Archivos. Integración

Temario

- Lectura y escritura de archivos.
- Integración numérica. Diferencias numéricas
- Integración por Monte Carlo.

Lectura de archivos

Supongamos que queremos leer datos de un archivo.
Para esto tenemos que abrir el archivo, open, leer read y cerrar el archivo close.

```
text_file =open("Salida.txt", "r")
data=text_file.read()
text_file.close()
```

En forma mas pythonica:

```
with open('data.txt', 'r') as myfile:
    data=myfile.read()
```

Cuando deja de haber identación, significa que python tiene que cerrar el archivo.

Lectura de archivos

```
text_file =open("Salida.txt", "r")
data=text_file.read()
text_file.close()
```

La sintaxis de estas instrucciones es coherente con objetos (recordar que toda estructura en python es un objeto: desde la mas sencilla, como una variable, hasta las mas complejas). En el primero instanciamos el objeto text_file. Y luego utilizamos ciertas funciones del objeto .read() y .close()

Escritura de archivos

El formato para escritura de un archivo es similar. Para abrir aclaramos es para escritura: “w”

```
text_file =open("Saluda.txt", "w")
text_file.write("Precio: %s" % Cantidad)
text_file.close()
```

```
with open("Output.txt", "w") as text_file:
text_file.write("Precio: %s" % Cantidad)
```

El formato es equivalente a lo que usamos en el print. ej. %6.2f.
El archivo se guarda en formato ascii es decir que si lo abren con un editor de texto pueden leer el contenido.

Guardado de arrays. Numpy

En formato ascii:

```
>>>a =np.array([1, 2, 3, 4])
>>>np.savetxt('test1.txt', a, fmt='%d')
>>>b =np.loadtxt('test1.txt', dtype=int)
```

En formato binario (recomendado):

```
>>>np.save('test3.npy', a)
>>>d =np.load('test3.npy')
```

La extensión *npy* es la que se utiliza para datos binarios en python. Si uds no la agregan python la tomará por default.

pickle

Guardado de objetos completos. Grabacion: Produce una serialización de todo el objeto. Lectura: “Deserialización” para producir el objeto.

```
import pickle

a ={'hello': 'world'}

with open('filename.pkl', 'wb') as handle:
   pickle.dump(a, handle, protocol=pickle.HIGHEST_PROTOCOL)

with open('filename.pkl', 'rb') as handle:
   b =pickle.load(handle)
```

Librería para guardar datos estructurados

En data science existe una librería muy utilizada “pandas” que permite leer y procesar datos estructurados de muy diversa índole.
Pandas trabaja con Series (un vector de datos) y con DataFrame (una tabla de datos mezclados).

```
import pandas as pd
data ={
   'apples': [3, 2, 0, 1],
   'oranges': [0, 3, 7, 2]
}
purchases =pd.DataFrame(data)
```

Trabaja con: (key, value)

