Python 3 para impacientes

"Simple es mejor que complejo" (Tim Peters)

jueves, 20 de octubre de 2016

Cálculo estadístico

El módulo **statistics** agrupa un conjunto de funciones para cálculo estadístico.

Las funciones están organizadas en dos grupos: las dedicadas a calcular valores medios y promedios y otras que miden la dispersión de los datos alrededor de un valor central en una muestra o población de datos.

Una **población** en la ciencia Estadística es el conjunto de todos los elementos objeto de un estudio y una **muestra** es un subconjunto, extraído de la población, cuyo estudio sirve para inferir características de toda la población. Por otro lado, un **individuo** es cada uno de los elementos que forman una población o una muestra.

El módulo **statistics** se incluyó en la librería estándar a partir de Python 3.4.

Cálculo de promedios y de valores medios

Media: mean(data)

La función **mean()** devuelve la **media aritmética** de los datos de una muestra o población. Esta media se calcula sumando todos los valores y después el resultado de la suma se divide entre el número de elementos.

Si no se facilitan elementos la función producirá el error ***StatisticsError***.

```
import statistics as stats

edades = [22, 23, 26, 26, 26, 34, 34, 38, 40, 41]
print(stats.mean(edades))  # 31
```

Media armónica: harmonic_mean(data)

La función **harmonic_mean()** -nueva en Python 3.6- devuelve la **media armónica** (H) de los datos de una muestra o población. La media armónica se corresponde con el recíproco de la media aritmética de los inversos de los datos.

Dicho de otro modo, la media armónica de tres valores (x, y, z) será equivalente a 3/(1/x+1/y+1/z). Es un tipo de media apropiada cuando se trabaja con tasas o proporciones en cálculos financieros o que emplean velocidades y tiempos.

Si la expresión utilizada no tiene datos o cualquiera de ellos es menor que 0 la función producirá el error ***StatisticsError***.

```
edades = [22, 23, 26, 26, 26, 34, 34, 38, 40, 41]
print(stats.harmonic_mean(edades))  # 30.89
```

Buscar

Buscar

Python para impacientes

Python
IPython
EasyGUI
Tkinter
JupyterLab
Numpy

Anexos

Guía urgente de MySQL
Guía rápida de SQLite3

Entradas + populares

Dar color a las salidas en la consola
En Python para dar color a las salidas en la consola (o en la terminal de texto) existen varias posibilidades. Hay un método basado ...

Instalación de Python, paso a paso
Instalación de Python 3.6 A finales de 2016 se produjo el lanzamiento de Python 3.6 . El propósito de esta entrada es mostrar, pas...

Añadir, consultar, modificar y suprimir elementos en Numpy
Acceder a los elementos de un array. [], [,], ... Acceder a un elemento de un array. Para acceder a un elemento se utiliz...

Variables de control en Tkinter
Variables de control Las variables de control son objetos especiales que se asocian a los widgets para almacenar sus valore...

Cálculo con arrays Numpy
Numpy ofrece todo lo necesario para obtener un buen rendimiento cuando se trata de hacer cálculos con arrays. Por como está concebid...

Tkinter: interfaces gráficas en Python
Introducción Con Python hay muchas posibilidades para programar una interfaz gráfica de usuario (GUI) pero Tkinter es fácil d...

Operaciones con fechas y horas. Calendarios
Los módulos datetime y calendar amplían las posibilidades del módulo time que provee funciones para manipular expresiones de ti...

Convertir, copiar, ordenar, unir y dividir arrays Numpy
Esta entrada trata sobre algunos métodos que se utilizan en Numpy para convertir listas en arrays y viceversa; para copiar arrays d...

Tkinter: Tipos de ventanas
Ventanas de aplicación y de diálogos En la entrada anterior tratamos los distintos gestores de geometría que se utilizan para di...

Threading: programación con hilos (I)
En programación, la técnica que permite que una aplicación ejecute

Mediana: median(data)

La función **median()** devuelve la **mediana** o valor medio de los datos. La mediana es el valor central de un grupo de números ordenados por tamaño. Si la cantidad de elementos es par, la mediana es el promedio de los dos números centrales.

La función no requiere que la secuencia de números esté ordenada. Si no hay elementos la función retorna el error ***StatisticsError***.

```
edades = [22, 23, 26, 26, 26, 34, 34, 38, 40, 41]
print(stats.median(edades))  # 30
```

Mediana baja: median_low(data)

La función **median_low()** devuelve la **mediana baja** o el valor central inferior de los datos. Este valor siempre será uno del conjunto de los datos. Cuando el número de elementos sea impar devolverá el valor central; y cuando sea par, el menor de los dos valores centrales.

Si no hay elementos la función devuelve el error ***StatisticsError***

```
edades = [22, 23, 26, 26, 26, 34, 34, 38, 40, 41]
print(stats.median_low(edades))  # 26
```

Mediana alta: median_high(data)

La función **median_high()** devuelve la **mediana alta** o el valor central superior de los datos y siempre será alguno del conjunto de datos. Cuando el número de elementos sea impar la función devolverá el valor central; y cuando sea par, el mayor de los dos valores centrales.

Si no hay elementos la función devuelve el error ***StatisticsError***

```
edades = [22, 23, 26, 26, 26, 34, 34, 38, 40, 41]
print(stats.median_high(edades))  # 34
```

Mediana agrupada: median_grouped(data, interval=1)

La función **median_grouped()** devuelve la **mediana de los datos agrupados** por su frecuencia. Opcionalmente, los datos pueden estar agrupados en **intervalos**.

