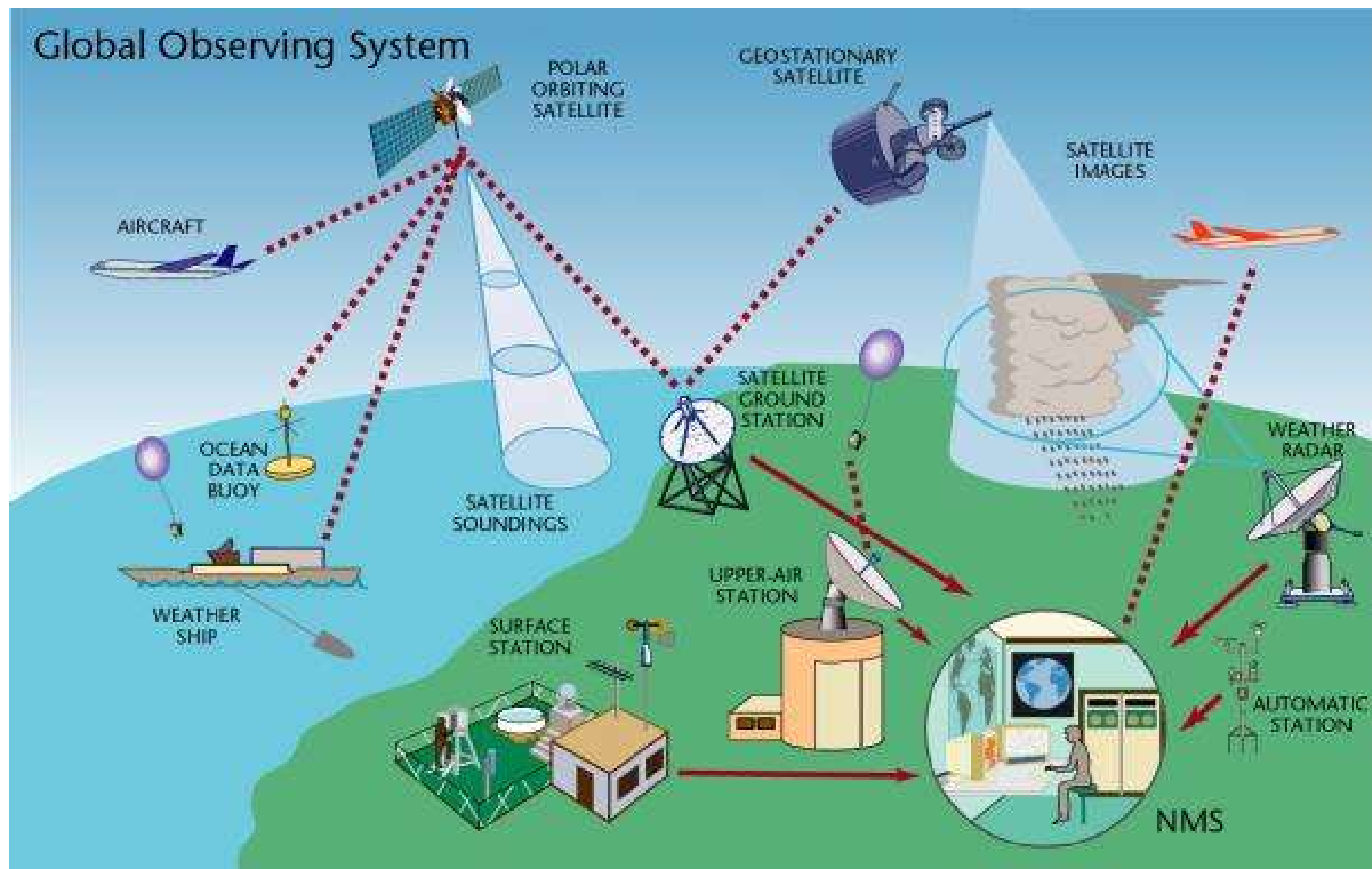
- Lecturas y papers base
- Planteamiento de ejercicios
- Creación/tipado/interpretación de códigos *.py con semi comentarios
- Resolución de bugs
- Finalización de códigos *.py con comentarios completos al detalle

Total: 8hr lectivas

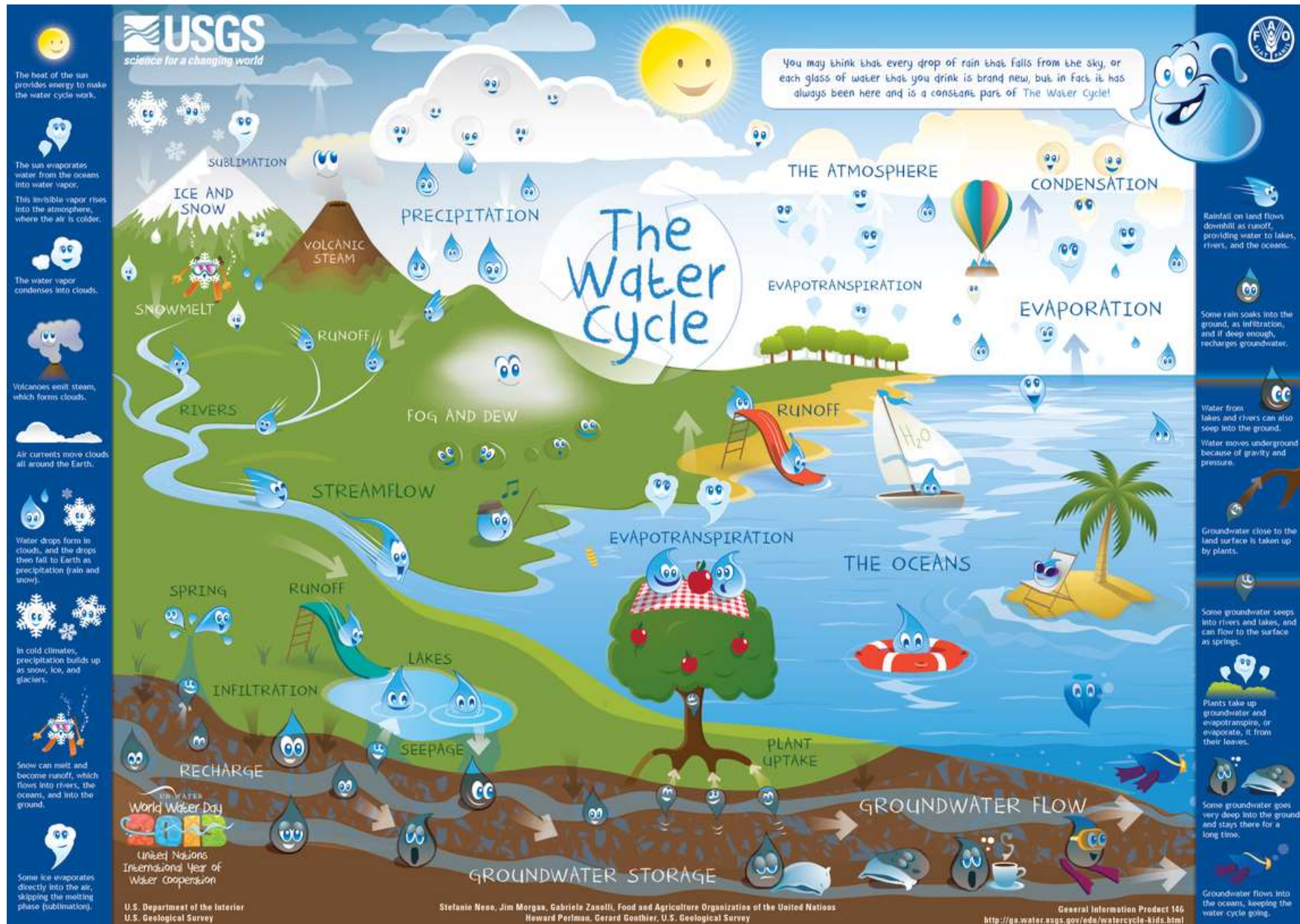
1. Introducción a las series de tiempo hidrológicas



1.1 Sistema de observación global

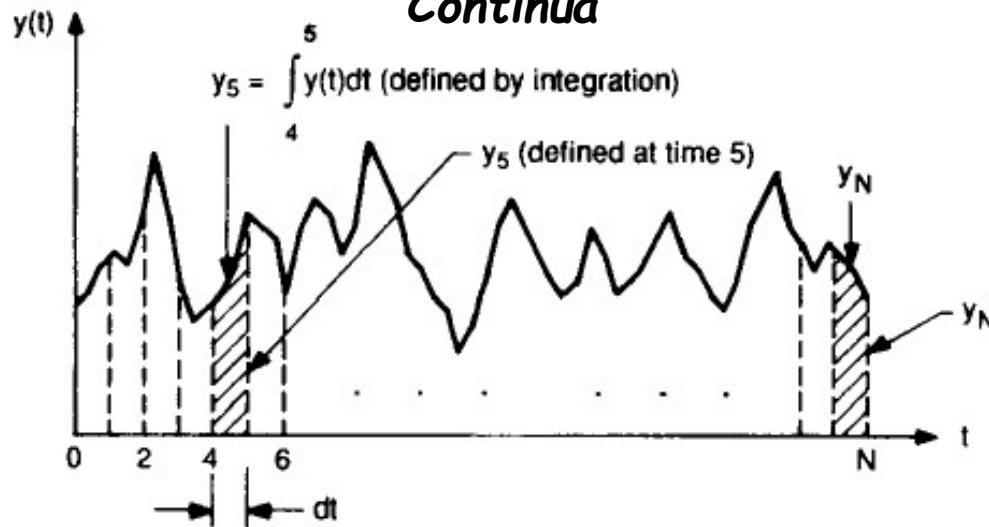


1.2 El ciclo hidrológico

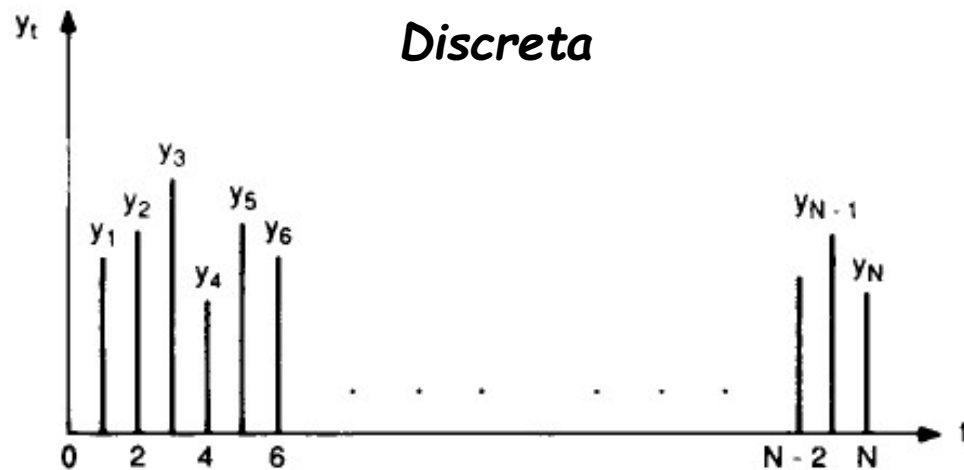


1.3 Tipos o categorías de series de tiempo

Continua



Discreta

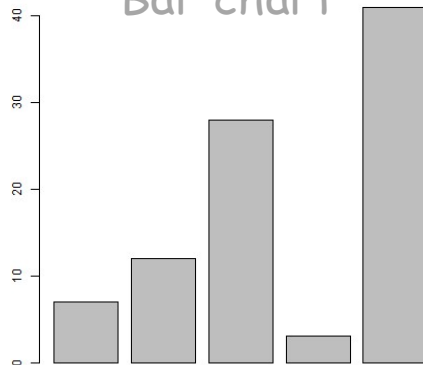


- ✓ Unicas
- ✓ Multiples
- ✓ Independientes, no correlacionadas; autocorrelacionados o dependientes
- ✓ Intermitentes
- ✓ De conteo
- ✓ Regulares o irregulares en espacios de tiempo
- ✓ Estacionarias (sin tendencia, ni salto ni ciclicidad / periodicidad) y no-estacionarias