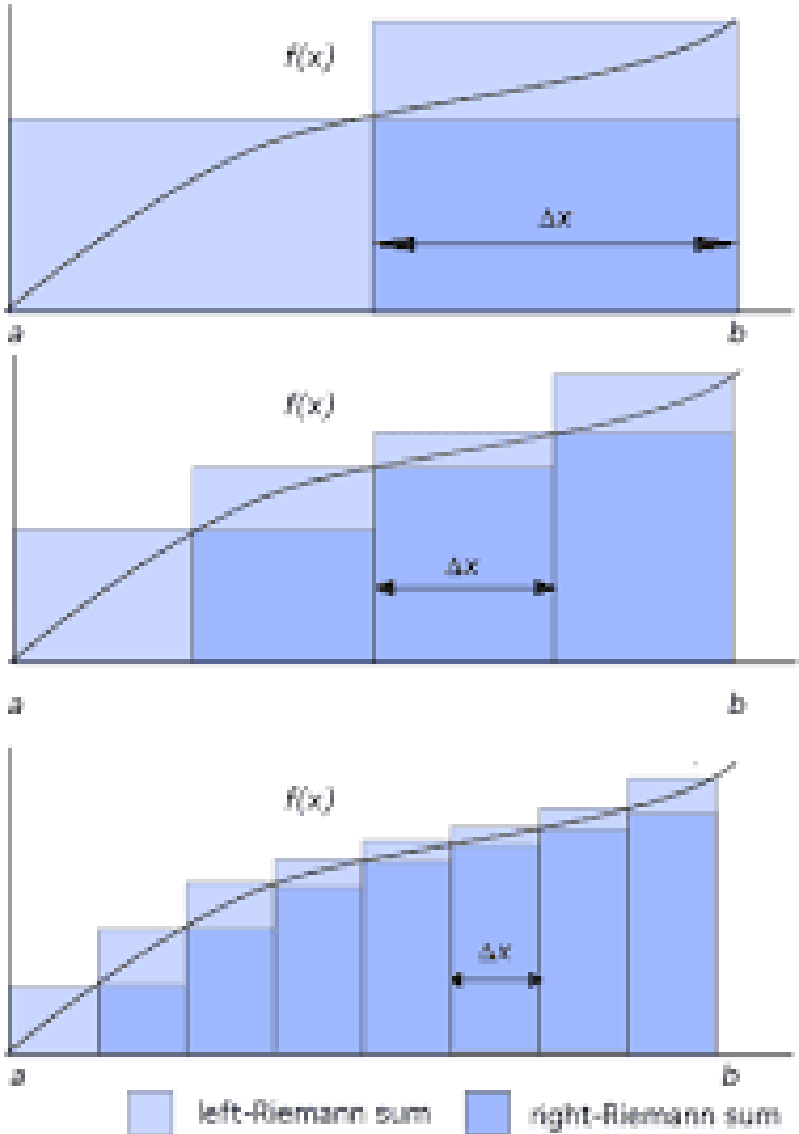
```
df =pd.read_csv('purchases.csv')
df =pd.read_json('purchases.json')

import sqlite3
con =sqlite3.connect("database.db")\
df =pd.read_sql_query("SELECT * FROM purchases", con)
```

Integración numérica

La interpretación de la integral es el área bajo la curva.

Podemos dividir el intervalo de integración en pequeños intervalitos y calcular las areas de los rectangulos.



Subdividimos el intervalo de integración.

Que áreas tomamos? El valor de la función en extremo inferior/superior del intervalito?

El error de la aproximación es el área que queda “sin considerar”.

Mientras mas chicos los intervalitos, mas preciso es la aproximacion del área.

Teorema de Riemann nos garantiza que en el límite $\Delta x \to 0$ converge al valor exacto de la integral.

Integración numérica

$$I = \int_a^b f(x) \approx \sum_{i=0}^{N} A_i f(x_i)$$

donde A_i estaría dando por el método que elijamos.

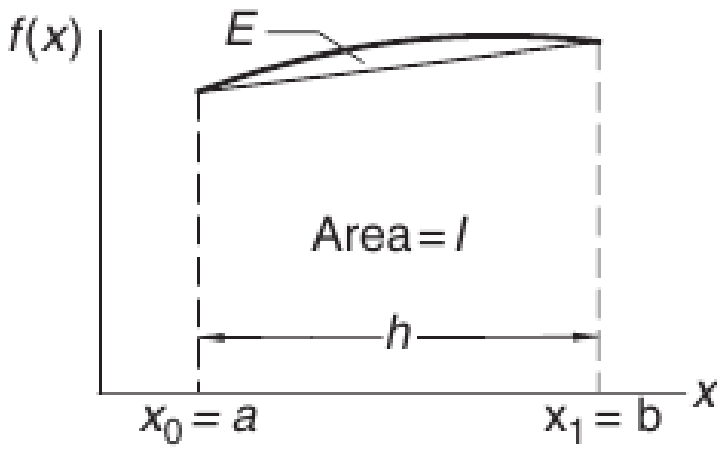
La primera posibilidad es $A_i = \Delta x$ donde $x_0 = a$ y $x_N = b - \Delta x$.