La mediana es el valor del término medio que divide la distribución de los datos ordenados en dos partes iguales. Para calcular la posición de la mediana con datos agrupados en tablas de frecuencias se utiliza la siguiente ecuación:

Posición_de_mediana = (número_de_datos + 1) / 2

La mediana es el valor de la posición resultante; o si el número de elementos es par será la media de los dos valores centrales.

Ejemplo: lista con 20 números (no es necesario que esté ordenada)

datos = [1, 2, 2, 2, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 6, 6]

Los datos anteriores agrupados por frecuencia quedarían así:

1 - 1
2 - 3
3 - 2
4 - 4
5 - 8
6 - 2

Total de elementos = 20

(20 + 1) / 2 = 10.5, la mediana se encuentra entre el elemento que ocupa la posición 10 y 11, es decir, los valores 4 y 5.

Finalmente, para calcular la mediana agrupada se calcula la media de los valores anteriores: (4+5)/2 = 4.5

Mediana agrupada = 4.5

```
datos=[1, 2, 2, 2, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 6, 6]
print(stats.median_grouped(datos))
```

Si los datos están agrupados en intervalos la mediana se encontrará en el intervalo de clase donde la frecuencia acumulada llega hasta la mitad de la suma de las frecuencias absolutas (número_de_datos / 2)

```
datos=[1, 2, 2, 2, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 6, 6]
print(stats.median_grouped(datos, interval=1)) # 4.5
print(stats.median_grouped(datos, interval=2)) # 4.0
```

Para conocer los cálculos que se realizan cuando se utilizan intervalos se recomienda examinar el código fuente de la función.

Si no hay elementos la función producirá el error ***StatisticsError***

Moda: mode(data)

La función **mode()** devuelve el valor más frecuente de los datos.

Si no hay elementos la función produce el error ***StatisticsError***

```
edades = [22, 23, 26, 26, 26, 34, 34, 38, 40, 41]
print(stats.mode(edades))  # 26
```

Medidas de dispersión

Varianza (población): pvariance(data, mu=None)

La función **pvariance()** devuelve la **varianza de una población** de datos. Esta función es equivalente a **variance()** pero se aplica a una población en lugar de a una muestra.

La varianza es la media de las diferencias con la media elevadas al cuadrado.

En definitiva, la función **pvariance()** para obtener la varianza primero calcula la media de los elementos:

edades = [22, 23, 26, 26, 26, 34, 34, 38, 40, 41]
media = (22 + 23 + 26 + 26 + 26 + 34 + 34 + 38 + 40 + 41) / 10 = 31

A continuación, a cada número se le resta la media obtenida y se eleva al cuadrado. Después, se suman todos los resultados:

dif_cuadrado = (22-31)**2 + (23-31)**2 + (26-31)**2 + (26-31)**2 + (26-31)**2 + (34-31)**2 + (34-31)**2 + (38-31)**2 + (40-31)**2 + (41-31)**2 = 468

Finalmente, se calcula la media de todas las diferencias al cuadrado:

varianza = 468 / 10 = 46.8

La varianza mide en qué medida se dispersan los valores de una muestra alrededor de un valor central.

```
edades = [22, 23, 26, 26, 26, 34, 34, 38, 40, 41]
print(stats.pvariance(edades))  # 46.8
```

El argumento **mu** es opcional y si se facilita debe ser la media de los datos de la población. En caso contrario, la función calculará este dato automáticamente.

Si no hay elementos o hay menos de dos valores la función producirá el error ***StatisticsError***

Desviación estándar (población): pstdev(data, mu=None)

La función **pstdev()** devuelve la **desviación estándar de una población** de datos, que se corresponde con la raíz cuadrada de la varianza de dicha población. Esta función es equivalente a **stdev()** pero se aplica a una población en lugar de a una muestra.

La desviación estándar mide cuánto se separan los datos. Así, para cada valor se tiene una forma de saber si es muy grande, muy pequeño o normal. Permite conocer a qué distancia están los valores de la desviación estándar de la media.

El argumento **mu** es opcional y si se facilita debe ser la media de los datos de la población. En caso contrario, la función lo calculará automáticamente.

```
edades = [22, 23, 26, 26, 26, 34, 34, 38, 40, 41]
print(stats.pstdev(edades))  # 6.84
```

Varianza (muestra): variance(data, xbar=None)

La función **variance()** devuelve la **varianza de una muestra** de datos. La varianza mide en qué medida se dispersan los valores de una muestra alrededor de un valor central.

El argumento **xbar** es opcional y si se facilita debe ser la media de los datos. En caso contrario, la función calculará este dato automáticamente.

Para calcular la varianza de una población entera utilizar la función **pvariance()**.

Si no hay elementos o hay menos de dos la función producirá el error ***StatisticsError***

```
muestra_edades = [22, 23, 26, 26, 34, 38, 40]
print(stats.variance(muestra_edades))  # 54.14
```

Desviación estándar (muestra): stdev(data, xbar=None)

La función **stdev()** devuelve la **desviación estándar de una muestra** de datos, que se corresponde con la raíz cuadrada de la varianza de dicha muestra.

La desviación estándar mide cuánto se separan los datos. Así, para cada valor se tiene una forma de saber si es muy grande, muy pequeño o normal. Permite conocer a qué distancia están los valores de la desviación estándar de la media.

Para calcular la desviación estándar de una población entera utilizar la función **pstdev()**.

El argumento **xbar** es opcional y si se facilita debe ser la media de los datos. En caso contrario, la función lo calculará automáticamente.

```
muestra_edades = [22, 23, 26, 26, 34, 38, 40]
print(stats.stdev(muestra_edades))  # 7.35
```

Ir al índice del tutorial de Python

Publicado por Pherkad en 15:31

Etiquetas: Python3

Entrada más reciente Inicio Entrada antigua