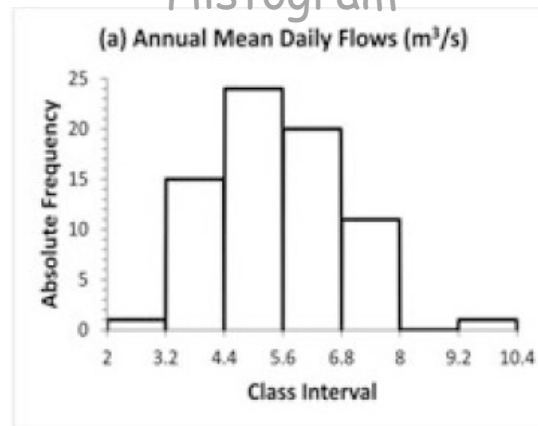
1.4 Análisis preliminar de datos hidrológicos

a. Representación gráfica

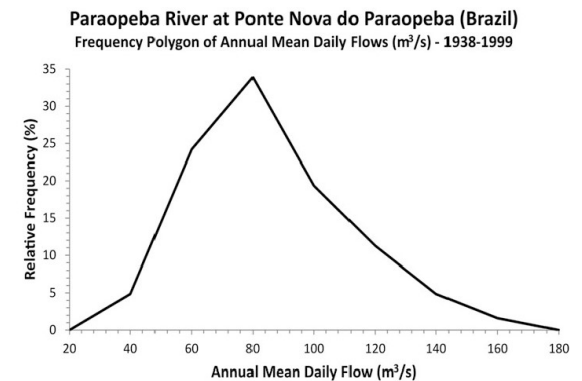
Diagrama de barras
Bar chart



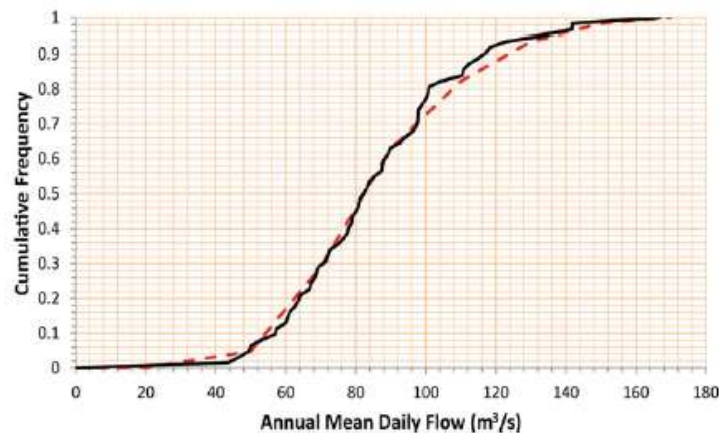
Histograma
Histogram



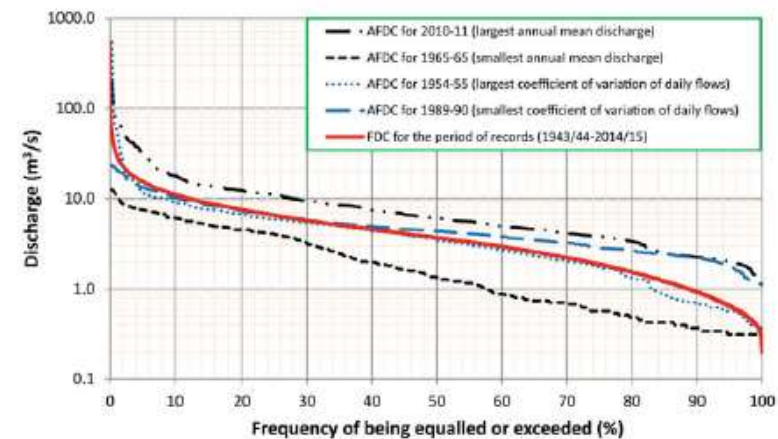
Poligonos de frecuencia
Frequency Polygon



Frecuencia relativa acumulada
Cumulative Relative Frequency Diagram



Curvas de duración
Duration curves



Naghetini (2017)

b. Estadísticos descriptivos

c. Métodos exploratorios

Diagrama de cajas Box plot

Parámetro de la población

Estadística de la muestra

1. Punto medio

Media aritmética

$$\mu = E(X) = \int_{-\infty}^{\infty} x f(x) dx$$

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$$

Mediana

x tal que $F(x) = 0.5$

Valor de la información en el 50o. percentil

Media geométrica

antilog $[E(\log x)]$

$$\left(\prod_{i=1}^n x_i \right)^{1/n}$$

2. Variabilidad

Varianza

$$\sigma^2 = E[(x - \mu)^2]$$

$$s^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$$

Desviación estándar

$$\sigma = \{E[(x - \mu)^2]\}^{1/2}$$

$$s = \left[\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2 \right]^{1/2}$$

Coefficiente de variación

$$CV = \frac{\sigma}{\mu}$$

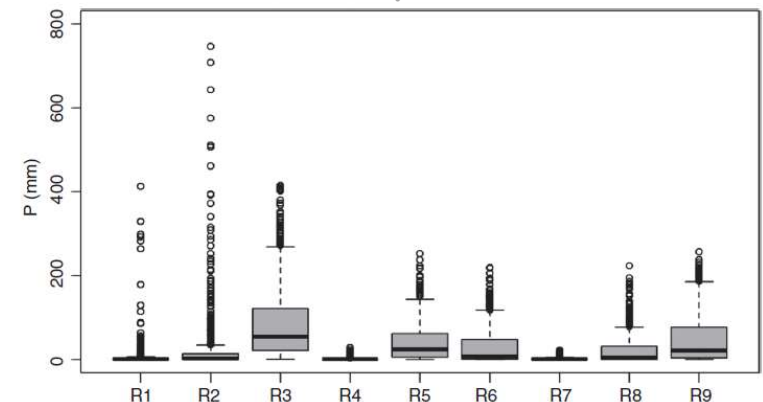
$$CV = \frac{s}{\bar{x}}$$

3. Simetría

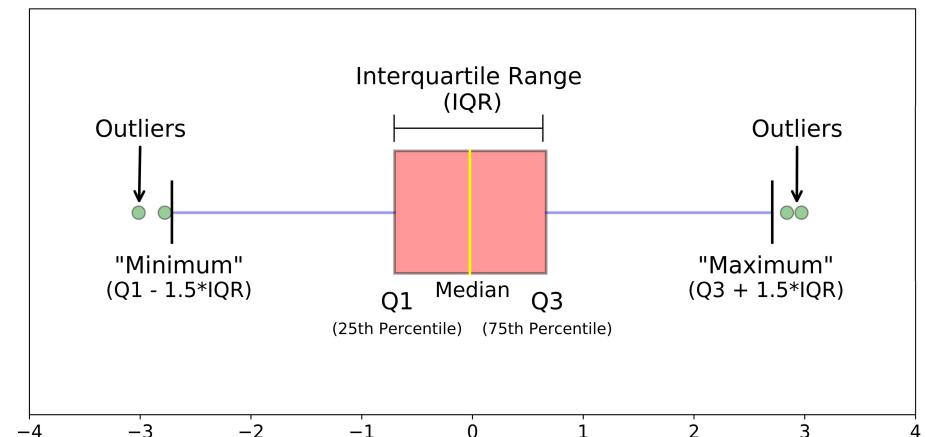
Coefficiente de asimetría (oblicuidad)

$$\gamma = \frac{E[(x - \mu)^3]}{\sigma^3}$$

$$C_s = \frac{n \sum_{i=1}^n (x_i - \bar{x})^3}{(n-1)(n-2)s^3}$$



Rau et al (2017)



Chow (1994)

d. Asociación de datos

Gráfico de dispersion
Scatter plot

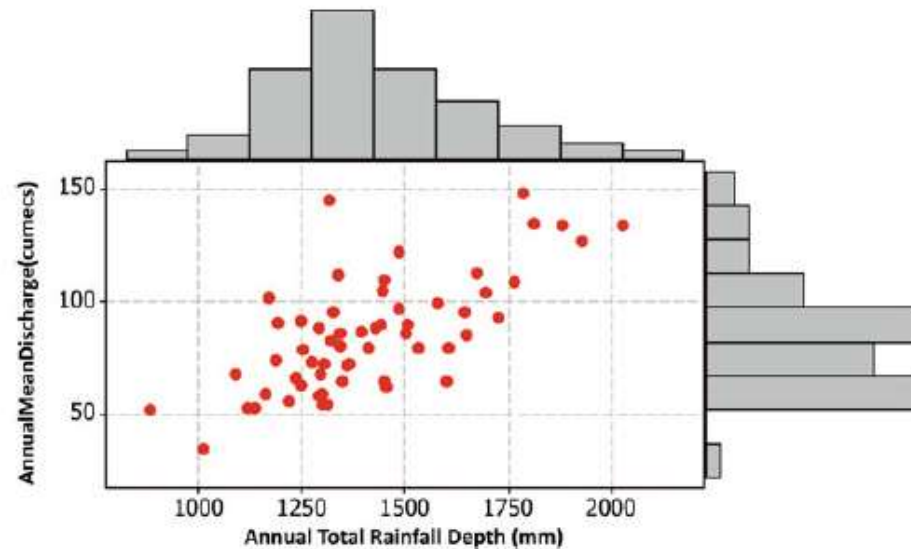
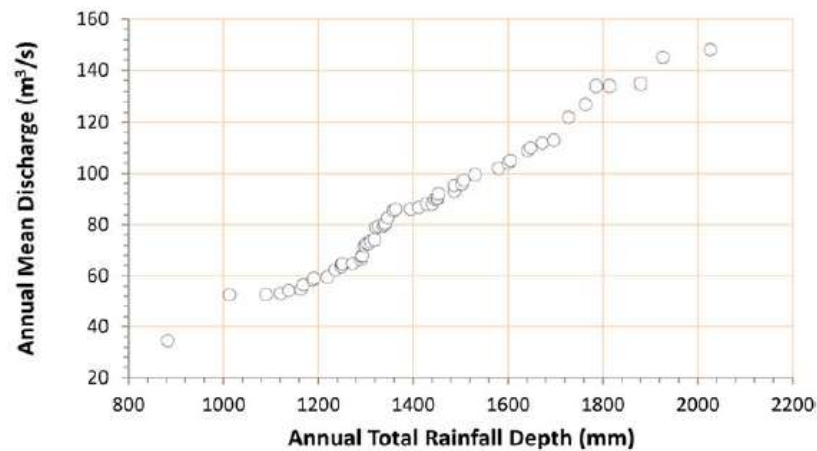


Diagrama Cuantil-Cuantil
Empirical Quantile-
Quantile Diagram
Q-Q Plot



Naghettini (2017)