Considerando los puntos extremos superiores. $x_0 = a + \Delta x$ y $x_N = b$.

También se podría pensar en $A_0 = 0$ o $A_N = 0$ y $x_0 = a$ y $x_N = b$.

Regla trapezoidal

Determino la linea que va desde $a, f(a)$ a $b, f(b)$



Supongamos que hay una sola division y queremos aproximar el area. La altura del punto intermedio es:

$$y = (f(a) + f(b))/2 \qquad I = y \cdot \Delta x$$

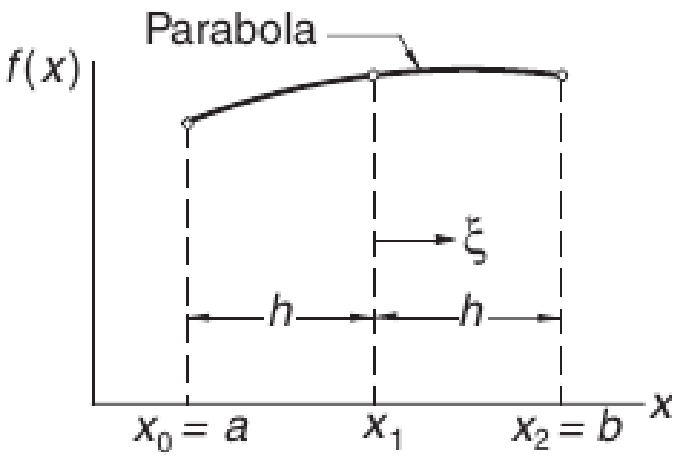
Esto es la regla del trapezio. Esto disminuye la aproximacion de rectangulos anterior.

Realizando la composición

$$I = [f(x_0) + 2f(x_1) + \cdots + 2f(x_{n-1}) + f(x_n)]\frac{\delta x}{2}$$

Regla de Simpson

Determino una parábola que pasa por $a, f(a)$, $(a+b)/2, f((a+b)/2)$ $b, f(b)$



]

Supongamos que hay una sola división y queremos aproximar el área. La integral de la cuadrática que pasa por los puntos es::

$$I = \left[f(a) + 4f(\frac{a+b}{2} + f(b) \right] \frac{\Delta x}{3}$$

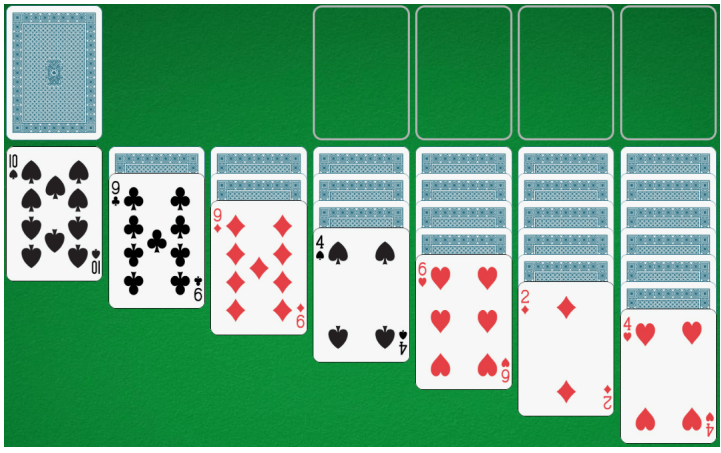
Esto es la regla de Simpson.

Realizando la composición

$$I = [f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \cdots + 4f(x_{n-1}) + f(x_n)] \frac{\delta x}{3}$$

Monte Carlo y Metropolis

La existencia de los metods de Monte Carlo se la debemos al solitario.



La historia cuenta que Stan Ulam estaba postrado enfermo jugando al solitario y quizo saber la probabilidad que tenia de que el juego fuera exitoso.

