Capacitación Python
Laboratorio 1

Nelson R. Salinas

Abril 4, 2018

En esta segunda sesión el estudiante se familiarizará con las operaciones de lectura de archivos externos, estructuras escalares, iteraciones.

1. El problema

Dada un tabla con datos dasométricos (diámetro y densidad de madera) correspondientes a una parcela de 0.1 ha, estimar la biomasa utilizando la ecuación de Chave et al. [2005] para bosque húmedo:

$$AGB = \rho \times \exp(-1{,}499 + 2{,}148 \times \log(D) + 0{,}207 \times \log(D)^2 - 0{,}028 \times \log(D)^3)$$

donde ρ es la densidad de la madera y D es el diámetro.

La tabla está presentada como un archivo csv: "plot.csv". En este punto se necesario visualizar la información contenida en dicho archivo utilizando una interfaz gráfica (v.g., Excel). ¿Cómo está distribuída la información?

2. Lectura de archivos

Los archivos de texto se pueden leer usando la función `open` bajo el modo de lectura (`r`). Se debe tener en cuenta que esta opción solo funcionará apropiadamente si el archivo está codificado con caracteres ASCII. Esto significa en términos prácticos que el archivo no será leído apropiadamente si contiene caracteres acentuados propios de algunos idiomas romances. Afortunadamente, nuestro archivo con datos de vegetación no sufre de ese problema. Usualmente, cuando se procesa un archivo de texto lo más indicado es leerlo línea por línea, lo cual es realizado con el iterador `for`.

Código 1: Lectura de archivos de texto.

```
1  # Inicializa un conector a un archivo en modo lectura
2  fh = open("plot.csv","r")
3
4  for line in fh: # lee linea por linea
5
6      # procesamiento del texto de la linea
7      print line.rstrip()
8
9  #Finalmente el conector debe ser cerrado
10 fh.close() # cierra el archivo
```

En el Código 1, línea 7 la función rstrip() elimina el caracter de salto de línea del texto en cuestión, algo importante si no es necesario mantener dicho caracter en el procesamiento posterior.

3. Iteración de escalares

Varios pasos del problema se pueden resolver usando escalares (típicamente listas): tanto para segmentar la información o para guardar la información temporalmente. Por ejemplo, cada linea contiene los datos de diámetro y densidad de madera de un individuo. Estos datos se pueden separar utilizando la función `split` de objetos de texto (`str` o `unicode`), la cual retorna un listado de palabras.

Cambie la línea 7 por `print line.split(',')`. ¿Cómo cambiaron los datos de salida?

Una vez se tiene segmentada la información de diámetro y densidad de madera se puede calcular la biomasa de cada árbol. El Código 2 contiene una versión actualizada de los comando necesarios para realizar el cálculo.

Código 2: Lectura de datos y estimación de biomasa.

```
1  from math import log,exp
2
3  # Variable de biomasa total
4  myabg = 0.0
5
6  # Inicializa un conector a un archivo en modo lectura
7  fh = open("plot.csv","r")
8
9  for indx,line in enumerate(fh): # lee linea por linea
10
11     if indx > 0: # Evita procesar encabezado
12
13         # procesamiento del texto de la linea
14         line = line.rstrip()+
15         bits = line.split(',')
16         D = float(bits[0])
17         wd = float(bits[1])
```

```
18          myagb += wd * exp(-1.499 + 2.148 * log(D) + 0.207 * \
19              log(D)**2 - 0.028 * log(D)**3)
20
21  #Finalmente el conector debe ser cerrado
22  fh.close() # cierra el archivo
23
24  print myagb
```

Aclaraciones sobre el Código 2:

Línea 1. Importación de funciones logarítmica y exponencial del módulo matemático.

Línea 9. La función `enumerate` permite iterar un escalar accediendo al tiempo a los índices y los valores; en éste caso, “indx” y “line”, respectivamente.

Líneas 16–17. Acceso a los componentes de la línea a través de indexación.

Líneas 18–19. Cálculo de la biomasa. Nótese la barra inversa al final de la línea 18, lo cual posibilita desplegar una sóla expresión a lo largo de múltiples líneas.

4. Pruebas condicionales

Las ecuación de Chave et al. [2005] fue diseñada principalmente para estimar la biomasa de árboles con un diámetro mayor o igual a 10 cm. Sin embargo, nuestra tabla de medidas contiene datos de árboles de menor diámetro. ¿Cómo se pueden filtrar dichos datos? En este caso es necesario evaluar un subconjunto de la información (diámetro) y realizar una decisión dependiente de dicha evaluación. La forma más sencilla de realizar esta clase de decisiones es utilizando la función `if`.

Código 3: Lectura de datos y estimación de biomasa de árboles con DAP ¿

```
1   from math import log,exp
2
3   # Variable de biomasa total
4   myabg = 0.0
5
6   # Inicializa un conector a un archivo en modo lectura
7   fh = open("plot.csv","r")
8
9   for indx,line in enumerate(fh): # lee linea por linea
10
11      if indx > 0: # Evita procesar encabezado
12
13          # procesamiento del texto de la linea
14          line = line.rstrip()+
15          bits = line.split(',')
16          D = float(bits[0])
17          wd = float(bits[1])
18          if D >= 10:
```

```
19              myagb += wd * exp(-1.499 + 2.148 * log(D) + 0.207 * \
20              log(D)**2 - 0.028 * log(D)**3)
21
22  #Finalmente el conector debe ser cerrado
23  fh.close() # cierra el archivo
24
25  print myagb
```

Referencias

J. Chave, C. Andalo, S. Brown, M. A. Cairns, J. Q. Chambers, D. Eamus, H. Fölster, F. Fromard, N. Higuchi, T. Kira, et al. Tree allometry and improved estimation of carbon stocks and balance in tropical forests. *Oecologia*, 145(1):87–99, 2005.