python™

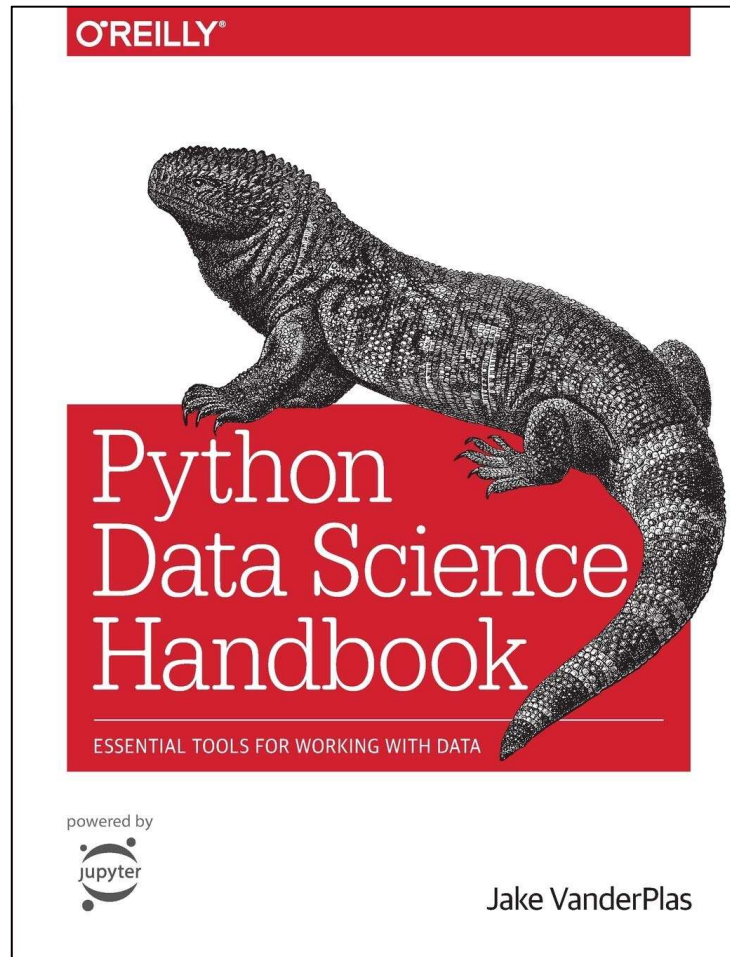


SPYDER

2. Entorno Python y Spyder



2.1 Ciencia de los Datos o Data Science

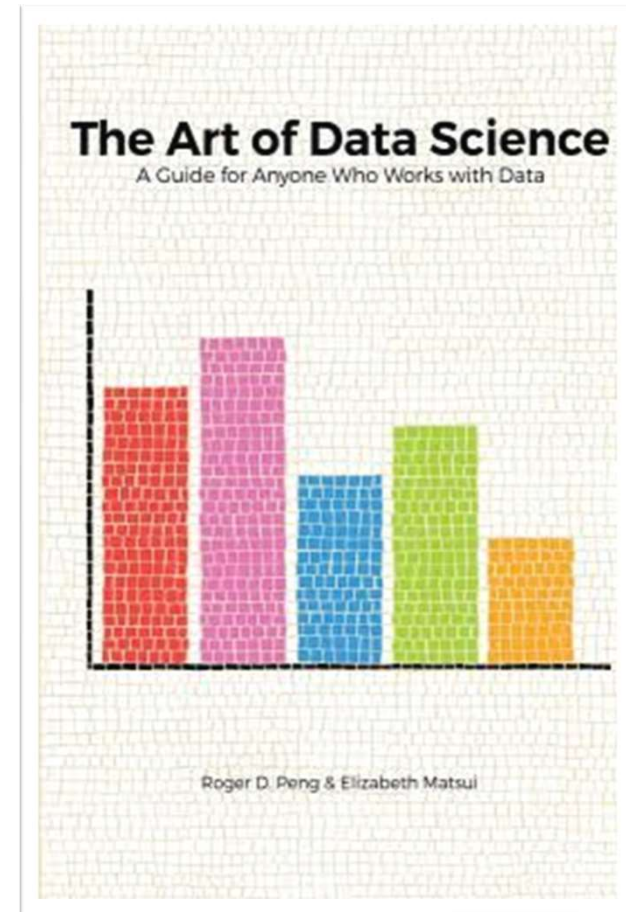


Fuente:

<https://jakevdp.github.io/PythonDataScienceHandbook/>

Repositorio:

<https://github.com/jakevdp/PythonDataScienceHandbook>



Fuente:

<https://bookdown.org/rdpeng/artofdatascience/>

Repositorio:

<https://github.com/waldronlab/The-Art-of-Data-Science/blob/master/README.md>

2.2 Python 3.9.4 for Windows (32/64 bit)

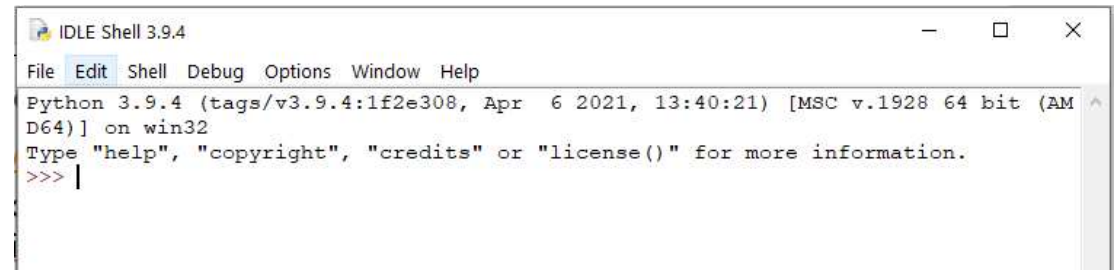
octubre 2020



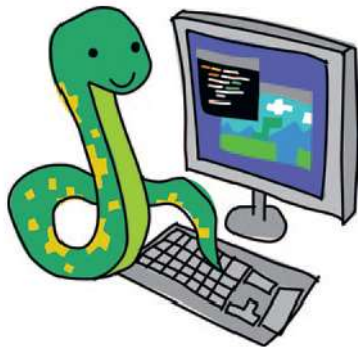
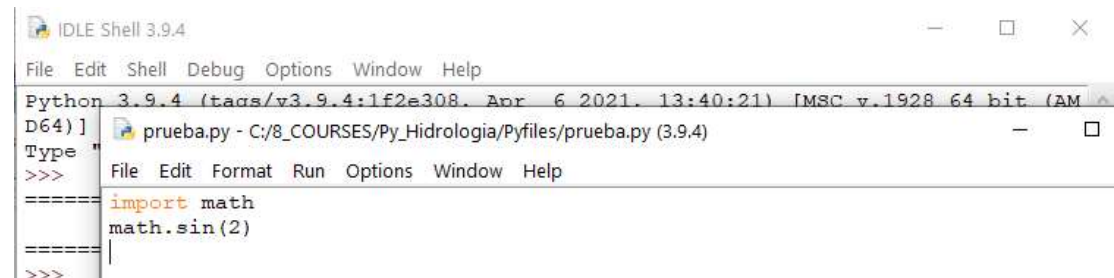
<https://www.python.org/>

- ✓ Lenguaje de programación « poderoso » y « rápido » (van Rossum, 1995).
- ✓ Lenguaje interpretado, multiparadigma y casi orientado a objetos.
- ✓ Fácil e intuitivo.

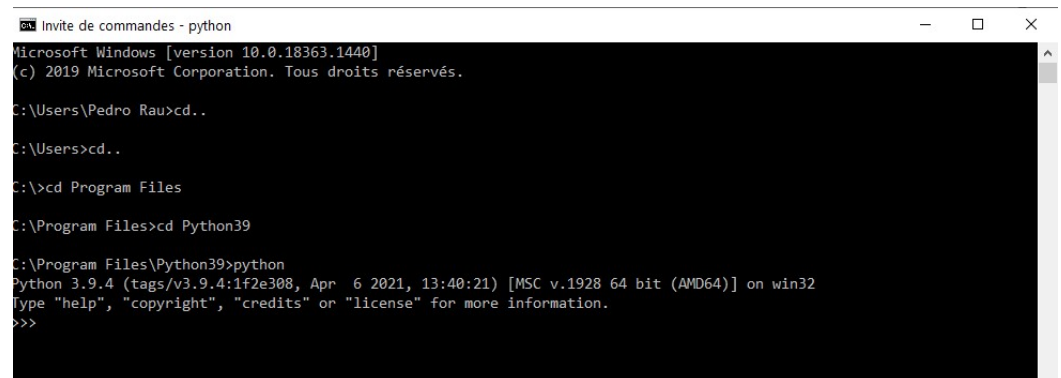
*El **IDLE** (Integrated Development and Learning Environment), es el intérprete y permite escribir en Python.*



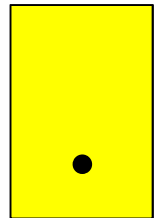
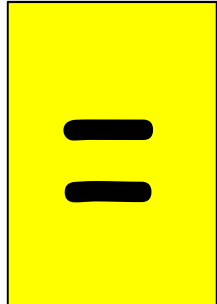
*File – New, para abrir el **editor** de IDLE.
También se puede usar el block de notas y guardar el archivo como *.py*



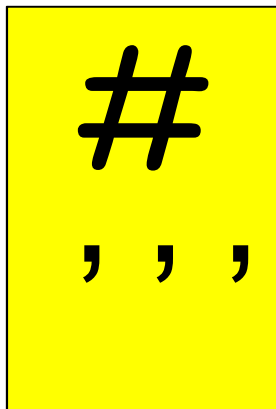
*También desde la **consola de comandos cmd**, usar **commandos UNIX “cd”** para **cambiar la ruta donde se encuentra instalado Python**.*



2.3 Tipos de objetos en Python



Type	Examples	Comments
None	None	# singleton null object
Boolean	True, False	
Integer	-1, 0, 1, sys.maxint	
Long	1L, 9787L	
Float	3.141592654 inf, float('inf') -inf nan, float('nan')	# infinity # neg infinity # not a number
Complex	2+8j	# note use of j
String	'this is a string', "also me" r'raw string', b'ASCII string' u'unicode string'	# use single or double quote
Tuple	empty = () (1, True, 'ML')	# empty tuple # immutable list or unalterable list
List	empty = [] [1, True, 'ML']	empty list # mutable list or alterable list
Set	empty = set() set(1, True, 'ML')	# empty set # mutable or alterable
dictionary	empty = {} {'1':'A', '2':'AA', True = 1, False = 0}	# mutable object or alterable object
File	f = open('filename', 'rb')	



Swamynathan (2017)

2.4 Operadores aritméticos

Operator	Description	Example
+	Addition	$x + y = 30$
-	Subtraction	$x - y = -10$
*	Multiplication	$x * y = 200$
/	Division	$y / x = 2$
%	Modulus	$y \% x = 0$
**	Exponent	$x ** b = 10$ to the power 20
//	Floor Division - Integer division rounded toward minus infinity	$9 // 2 = 4$ and $9.0 // 2.0 = 4.0$, $-11 // 3 = -4$, $-11.0 /$

2.5 Listas en Python

Description	Python Expression	Example	Results
Creating a list of items	[item1, item2, ...]	list = ['a', 'b', 'c', 'd']	['a', 'b', 'c', 'd']
Accessing items in list	list[index]	list = ['a', 'b', 'c', 'd'] list[2]	c
Length	len(list)	len([1, 2, 3])	3
Concatenation	list_1 + list_2	[1, 2, 3] + [4, 5, 6]	[1, 2, 3, 4, 5, 6]
Repetition	list * int	['Hello'] * 3	['Hello', 'Hello', 'Hello']
Membership	item in list	3 in [1, 2, 3]	TRUE
Iteration	for x in list: print x	for x in [1, 2, 3]: print x,	1 2 3
Count from the right	list[-index]	list = [1,2,3]; list[-2]	2
Slicing fetches sections	list[index:]	list = [1,2,3]; list[1:]	[2,3]
Comparing lists	cmp(list_1, list_2)	print cmp([1,2,3,4], [5,6,7]); print cmp([1,2,3], [5,6,7,8])	1 -1
Return max item	max(list)	max([1,2,3,4,5])	5
Return min item	min(list)	max([1,2,3,4,5])	1
Append object to list	list.append(obj)	[1,2,3,4].append(5)	[1,2,3,4,5]
Count item occurrence	list.count(obj)	[1,1,2,3,4].count(1)	2
Append content of sequence to list	list.extend(seq)	['a', 1].extend(['b', 2])	['a', 1, 'b', 2]
Return the first index position of item	list.index(obj)	['a', 'b', 'c', 1,2,3].index('c')	2
Insert object to list at a desired index	list.insert(index, obj)	['a', 'b', 'c', 1,2,3].insert(4, 'd')	['a', 'b', 'c', 'd', 1,2,3]
Remove and return last object from list	list.pop(obj=list[-1])	['a', 'b', 'c', 1,2,3].pop() ['a', 'b', 'c', 1,2,3].pop(2)	3 c
Remove object from list	list.remove(obj)	['a', 'b', 'c', 1,2,3].remove('c')	['a', 'b', 1,2,3]
Reverse objects of list in place	list.reverse()	['a', 'b', 'c', 1,2,3].reverse()	[3,2,1, 'c', 'b', 'a']
Sort objects of list	list.sort()	['a', 'b', 'c', 1,2,3].sort() ['a', 'b', 'c', 1,2,3].sort(reverse = True)	[1,2,3, 'a', 'b', 'c'] ['c', 'b', 'a', 3,2,1]

