ACÁMICA

¡Bienvenidos/as a Data Science!

Agenda

Repaso: Python + Numpy

Exposición: Probabilidad y Estadística

Hands-on training

Break

Exposición: Pandas

Hands-on training

ENCUESTA

Repaso:
Python + Numpy

PRIMEROS PASOS CON PYTHON

UN LENGUAJE DE PROGRAMACIÓN VIENE CON...

tipos de datos	Números, texto, variables de verdad (bool), etc.
estructuras de datos	Podemos hacer “conjuntos” de cosas y agruparlas de formas específicas. ¡Y vienen con funcionalidades propias! Ejemplo: listas.
funciones propias	Ejemplo: print(), type(), etc.

Vimos, además, que podemos definir Variables.

PRIMEROS PASOS CON PYTHON

TIPOS DE DATOS

Enteros	Floats	Strings	Booleanos
Son los números que usamos para contar, el 0 y los negativos	Son los números “con coma” Se introducen usando puntos	Texto Se introducen entre comillas dobles, “”, o simples, ‘’.	Variables de “verdad”: verdadero o Falso
-1 0 1 2	5.1 -1.3 1.0 10.0	“Hola Mundo” “A” ‘Mi nombre es Esteban’	True False 1 == 2 1 == 1
[1]: `type(3)` [1]: `int`	[1]: `type(3.0)` [1]: `float`	[1]: `type("Hola")` [1]: `str`	[1]: `type(2==2)` [1]: `bool`

PYTHON

¿Qué es una lista? ¿Cuáles son las diferencias con un arreglo de Numpy?

```
In [47]: lista_1 = [2, 4.7, True, 'Texto']
         type(lista_1)

Out[47]: list

In [49]: lista_2 = [0, lista_1, 'Mas texto']
         print(lista_2)

         [0, [2, 4.7, True, 'Texto'], 'Mas texto']
```

PYTHON

¿Qué es una lista? ¿Cuáles son las diferencias con un arreglo de Numpy?

```
In [47]: lista_1 = [2, 4.7, True, 'Texto']
         type(lista_1)

Out[47]: list

In [49]: lista_2 = [0, lista_1, 'Mas texto']
         print(lista_2)

         [0, [2, 4.7, True, 'Texto'], 'Mas texto']
```

¿Qué es un Loop y cómo se hacen en Python?

```
In [64]: lista_1 = [10, 20, 30,]

         for item in lista_1:
             doble = 2*item
             print(doble)

         20
         40
         60
```

PYTHON

¿Qué es una lista?
¿Cuáles son las diferencias con un arreglo de Numpy?

```
In [47]: lista_1 = [2, 4.7, True, 'Texto']
         type(lista_1)

Out[47]: list

In [49]: lista_2 = [0, lista_1, 'Mas texto']
         print(lista_2)

         [0, [2, 4.7, True, 'Texto'], 'Mas texto']
```

¿Qué es un Loop y cómo se hacen en Python?

```
In [64]: lista_1 = [10, 20, 30,]

         for item in lista_1:
             doble = 2*item
             print(doble)

         20
         40
         60
```

¿A qué llamamos condiciones?
Escribir un ejemplo.

```
In [77]: nombre = 'Pedro'

         if nombre == 'Juan':
             print('Esta persona se llama Juan')
         else:
             print('Esta persona NO se llama Juan')

         Esta persona NO se llama Juan
```

¿Y si Python no alcanza?

Numpy

A veces, las estructuras de datos que vienen con Python - y sus funcionalidades asociadas - no son suficientes. Para eso necesitamos usar Librerías.

Nuestra primera librería:

Numpy

- **Fundamental para hacer cálculo numérico con Python**
- **Muy buena documentación**
- **Como muchas librerías, trae una estructura de datos propia: los arrays o arreglos.**

Numpy

- **Fundamental para hacer cálculo numérico con Python**
- **Muy buena documentación**
- **Como muchas librerías, trae una estructura de datos propia: los arrays o arreglos.**

array: a primer orden, es como una lista. De hecho, se pueden crear a partir de una lista.

Importamos la librería (*numpy*) y le ponemos un nombre (*np*)

```
[1]: import numpy as np

     arreglo = np.array([1,2,3,4,5])
     arreglo

[1]: array([1, 2, 3, 4, 5])

[2]: print(arreglo)

     [1 2 3 4 5]
```

Es una lista

Numpy

Si bien lo creamos a partir de una lista, tiene muchas más funcionalidades:

```
[1]: import numpy as np

     arreglo = np.array([1,2,3,4,5])
     arreglo

[1]: array([1, 2, 3, 4, 5])

[2]: print(arreglo)

     [1 2 3 4 5]

[3]: arreglo + 2

[3]: array([3, 4, 5, 6, 7])
```

¡ANDUVO!