El manejo combinatorio es extremadamente complicado sino imposible, por lo que rapidamente desistio, y propuso jugar una gran cantidad de veces y registrar la cantidad de veces que era exitoso.

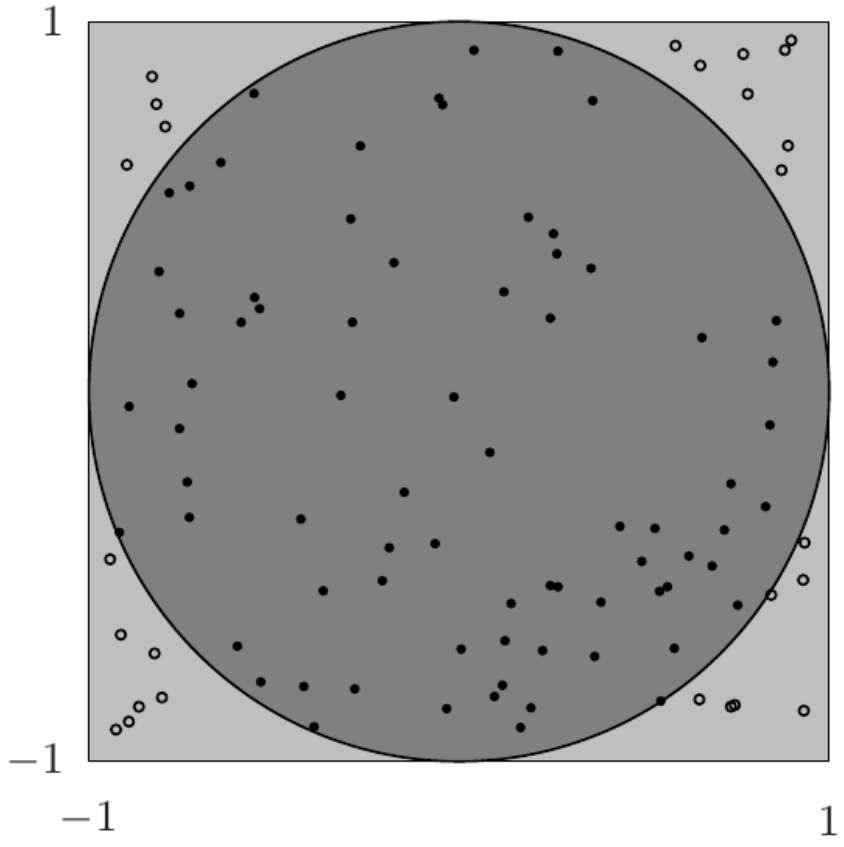
Si las muestras eran aleatorias, y la cantidad de juegos grande esto se deberia aproximar al valor exacto.