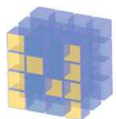
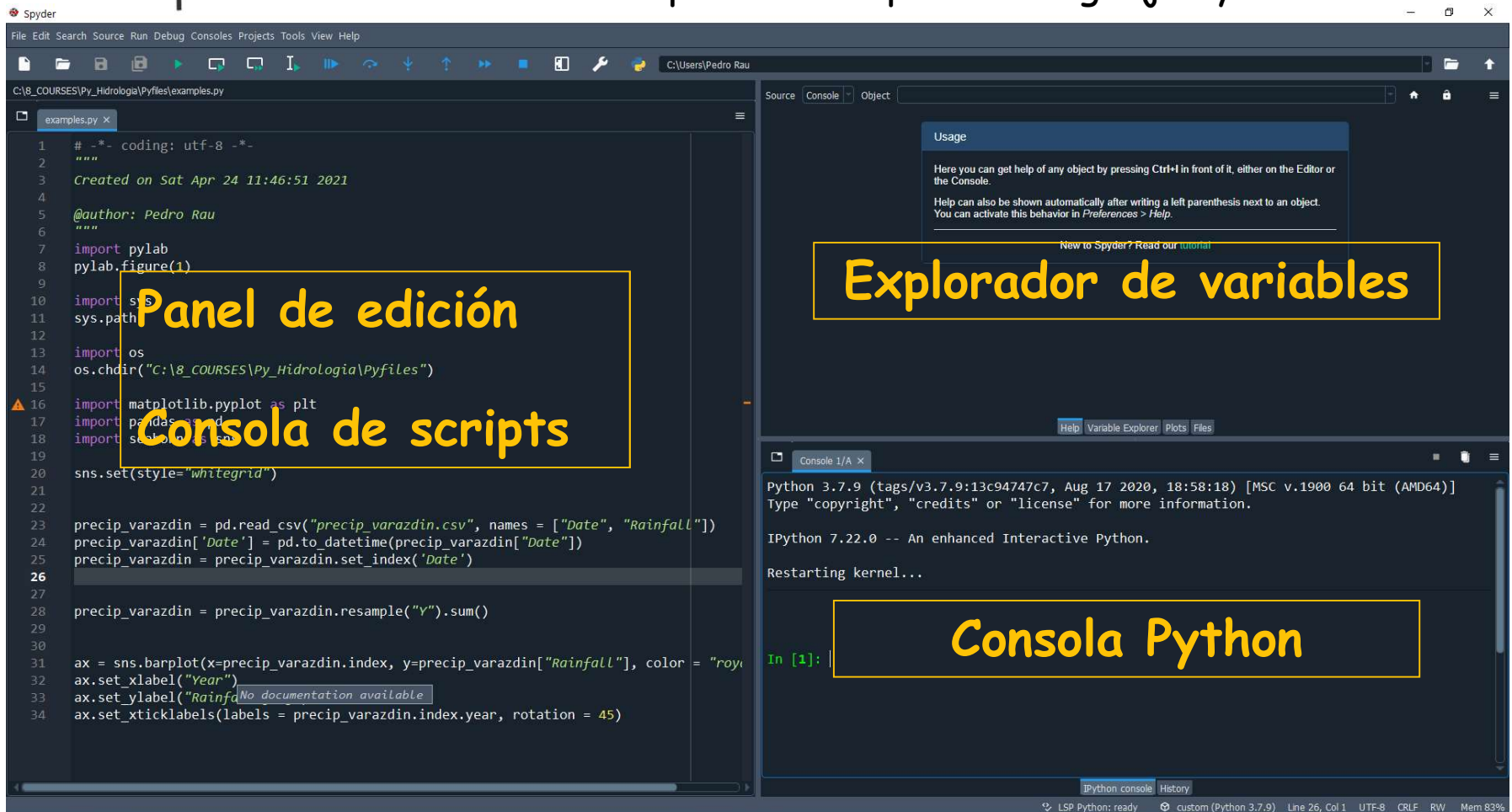
2.6 Spyder 5.0.1

<https://www.spyder-ide.org/>



SPYDER

Entorno de desarrollo integrado libre y multiplataforma para el lenguaje Python



NumPy

matplotlib



pandas

A. Comandos y códigos en el entorno Py

Ejercicio A.1

En la consola Python original, efectuar las siguientes operaciones:

- 1) $3^2 - 5 * 9 * (25 - 18)$
- 2) Asignar la operación $28*3/9$ a la variable *value*.
- 3) Visualizar la variable *value* creada.
- 4) Calcular el coseno de 30° sexagesimales y asignarle a la variable *x*.
- 5) Almacenar los siguientes valores de lluvia anual: 200, 210.2, 490, 100.5, 150.1, 190, 310 y 200.2 en la variable *rain*
- 6) Visualizar la variable *rain*.
- 7) Obtener el 5to elemento de *rain*.
- 8) Crear una serie consecutiva del 1 al 8 y asignar como variable *p*.
- 9) Visualizar *p*.
- 10) Crear la variable *q*, almacenando una secuencia desde 5 hasta 15 de 2 en 2.
- 11) Visualizar *q*.
- 12) Crear una serie consecutiva de letras desde "a" hasta "h".

Respuestas A.1

```
>>> 3**2-5*9*(25-18)
-306
>>> value = 28*3/9 #Asignamos una operación a la variable value
>>> value #Escribimos el objeto "value"
9.333333333333334
>>> import math #Se requiere llamar al modulo math
>>> math.cos(math.radians(30)) #Las funciones trabajan con radianes
>>> import numpy as np #Se requiere llamar al modulo numpy
>>> rain = np.array([200, 210.2, 490, 100.5, 150.1, 190, 310, 200.2]) #Uso de la función
array dentro de numpy denotado como np
>>> print(rain)
[200.  210.2 490.  100.5 150.1 190.  310.  200.2]
>>> rain[4] #Python inicia el conteo en 0; 4 representa a la 5ta posicon
>>> for p in range(1,9): #No olvidar el : al final de la sentencia for
    print(p)
1
2
3
4
5
6
7
8
>>> for q in range(5,16,2):
    print(q)
>>> r = []
>>> for i in range(ord('a'), ord('h')+1):
    r.append(chr(i))
>>> print(r)
```


Ejercicio A.2

En la consola de scripts Python original, efectuar las siguientes operaciones:

- 1) Crea la variable rain con los datos anteriores.
- 2) Obtener la media, mediana, desviación estándar, varianza de la variable rain.
- 3) Visualizar un resumen de los estadísticos notables.
- 4) Plotear un diagrama de cajas con los valores del objeto rain.
Interpretar.
- 5) Añadir a lo anterior con una etiqueta en el eje vertical indicando: P (mm/y).

Respuestas A.2

```
import statistics as stat
```

```
rain = [200, 210.2, 490, 100.5, 150.1, 190, 310, 200.2]
```

```
stat.mean(rain)
```

```
stat.median(rain)
```

```
stat.stdev(rain)
```

```
stat.variance(rain)
```

```
import pandas as pd
```

```
r = pd.Series(rain)
```

```
r.describe()
```

```
import matplotlib.pyplot as plt
```

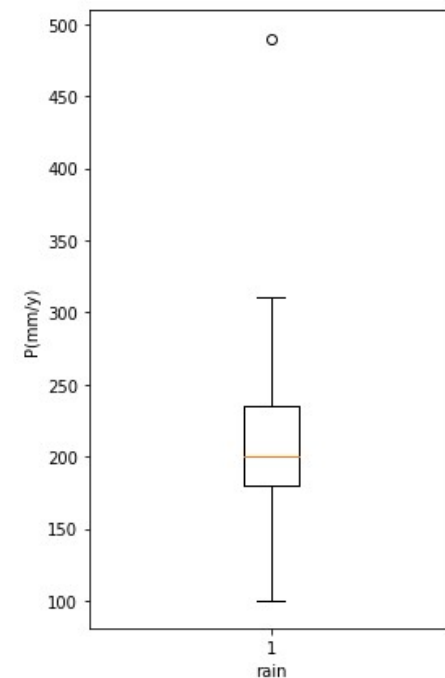
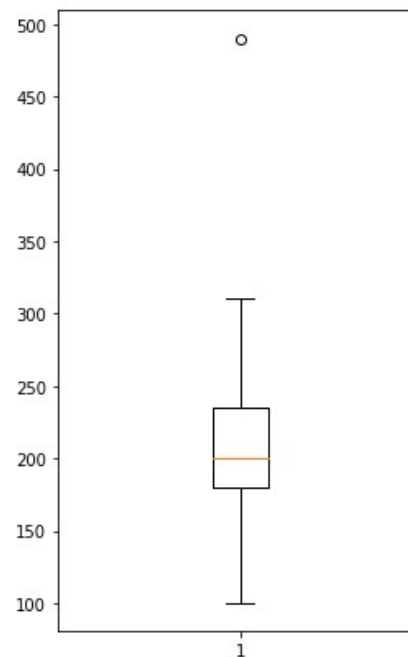
```
fig = plt.figure(figsize=(4, 7))
```

```
plt.boxplot(rain)
```

```
plt.xlabel("rain")
```

```
plt.ylabel("P(mm/y)")
```

```
plt.show()
```



Ejercicio A.3

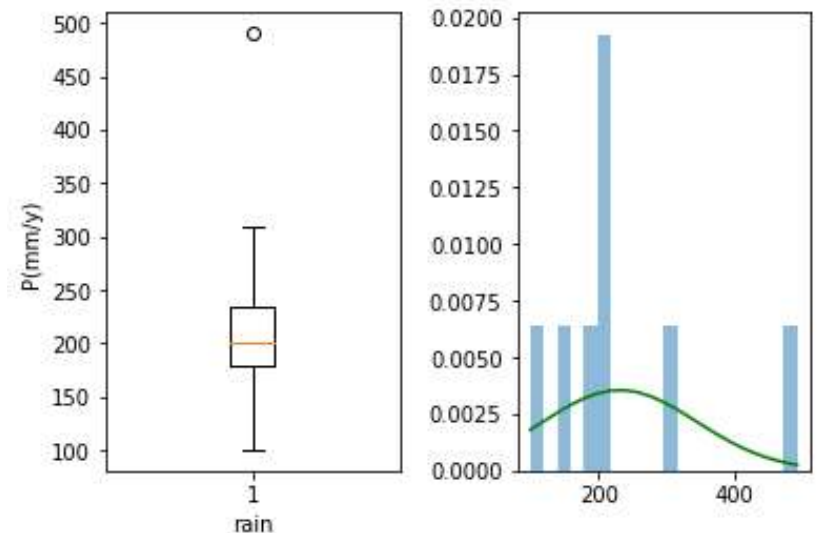
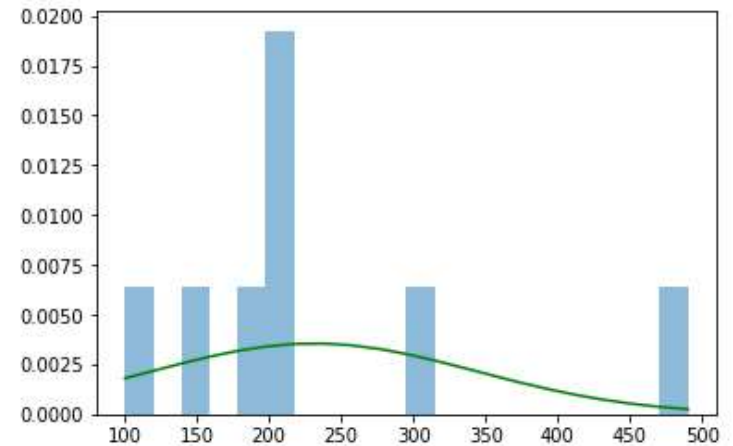
En la consola de scripts de Spyder, efectuar las siguientes operaciones:

- 1) Plotear un histograma de frecuencias para rain, empleando la frecuencia.
- 2) Plotear un histograma de frecuencias para rain, empleando la densidad.
- 3) Agregar al histograma una curva de distribución normal en color verde con la media y desviación estimados anteriormente.
- 4) Agregar en un solo gráfico (1filas x 2 columnas), el grafico de cajas anterior y el histograma.
- 5) Crear la variable elev con los sgtes datos de elevaciones: 3200, 3500, 4500, 3050, 3100, 2800, 3800, 3500 msnm.
- 6) Calcular la covarianza entre la lluvia y la elevación.
- 7) Calcular el coeficiente de correlación entre la lluvia y elevación.
- 8) Calcular la ecuación de regresión lineal entre la lluvia y elevación.
- 9) Plotear un gráfico de dispersión entre la lluvia y la elevación.
- 10) Agregar la línea de tendencia al gráfico anterior.
- 11) Guardar como un archivo *.py