Numpy

Formas de crear arreglos de numpy

- Ya vimos a partir de una lista

```
[1]: import numpy as np

arreglo = np.array([1,2,3,4,5])
arreglo
```

- ¿Qué hace np.arange()?

Numpy

Formas de crear arreglos de numpy

- Ya vimos a partir de una lista

```
[1]: import numpy as np

arreglo = np.array([1,2,3,4,5])
arreglo
```

- ¿Qué hace np.arange()? Arreglo en un rango de valores, de “a.saltos”.

```
[4]: arreglo_1 = np.arange(2,9)
arreglo_1

[4]: array([2, 3, 4, 5, 6, 7, 8])
```

```
[6]: arreglo_2 = np.arange(2,9,2)
arreglo_2

[6]: array([2, 4, 6, 8])
```

Numpy

Formas de crear arreglos de numpy

- **¿Qué hace np.linspace()?**

Numpy

Formas de crear arreglos de numpy

- ¿Qué hace np.linspace()? Arreglo equiespaciado

```
[10]: arreglo_3 = np.linspace(2,9,3)
      arreglo_3

[10]: array([2. , 5.5, 9. ])
```

```
[11]: arreglo_4 = np.linspace(2,9,20)
      arreglo_4

[11]: array([2.        , 2.36842105, 2.73684211, 3.10526316, 3.47368421,
             3.84210526, 4.21052632, 4.57894737, 4.94736842, 5.31578947,
             5.68421053, 6.05263158, 6.42105263, 6.78947368, 7.15789474,
             7.52631579, 7.89473684, 8.26315789, 8.63157895, 9.        ])
```

Y algunas más que veremos más adelante.

Numpy

Seleccionando elementos de un arreglo:

- Si queremos ver una posición arbitraria:

```
[21]: arreglo = np.arange(2,20,4)
      arreglo

[21]: array([ 2,  6, 10, 14, 18])

[22]: print(arreglo[0], arreglo[2], arreglo[-1], arreglo[-4])

      2 10 18 6
```

Numpy

Seleccionando elementos de un arreglo:

- **Si queremos ver una posición arbitraria:**

```
[21]: arreglo = np.arange(2,20,4)
      arreglo

[21]: array([ 2,  6, 10, 14, 18])

[22]: print(arreglo[0], arreglo[2], arreglo[-1], arreglo[-4])
      2 10 18 6
```

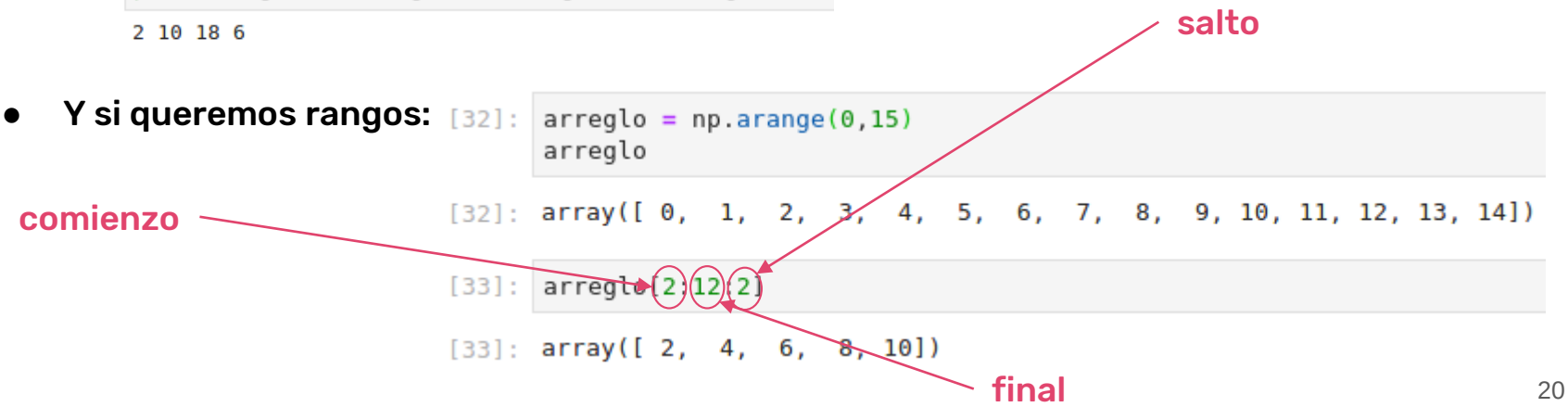
- **Y si queremos rangos:**

```
[32]: arreglo = np.arange(0,15)
      arreglo

[32]: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14])

[33]: arreglo[2:12:2]

[33]: array([ 2,  4,  6,  8, 10])
```