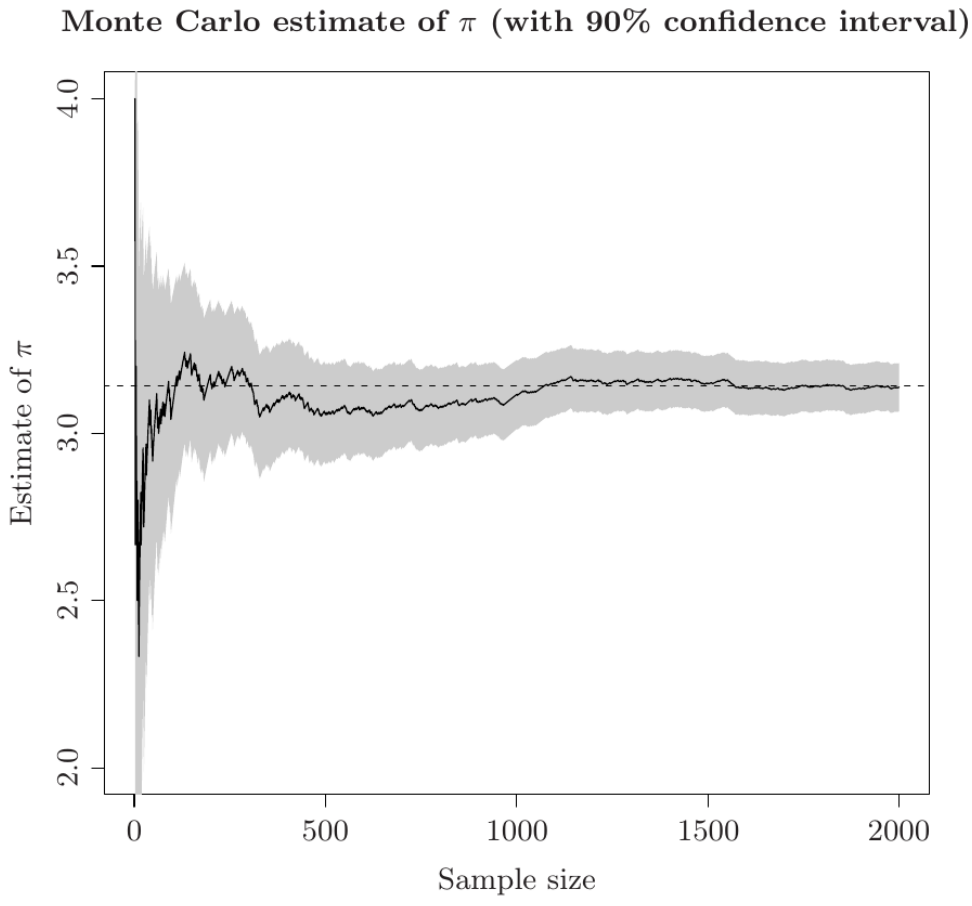
Estimación de π con Monte Carlo

$A = \pi(1/2)^2$ entonces $\pi = 4A = 4\frac{n_{cir}}{n}$.

n_{circ} numero de puntos adentro del circulo. n total de puntos aleatorios.

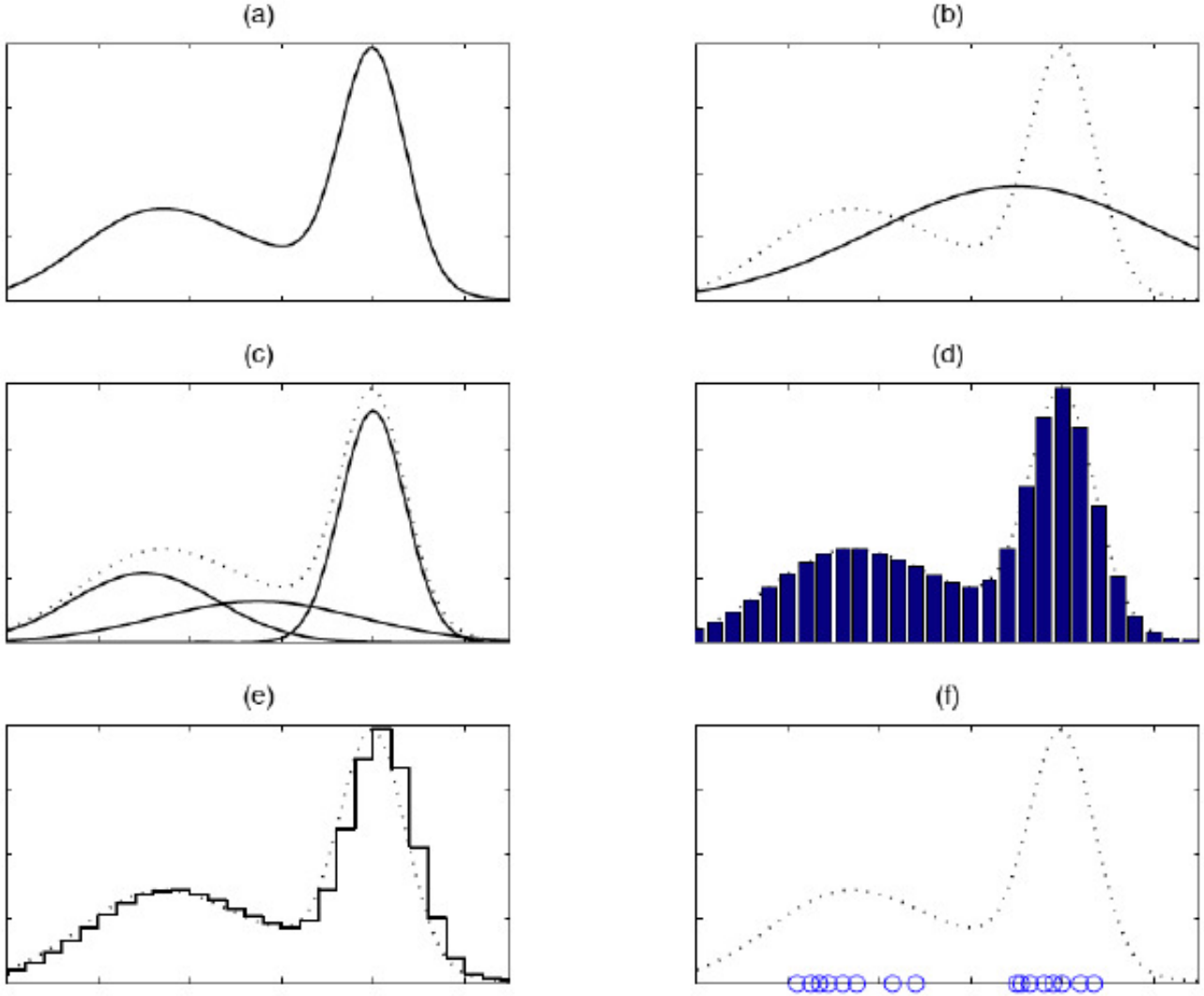
Distribución uniforme en el rectángulo.