Respuestas A.3

```
import matplotlib.pyplot as plt
import scipy.stats
rain = [200, 210.2, 490, 100.5, 150.1, 190, 310, 200.2]
_, bins, _ = plt.hist(rain, 20, density=1, alpha=0.5)
mu, sigma = scipy.stats.norm.fit(rain)
best_fit_line = scipy.stats.norm.pdf(bins, mu, sigma)
plt.plot(bins, best_fit_line, color='g')
```

```
fig = plt.figure()
fig.subplots_adjust(wspace=0.4)
plt.subplot(121)
plt.boxplot(rain)
plt.xlabel("rain")
plt.ylabel("P(mm/y)")
plt.subplot(122)
_, bins, _ = plt.hist(rain, 20, density=1, alpha=0.5)
plt.plot(bins, best_fit_line, color='g')
plt.show()
```



```
elev = [3200, 3500, 4500, 3050, 3100, 2800, 3800, 3500]
```

```
# covarianza y correlacion
```

```
import numpy
```

```
Covarianza = numpy.cov(rain, elev)[0][1] #la opcion bias=True permite el analisis con N elementos
```

```
print(Covarianza)
```

```
Correlacion = numpy.corrcoef(rain, elev)[0][1]
```

```
print(Correlacion)
```

```
# modelo de regression lineal
```

```
import matplotlib.pyplot as plt
```

```
from scipy import stats
```

```
slope, intercept, r, p, std_err = stats.linregress(rain, elev)
```

```
def myfunc(rain):
```

```
    return slope * rain + intercept
```

```
mymodel = list(map(myfunc, rain))
```

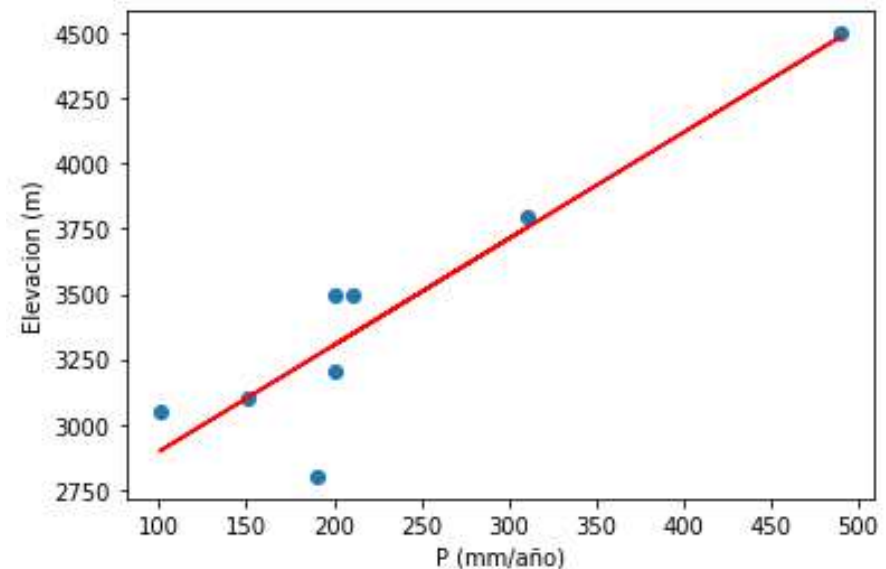
```
plt.scatter(rain, elev)
```

```
plt.plot(rain, mymodel, color='r')
```

```
plt.xlabel("P (mm/año)")
```

```
plt.ylabel("Elevacion (m)")
```

```
plt.show()
```



Ejercicio A.4

En la consola de scripts de Spyder :

- 1) Crear una función que calcule la evaporación en mm/día por el método del balance energía con los sgtes datos de ingreso: radiación neta en W/m^2 y temperatura ambiente en $^{\circ}\text{C}$.
- 2) Calcular la evaporación para una radiación de 80 W/m^2 y una temperatura de 15°C .
- 3) Calcular la evaporación para una radiación de 120 W/m^2 y una temperatura de 21°C .

Respuestas A.4

$$E_r = \frac{R_n}{l_v \rho_w}$$

Rn: Radiacion neta (W/m2)

Iv: calor latente de vaporizacion (KJ/Kg)

$$I_v = 2501 - 2.37 * T$$

pw: Densidad del agua = 997 Kg/m3

m/s a mm/dia, se emplea el factor 1000*86400



```
def evap_r(Rn, T):
```

```
    result = Rn*86400/(997*(2501-2.37*T))
```

```
    print(result)
```

```
# La funcion considera las conversiones del calor latente de vaporizacion
```

```
evap_r(80,15)
```

```
evap_r(120,21)
```

```
evap_r(80,15)
```

```
2.81198093459026
```

```
evap_r(120,21)
```

```
4.242440567706145
```

3. Análisis exploratorio de datos





Pandas es un paquete o librería libre para Python que permite trabajar bases de datos. Panda permite trabajar con dos nuevas estructuras de datos: **Series** y **DataFrames**

Lectura de archivos

Desde un archivo csv:

```
df=pd.read_csv('Data/filename.csv')
```

Desde un archivo txt:

```
df=pd.read_csv('Data/filename.txt', sep='\t')
```

Desde un archivo MS Excel y una hoja específica:

```
df=pd.read_excel('Data/filename.xlsx', 'Sheet2')
```

Algunas operaciones básicas con Pandas

Description	Syntax
Convert string to date series	pd.to_datetime(pd.Series(['2017-04-01','2017-04-02','2017-04-03']))
Rename a specific column name	df.rename(columns={'old_columnname':'new_columnname'}, inplace=True)
Rename all column names of DataFrame	df.columns = ['col1_new_name','col2_new_name'....]
Flag duplicates	df.duplicated()
Drop duplicates	df = df.drop_duplicates()
Drop duplicates in specific column	df.drop_duplicates(['column_name'])
Drop duplicates in specific column, but retain the first or last observation in duplicate set	df.drop_duplicates(['column_name'], keep = 'first') # change to last for retaining last obs of duplicate
Creating new column from existing column	df['new_column_name'] = df['existing_column_name'] + 5
Creating new column from elements of two columns	df['new_column_name'] = df['existing_column1'] + '_' + df['existing_column2']
Adding a list or a new column to DataFrame	df['new_column_name'] = pd.Series(mylist)
Drop missing rows and columns having missing values	df.dropna()
Replaces all missing values with 0 (or you can use any int or str)	df.fillna(value=0)

B. Explorando datos de precipitación diaria de varias estaciones almacenados en un archivo csv

Uso de las librerías: pandas, numpy, matplotlib

- Librerías instaladas por defecto (verificar versión actual)
- Descargar archivo "p_diarias.csv"
https://github.com/hydrocodes/py.hidrologia/blob/main/p_diarias.csv
- Revisión de archivo csv (separados por comas con formato de fecha %Y-%m-%d)

Ejercicio B

En la consola de scripts de Spyder, crear un código que realice lo siguiente:

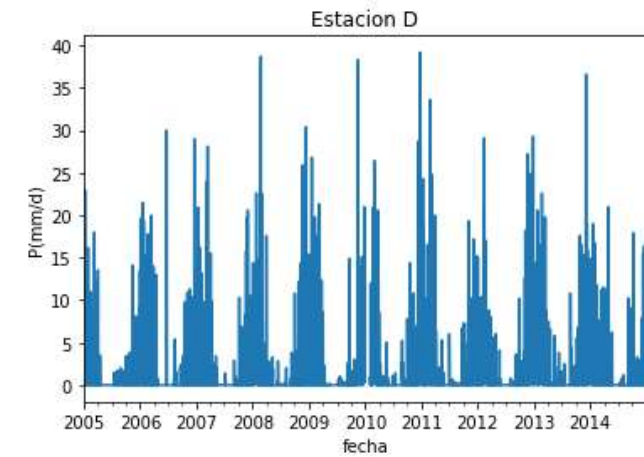
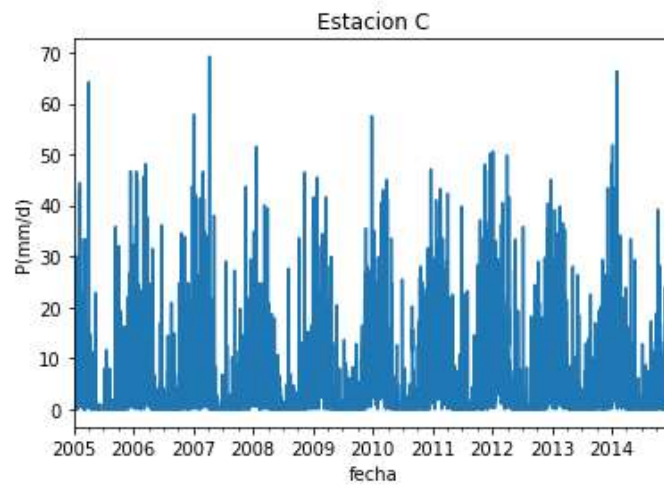
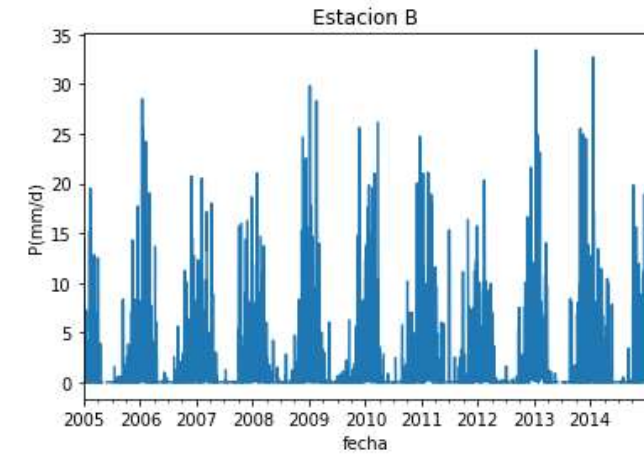
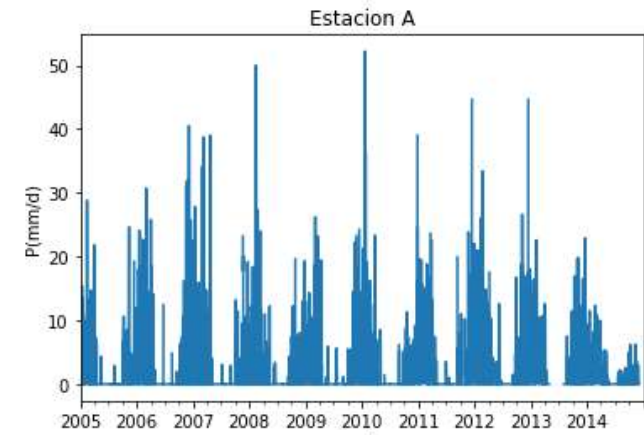
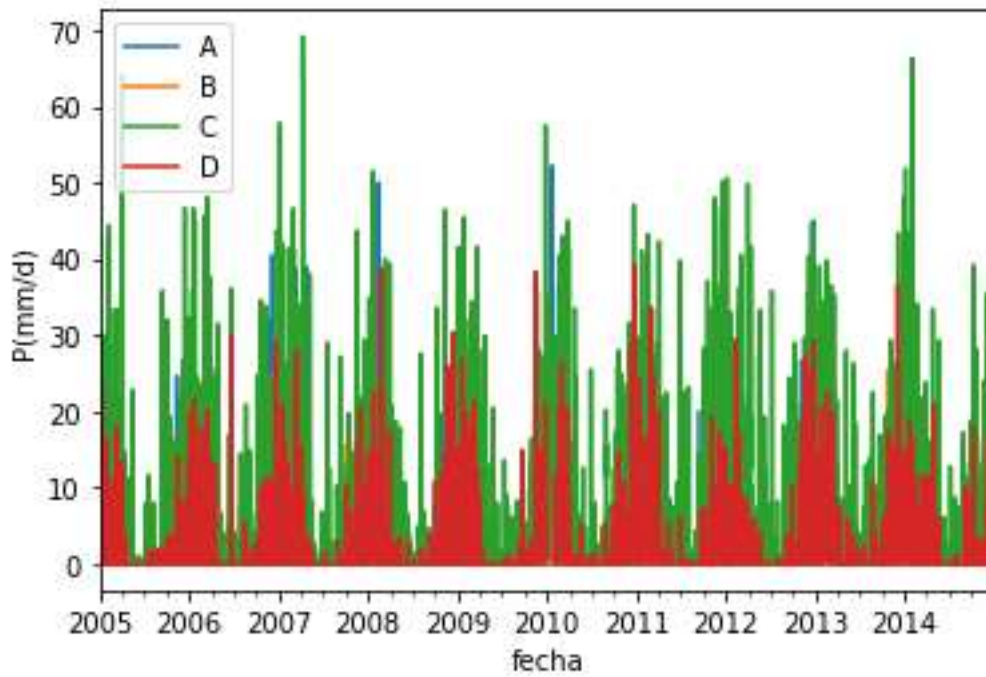
- 1) Leer el archivo de precipitaciones diarias "p_diarias.csv" almacenados en formato matriz con 4 estaciones (identificadores A, B, C, D) desde el 2005-01-01 al 2014-12-31.
- 2) Plotear las series de tiempo de las estaciones superpuestas.
- 3) Plotear la serie de tiempo para cada estación.
- 4) Plotear las series de tiempo de las estaciones por separado en el mismo plot.
- 5) Del anterior, plotear con una misma escala vertical.