Numpy

Seleccionando también podemos asignar:

```
[34]: arreglo = np.arange(0,15)
      arreglo
```

```
[34]: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14])
```

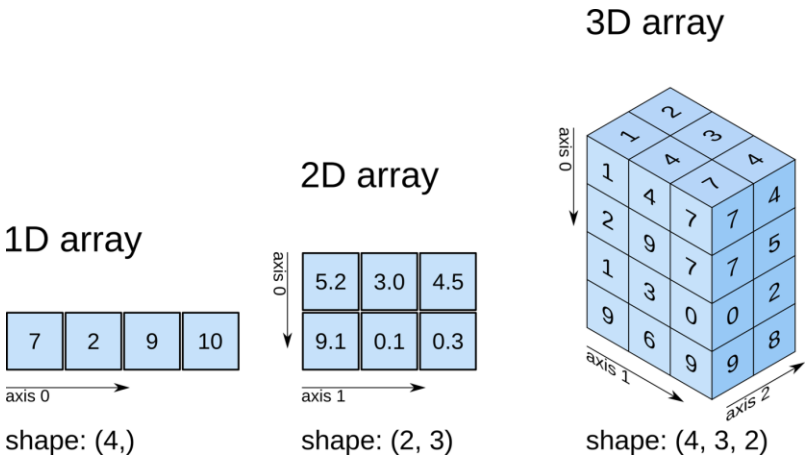
```
[35]: arreglo[2:7] = 25
      arreglo
```

```
[35]: array([ 0,  1, 25, 25, 25, 25, 25,  7,  8,  9, 10, 11, 12, 13, 14])
```

Numpy

Arreglos multidimensionales

“Shape” y “axis” de los arreglos



No es la forma más cómoda de crearlo

```
[39]: arreglo2d = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])
      arreglo2d

[39]: array([[ 1,  2,  3,  4],
             [ 5,  6,  7,  8],
             [ 0, 10, 11, 12]])

[40]: arreglo2d.shape

[40]: (3, 4)
```

filas

columnas

Numpy

Arreglos multidimensionales

```
[42]: arreglo2d = np.arange(100).reshape(10,10)
      arreglo2d
```

```
[42]: array([[ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9],
             [10, 11, 12, 13, 14, 15, 16, 17, 18, 19],
             [20, 21, 22, 23, 24, 25, 26, 27, 28, 29],
             [30, 31, 32, 33, 34, 35, 36, 37, 38, 39],
             [40, 41, 42, 43, 44, 45, 46, 47, 48, 49],
             [50, 51, 52, 53, 54, 55, 56, 57, 58, 59],
             [60, 61, 62, 63, 64, 65, 66, 67, 68, 69],
             [70, 71, 72, 73, 74, 75, 76, 77, 78, 79],
             [80, 81, 82, 83, 84, 85, 86, 87, 88, 89],
             [90, 91, 92, 93, 94, 95, 96, 97, 98, 99]])
```

```
[43]: arreglo2d[2:5,::2]
```

```
[43]: array([[20, 22, 24, 26, 28],
             [30, 32, 34, 36, 38],
             [40, 42, 44, 46, 48]])
```

¿Qué está haciendo?

Numpy

Filtros Booleanos/Máscaras

```
[66]: arreglo2d = np.arange(30).reshape(6,5)
      arreglo2d
```

```
[66]: array([[ 0,  1,  2,  3,  4],
             [ 5,  6,  7,  8,  9],
             [10, 11, 12, 13, 14],
             [15, 16, 17, 18, 19],
             [20, 21, 22, 23, 24],
             [25, 26, 27, 28, 29]])
```

Creamos la máscara

```
[67]: mask = arreglo2d < 20
      mask
```

```
[67]: array([[ True,  True,  True,  True,  True],
             [ True,  True,  True,  True,  True],
             [ True,  True,  True,  True,  True],
             [ True,  True,  True,  True,  True],
             [False, False, False, False, False],
             [False, False, False, False, False]])
```

Y seleccionamos aquellos elementos que cumplen la condición que representa la máscara

```
[68]: arreglo2d[mask]
```

```
[68]: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
             17, 18, 19])
```

Numpy

Funciones de Numpy

Hay muchas funciones: vamos a mostrar un ejemplo, ya que la mayoría tiene una sintaxis similar

```
[52]: arreglo2d = np.arange(9).reshape(3,3)
      arreglo2d

[52]: array([[0, 1, 2],
             [3, 4, 5],
             [6, 7, 8]])

[53]: arreglo2d.sum()

[53]: 36

[54]: arreglo2d.sum(axis = 0)

[54]: array([ 9, 12, 15])

[55]: arreglo2d.sum(axis = 1)

[55]: array([ 3, 12, 21])
```

¿Qué está haciendo?