Muestreo de Monte Carlo

Formas de representar a una densidad de probabilidad.



Muestreo y Monte Carlo

El objetivo es evaluar la integral

$$\mathcal{E}_p(f(\mathbf{x})) \triangleq \int f(\mathbf{x})p(\mathbf{x})\mathrm{d}\mathbf{x}$$

Lo que se hace es tomar una muestra de N realizaciones independientes de la densidad de probabilidad p (i.i.d.).
Tomamos como representación de la densidad de probabilidad a la muestra

$$p(\mathbf{x}) \doteq \frac{1}{N}\sum_{j=1}^{N}\delta(\mathbf{x}-\mathbf{x}^{(j)})$$

Por lo que el estimador de la integral resulta

$$\mathcal{E}_p^{MC}(f(\mathbf{x})) \doteq \frac{1}{N_p}\sum_{j=1}^{N_p} f(\mathbf{x}^{(j)})$$

Referencia. Andrieu, C., N. de Freitas, A. Doucet, and M. I. Jordan, 2003: An introduction to MCMC for machine learning. Machine Learning 50, 5-43.

Error en la estimación de Monte Carlo

Cuando N_p es lo suficientemente grande, el error de aproximación es arbitrariamente pequeño.
Varianza del estimador:

$$\sigma_N^2(p,f) = \frac{1}{N_p}\mathcal{E}_p[(f - \mathcal{E}(f))^2]$$

Asumiendo la muestra $\{\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(N_p)}\}$ de p

$$\sigma_N^2(p,f) = \frac{1}{N_p}\sum_{j=1}^{N_p}[f(\mathbf{x}^{(j)}) - \mathcal{E}_p^{MC}(f(\mathbf{x}))]^2$$

El intervalo de confianza es $\mathcal{E}_p^{MC}(f(\mathbf{x})) \pm c_\alpha N_p^{-1/2}\sigma_N(p,f)$
Lo importante en la razón de convergencia $N_p^{-1/2}$ es que esta no depende de la dimensionalidad.