Respuestas B

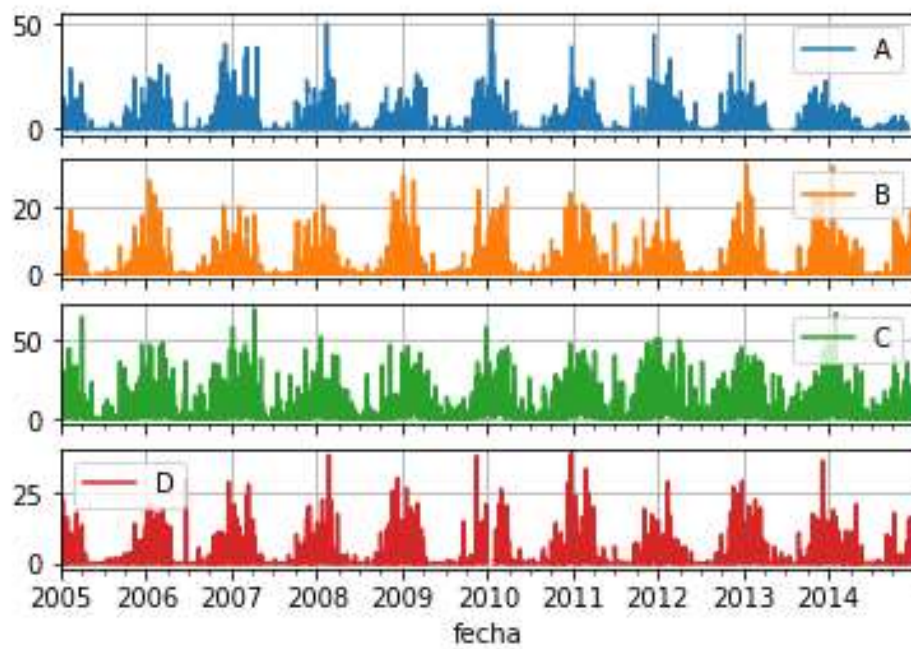
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
data = pd.read_csv('p_diarias.csv', parse_dates = ['fecha'])
print(data.head())
data.fecha = pd.to_datetime(data.fecha)
data.set_index('fecha', inplace=True)
data.plot()
plt.ylabel("P(mm/d)")

data['A'].plot()
plt.title("Estacion A")
plt.ylabel("P(mm/d)")
data['B'].plot()
plt.title("Estacion B")
plt.ylabel("P(mm/d)")
data['C'].plot()
plt.title("Estacion C")
plt.ylabel("P(mm/d)")
data['D'].plot()
plt.title("Estacion D")
plt.ylabel("P(mm/d)")