Numpy

Funciones de Numpy

Hay muchas funciones: vamos a mostrar un ejemplo, ya que la mayoría tiene una sintaxis similar

```
[52]: arreglo2d = np.arange(9).reshape(3,3)
      arreglo2d

[52]: array([[0, 1, 2],
             [3, 4, 5],
             [6, 7, 8]])

[53]: arreglo2d.sum()

[53]: 36

[54]: arreglo2d.sum(axis = 0)

[54]: array([ 9, 12, 15])

[55]: arreglo2d.sum(axis = 1)

[55]: array([ 3, 12, 21])
```

Es equivalente a:

```
[56]: np.sum(arreglo2d)

[56]: 36

[57]: np.sum(arreglo2d, axis = 0)

[57]: array([ 9, 12, 15])

[58]: np.sum(arreglo2d, axis = 1)

[58]: array([ 3, 12, 21])
```

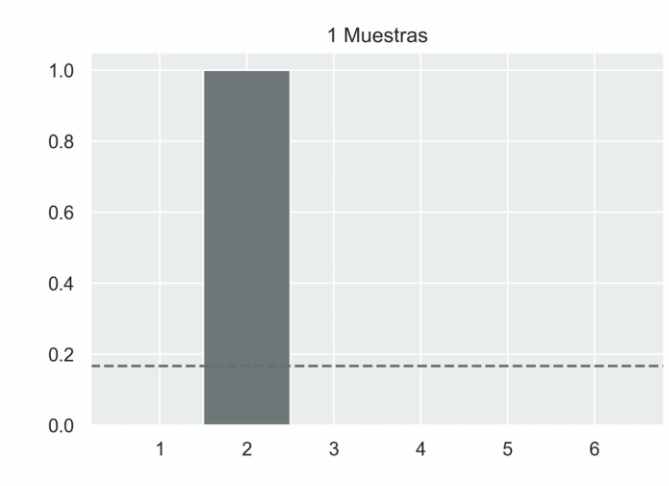
Probabilidad y Estadística

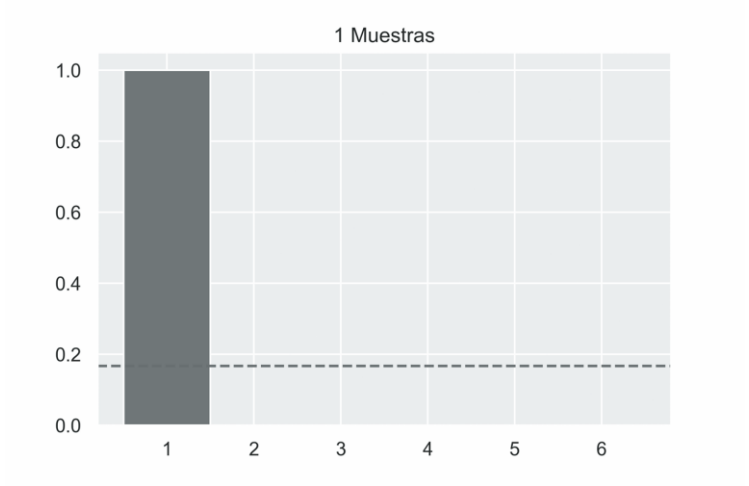
Probabilidad y Estadística

Tenemos dos dados. Suponemos que uno está cargado. ¿Cómo nos damos cuenta cuál?

Probabilidad y Estadística

Tenemos dos dados. Suponemos que uno está cargado. ¿Cómo nos damos cuenta cuál?





Probabilidad y Estadística

PROBABILIDAD	Qué espero ver. Modelos sobre la naturaleza o nuestro problema
ESTADÍSTICA	Lo que vi. **Preguntas**: ¿tiene sentido con mi modelo?¿Qué puedo aprender? (Lo que mido en el laboratorio)
NUEVA PROBABILIDAD	¿Entendí lo que estaba pasando? (Lo que aprendí sobre la naturaleza. ¿Nuevos modelos?)