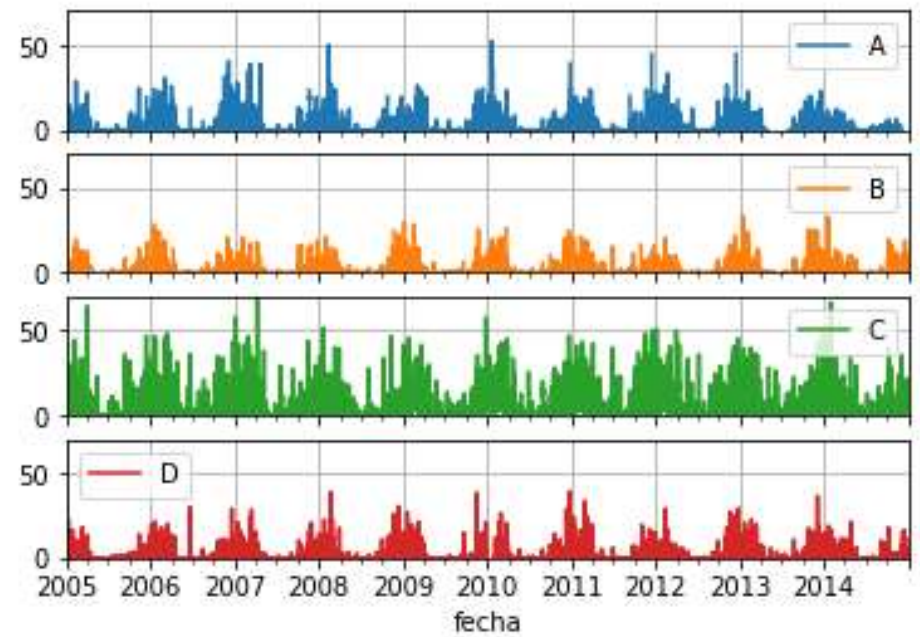
data.plot(kind='line', subplots=True, grid=True, title="Precipitaciones diarias")
data.max()
data.plot(kind='line', subplots=True, grid=True, title="Precipitaciones diarias", ylim=(0,70))
```



Precipitaciones diarias



Precipitaciones diarias



C. Convirtiendo datos diarios a mensuales y anuales

Ejercicio C

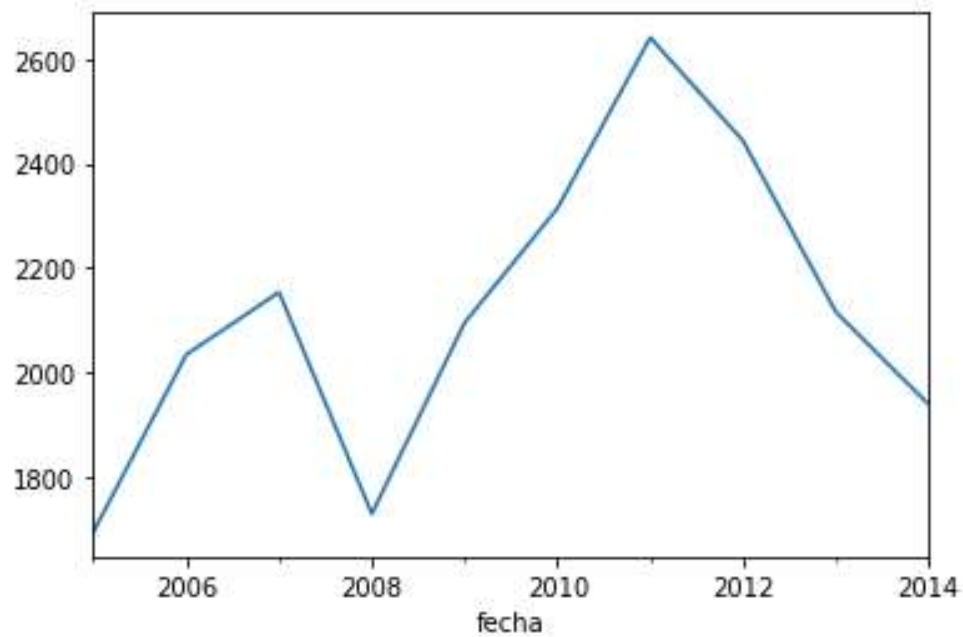
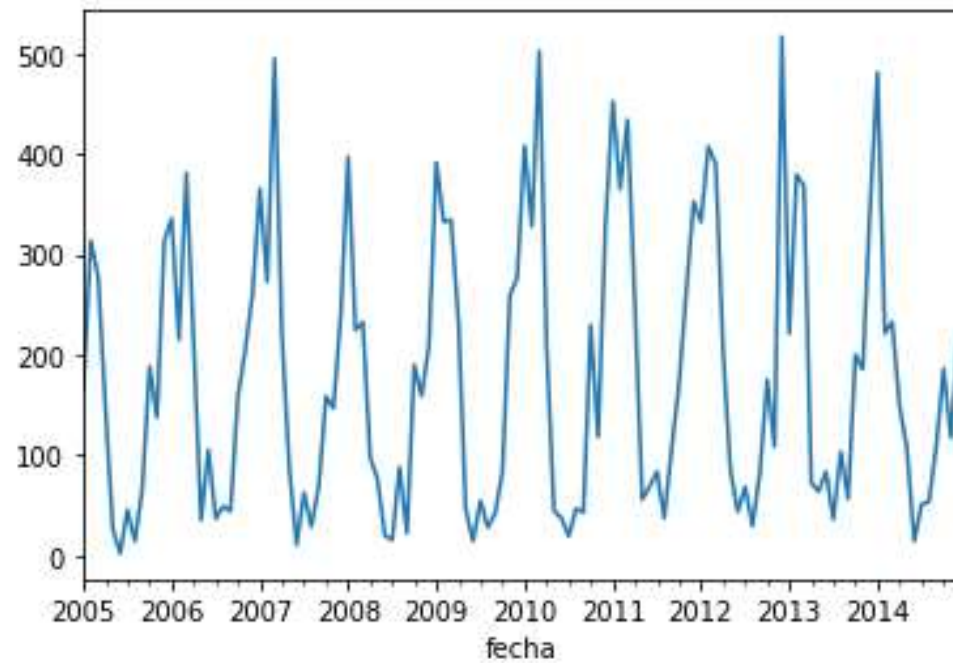
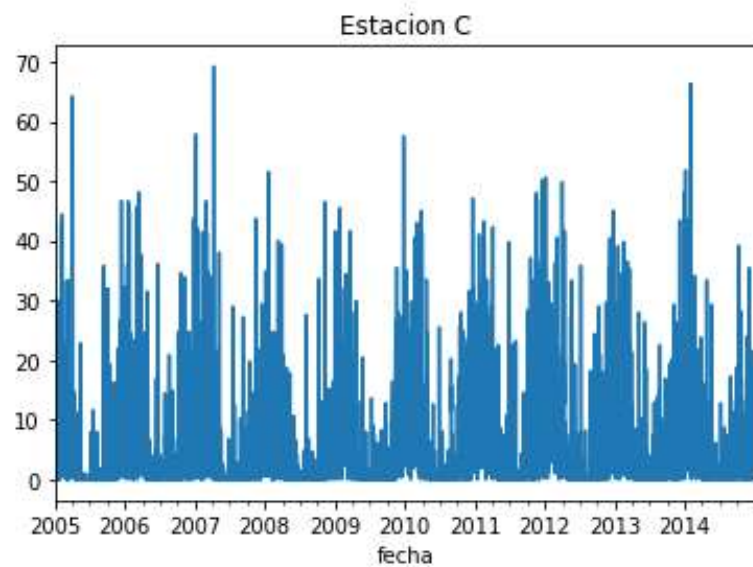
En la consola de scripts de Spyder, crear un código que realice lo siguiente:

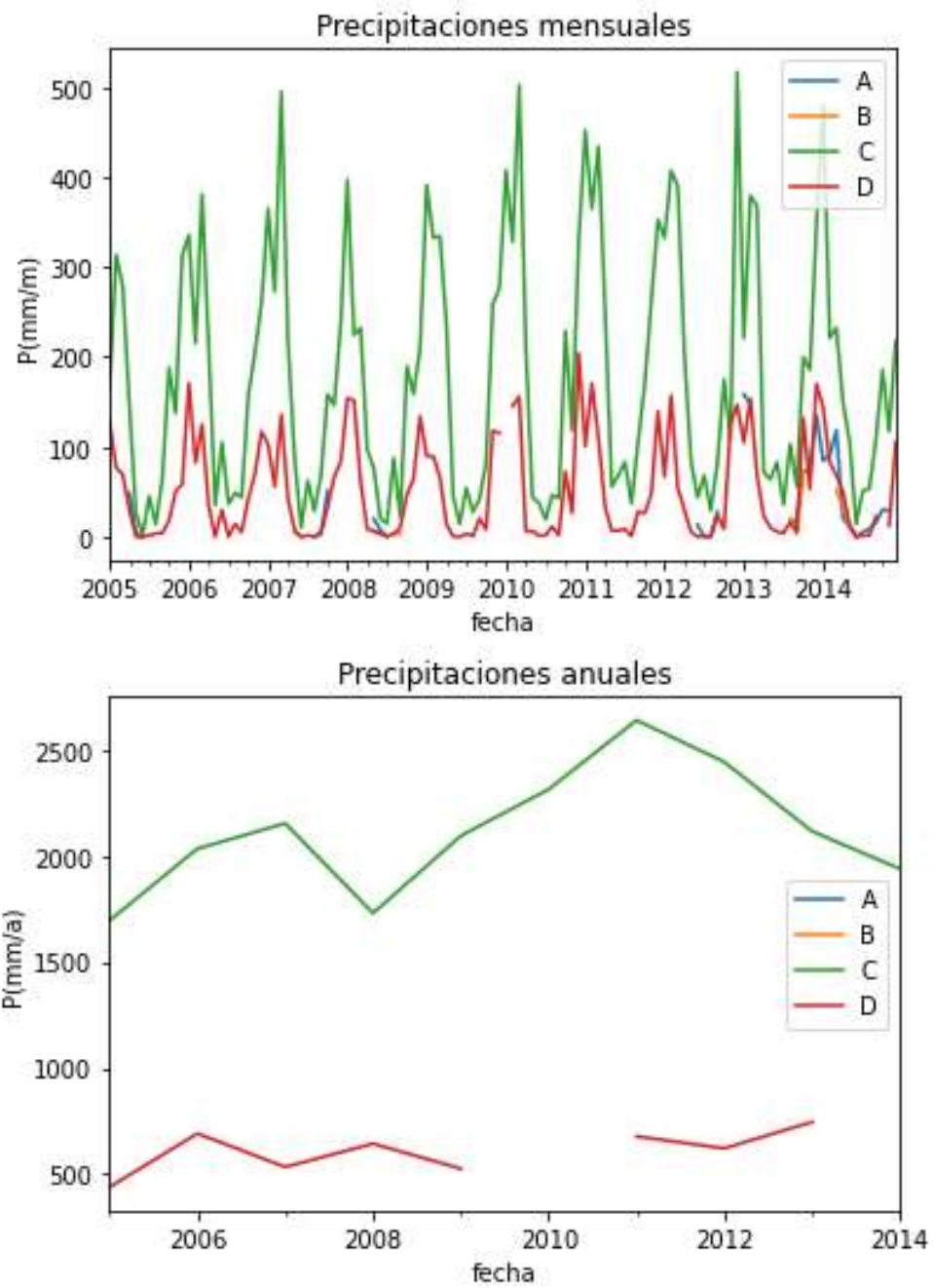
- 1) Leer el archivo de precipitaciones diarias "p_diarias.csv" almacenados en formato matriz con 4 estaciones (identificadores A, B, C, D) desde el 2005-01-01 al 2014-12-31.
- 2) La estación C se encuentra completa, convertir sus datos diarios en mensuales y éstos en anuales. Plotear las series de tiempo mensuales y anuales.
- 3) Plotear todas las estaciones en conjunto y analizar el vacío de información.
- 4) Plotear un boxplot para los datos diarios de cada estación.
- 5) Plotear un boxplot para los datos mensuales de cada estación.
- 6) Plotear un boxplot para los datos anuales de cada estación.

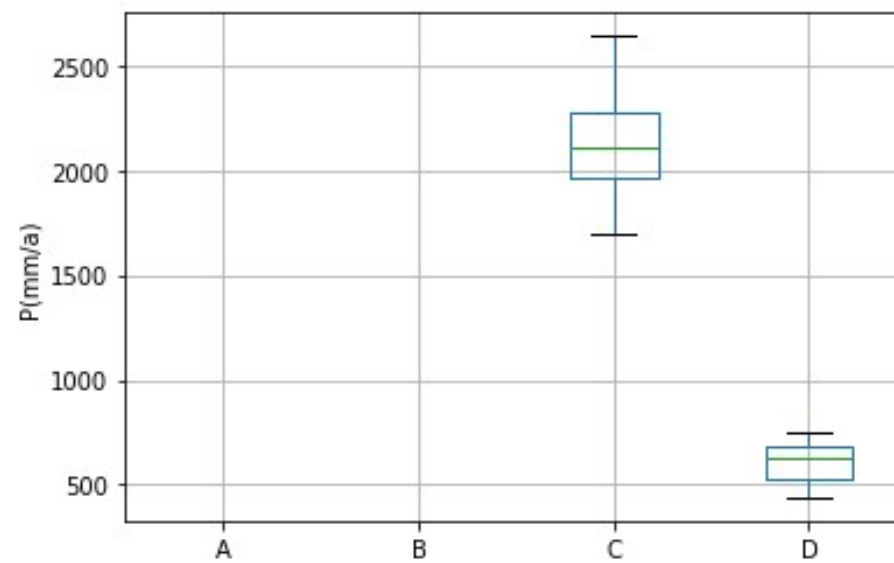
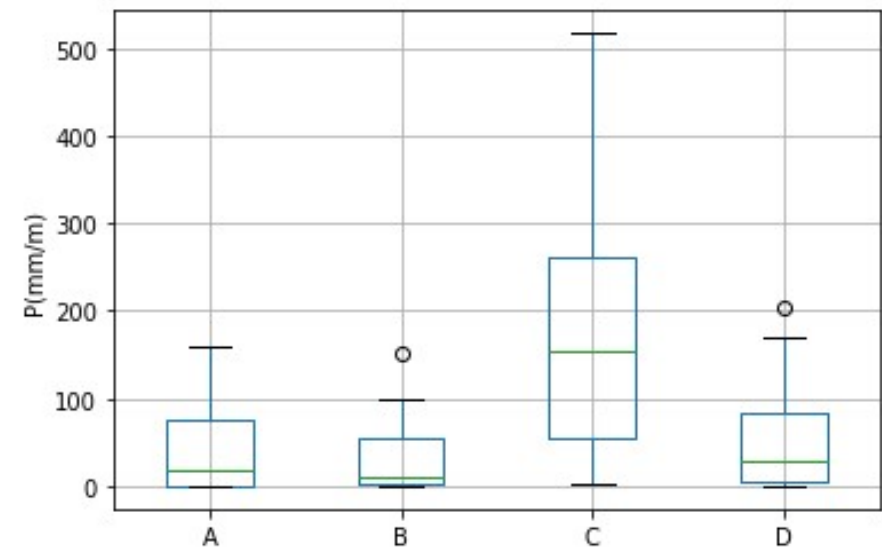
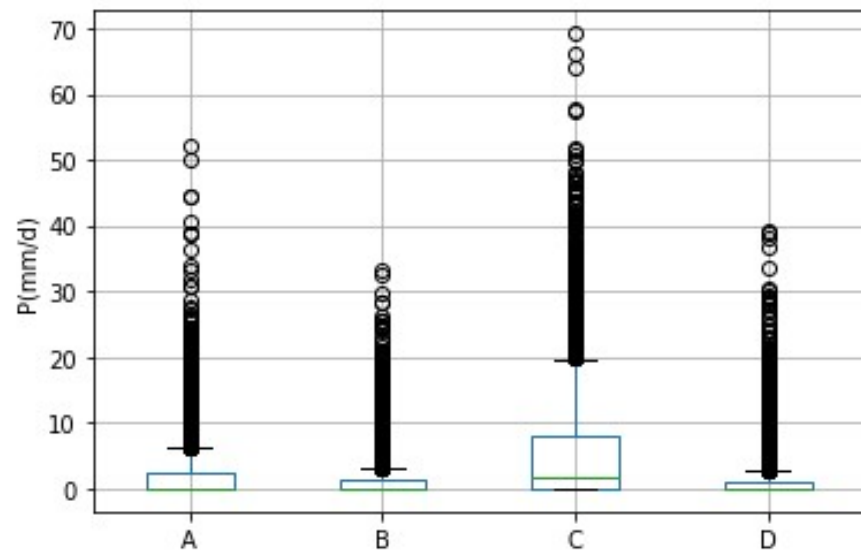
Respuestas C

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
data = pd.read_csv('p_diarias.csv', parse_dates = ['fecha'])
print(data.head())
data.fecha = pd.to_datetime(data.fecha)
data.set_index('fecha', inplace=True)
data['C'].plot()
plt.title("Estacion C")
plt.ylabel("P(mm/d)")
# Agregacion sin considerar a los vacios
prec_m = data.resample('M').agg(pd.Series.sum, skipna=False)
prec_m['C'].plot()
# Agregacion incorrecta considerando a los vacios como valores nulos
prec_a = data.resample('Y').sum()
prec_a['C'].plot()

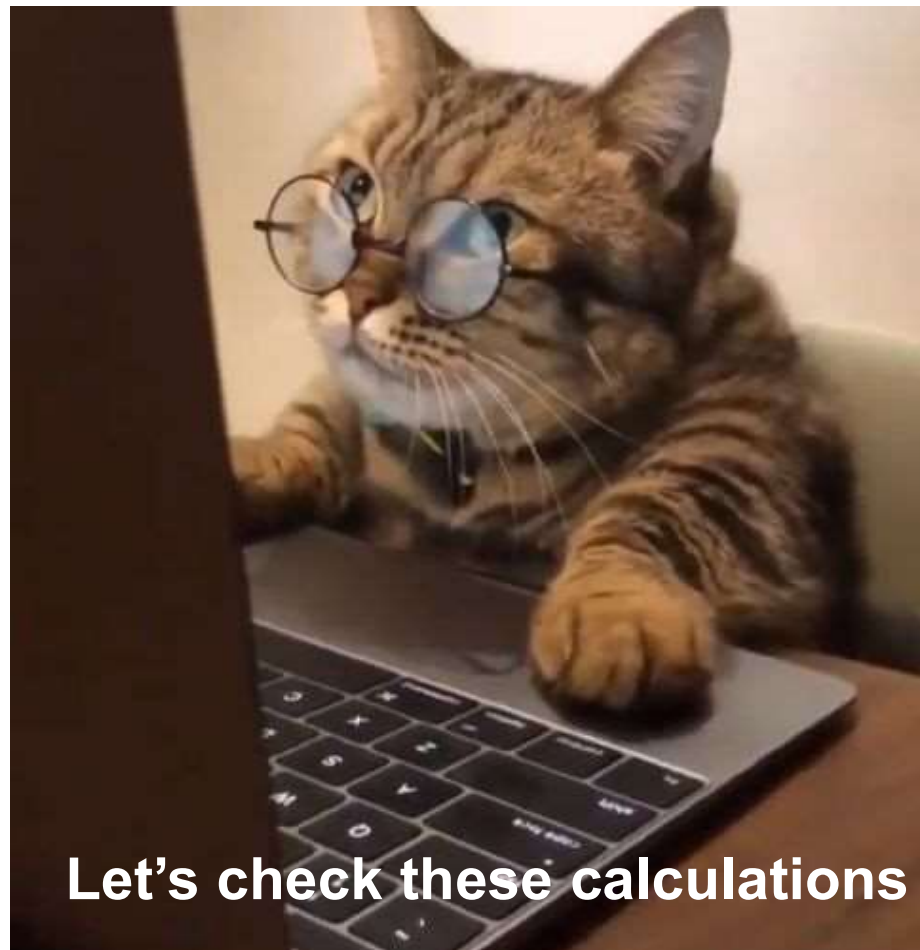
prec_m.plot()
plt.title("Precipitaciones mensuales")
plt.ylabel("P(mm/m)")
prec_a = data.resample('Y').agg(pd.Series.sum, skipna=False)
prec_a.plot()
plt.title("Precipitaciones anuales")
plt.ylabel("P(mm/a)")
# boxplot de diarios
data.boxplot()
plt.ylabel("P(mm/d)")
# boxplot de mensuales
prec_m.boxplot()
plt.ylabel("P(mm/m)")
# boxplot de anuales
prec_a.boxplot()
plt.ylabel("P(mm/a)")
```







4. Aplicaciones hidrológicas



Let's check these calculations

D. Ajuste de un modelo de infiltración

Ejercicio D

Se tiene un suelo de características margo limosas con una saturación efectiva del 25%, una porosidad efectiva de 0.486, una carga de succión en el frente mojado de 16.68 cm y una conductividad hidráulica saturada de 0.65 cm/hr.

- 1) Calcular la infiltración acumulada al cabo de 3hrs, para una carga constante de agua.
- 2) Plotear el gráfico de iteración de la función teorica de infiltración acumulada.



elmercurio

Modelo de Green-Ampt

$$F(t) = \psi \Delta \theta \ln \left(1 + \frac{F(t)}{\psi \Delta \theta} \right) + Kt$$

$F(t)$: Infiltración acumulada después del tiempo t

Ψ : carga de succión en el frente mojado

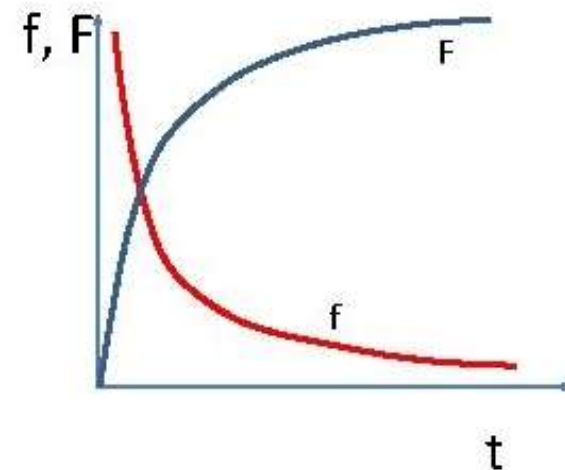
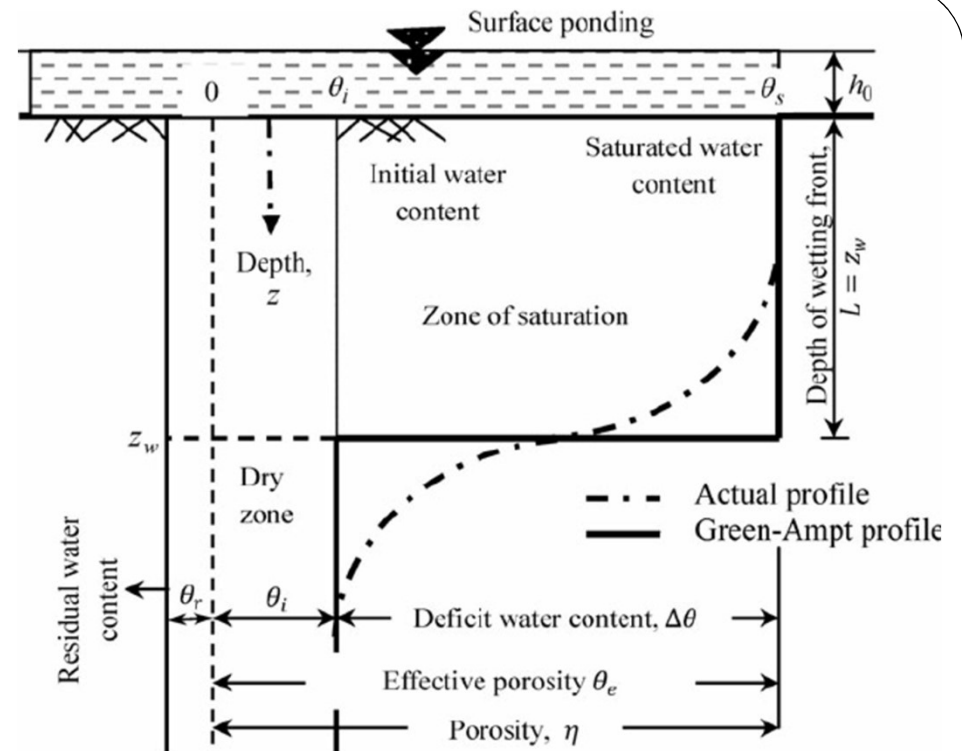
K : Conductividad hidráulica saturada

s_e : saturación efectiva

θ_e : porosidad efectiva

η : porosidad

$$\Delta \theta = \eta - \theta_i = (1 - s_e) \theta_e$$



Respuestas D

```
import numpy as np
import matplotlib.pyplot as plt
# se definen las variables
theta_e = 0.486
psi = 16.68
K = 0.65
S_e = 0.25
t = 2
```

```
#calculando dtheta
dtheta = (1-S_e)*theta_e
```

```
# inicio de F
```

```
F_old = K*t
```

```
epsilon = 1
```

```
F = []
```

```
while epsilon > 1e-4:
```

```
    F_new = psi*dtheta * np.log(1+F_old/(psi*dtheta)) + K*t
```

```
    epsilon = F_new - F_old
```

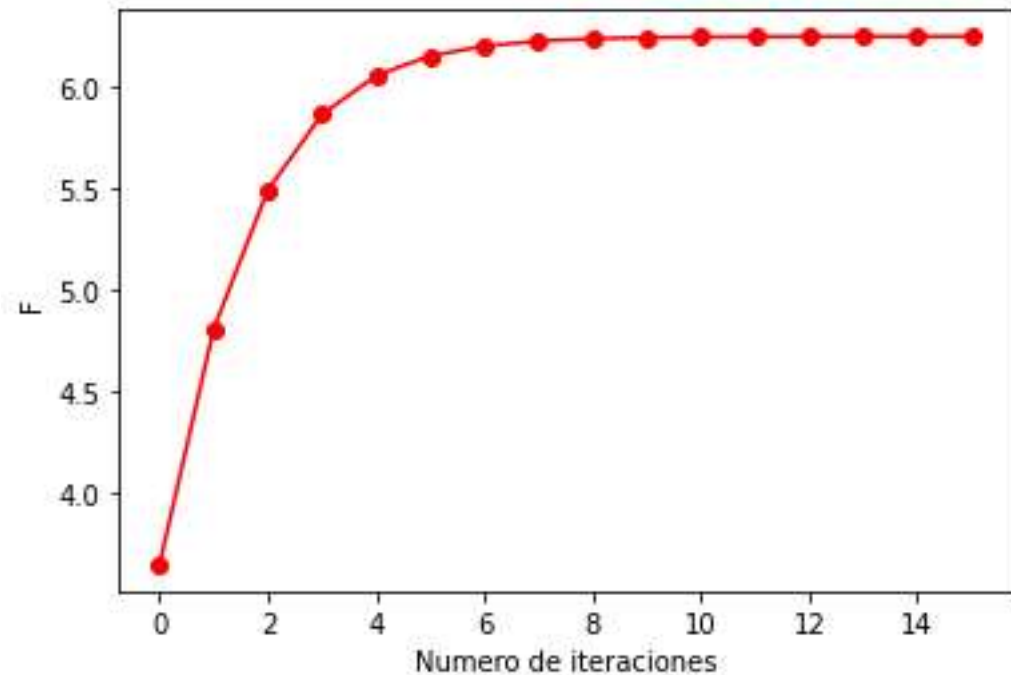
```
    F_old = F_new
```

```
    F.append(F_new)
```

```
plt.plot(F,'-ok')
```

```
plt.xlabel('Numero de iteraciones')
```

```
plt.ylabel('F')
```



F:

```
[3.641328430679825,
4.803437639773873,
5.48998400276456,
5.861905885112726,
6.054272262734779,
6.151430468563465,
6.1999181493797,
6.22397248174879,
6.23587041017472,
6.2417468544535035,
6.244647164564665,
6.246078098423059,
6.246783958089711,
6.247132118614578,
6.247303839105808,
6.247388533673947]
```


E. Profundidad y velocidad en un río mediante optimización

Ejercicio E

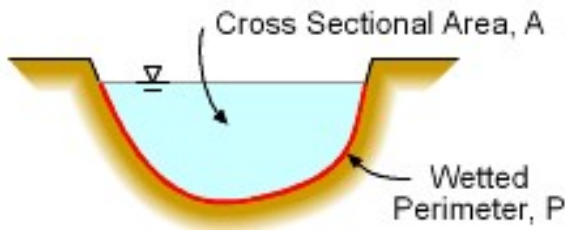
Se tiene un río de sección casi trapezoidal, con una rugosidad de Manning de 0.025, una pendiente de 1%, un ancho de fondo de 10 m y un talud 2H:1V

- 1) Crear una función para el cálculo de la profundidad o tirante del río, empleando una optimización por el método del downhill simplex algorithm (Nelder-Mead).
- 2) Calcular la profundidad, velocidad y número de Froude para un caudal de 8 m³/s.



Respuestas E

Formula de Manning



$$Q = \frac{A \cdot R_h^{2/3} \cdot S^{1/2}}{n}$$

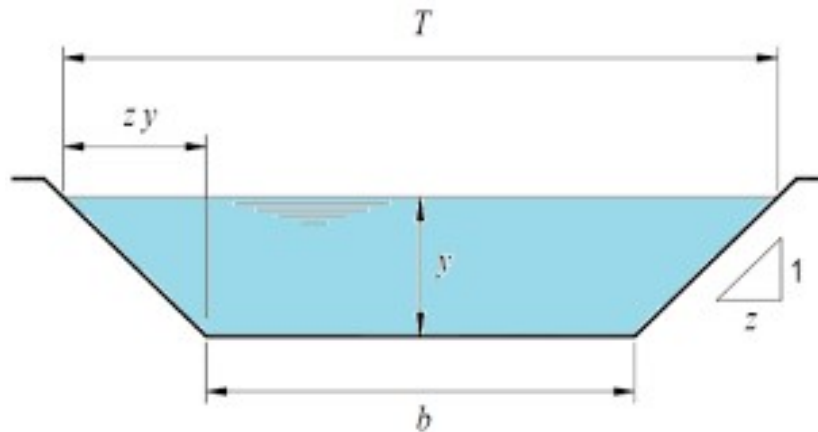
Q: Caudal (m³/s)

A: Area (m²)

R_h: Radio Hidráulico = A/P

S: Pendiente del río

n: Coeficiente de Manning



Para una sección trapezoidal

$$R_h = \frac{(b + zy)y}{b + 2y\sqrt{1 + z^2}}$$

$$V = \frac{Q}{A} \quad V: \text{Velocidad (m/s)}$$

$$Q = \frac{((b + zy)y)^{5/3} \cdot S^{1/2}}{n \cdot (b + 2y\sqrt{1 + z^2})^{2/3}}$$

$$Fr = \frac{V}{\left(\frac{gA}{T}\right)^{1/2}} \quad \begin{array}{l} Fr: \text{Número de Froude} \\ T: \text{Ancho en la superficie} \\ \text{o espejo} \end{array}$$

Respuestas E

```
import numpy as np
from scipy.optimize import fmin
# definiendo las variables
n = 0.025
S = 0.01
Q = 8
b = 10
z = 2
# definiendo la funcion del caudal
def flow(y):
    Q_est = (1/n)*(S**0.5)*((b+z*y)*y)**(5/3)/(b+2*y*(1+z**2)**0.5)**(2/3)
    epsilon = np.abs(Q_est - Q)
    return epsilon

y_optimum = fmin(flow,0.2)
print("tirante =",y_optimum)
vel = Q/((b+z*y_optimum)*y_optimum)
print("velocidad =",vel)
Fr = vel/(9.81*((b+z*y_optimum)*y_optimum)/(b+2*z*y_optimum))**0.5
print("Nro de Froude =",Fr)
```

Optimization terminated successfully.
Current function value: 0.000303
Iterations: 16
Function evaluations: 32
[0.37679688]
[1.97437239]
[1.06230372]

Referencias

Chow V, Maidment D, Mays L. 1994. Hidrologia Aplicada. McGraw-Hill.

Kumar S. 2011. Python in hydrology. Green Tea Press.

Naghetini M. 2017. Fundamentals of Statistical Hydrology. Springer.

Rau P, Bourrel L, Labat D, et al. 2017. Regionalization of rainfall over the Peruvian Pacific slope and coast. *International Journal of Climatology* 37(1):143-158.

Salas J. 1996. Analysis and modelling of hydrologic time series (in Handbook of Hydrology). McGraw-Hill Education.

Swamynathan, M. 2017. Mastering Machine Learning with Python in Six Steps. A Practical Implementation Guide to Predictive Data Analytics Using Python. Apress.

van Rossum, G., Drake, F. 2011. The Python Language Reference Manual. Network Theory Ltd.

VanderPlas J. 2017. Python Data Science Handbook. O'Reilly Media.

Waterloo MJ, Post VEA. 2015. Python programming guide for earth scientists. Amsterdam Critical Zone Hydrology group.