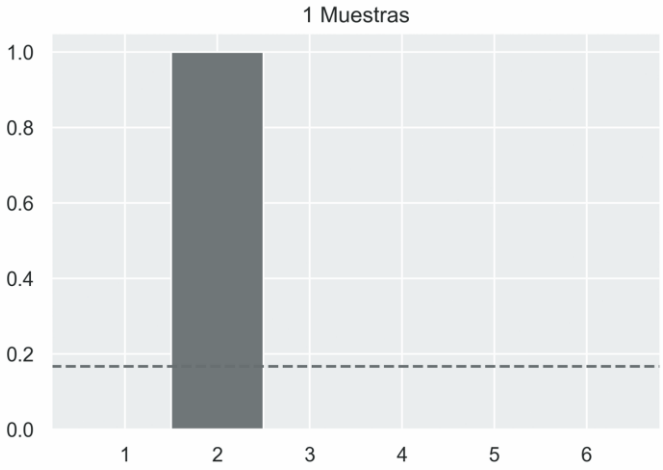
“Todos los modelos están equivocados, pero algunos son útiles.”

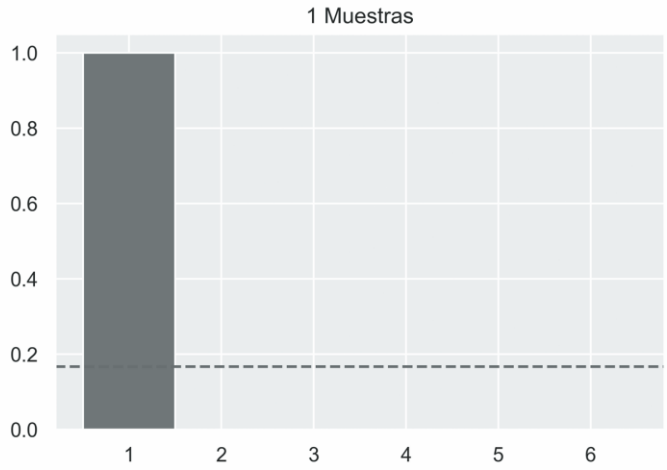
George E. P. Box

Fuente:
Wikipedia

Probabilidad y Estadística

¿Cuál estaba cargado?





Probabilidad: Definición

Frecuentista: si hacemos un experimento muchas (!) veces, la probabilidad está asociada a la **frecuencia** con que ocurre cada posible valor de la variable aleatoria.

Bayesiana: medida de la *confianza* o *certidumbre* de que un suceso ocurra. La mejor medida de la incertidumbre es la probabilidad.

Probabilidad: Variables aleatorias

***X* variable aleatoria.** Posibles resultados de un proceso aleatorio:

X_{moneda}: {cara, ceca}

X_{dado}: {1,2,3,4,5,6}

X_{clima} : {lluvia, no lluvia}

X_{clima} : {cuánto llovió}

$X_{avión}$: {accidente, no-accidente}

Probabilidad: Variables aleatorias

PROBABILIDAD

Variables discretas

- Son aquellas que se *cuentan*
- Pueden estar acotadas o no

Ejemplo: edades (en años), número de hijos, cantidad de dormitorios en una casa, etc.

Probabilidad: Variables aleatorias

PROBABILIDAD

Variables discretas

- Son aquellas que se *cuentan*
- Pueden estar acotadas o no

Ejemplo: edades (en años), número de hijos, cantidad de dormitorios en una casa, etc.

Variables continuas

- Son aquellas que se *miden*
- Pueden estar acotadas o no

Ejemplo: altura de una persona, temperaturas, edades (medidas en tiempo transcurrido desde el nacimiento), etc.

En programación, en Probabilidad y Estadística, Machine Learning y muchas áreas es MUY importante prestarle atención a con qué tipo de variable estamos trabajando.

Probabilidad: Variables aleatorias

PROBABILIDAD

Variables discretas

- Son aquellas que se *cuentan*
- Pueden estar acotadas o no

Ejemplo: edades (en años), número de hijos, cantidad de dormitorios en una casa, etc.

Variables continuas

- Son aquellas que se *miden*
- Pueden estar acotadas o no

Ejemplo: altura de una persona, temperaturas, edades (medidas en tiempo transcurrido desde el nacimiento), etc.

Variables discretas: Distribución

La distribución de probabilidad de una variable aleatoria es una función que asigna a cada suceso definido sobre la variable la probabilidad de que dicho suceso ocurra.

Fuente: Wikipedia

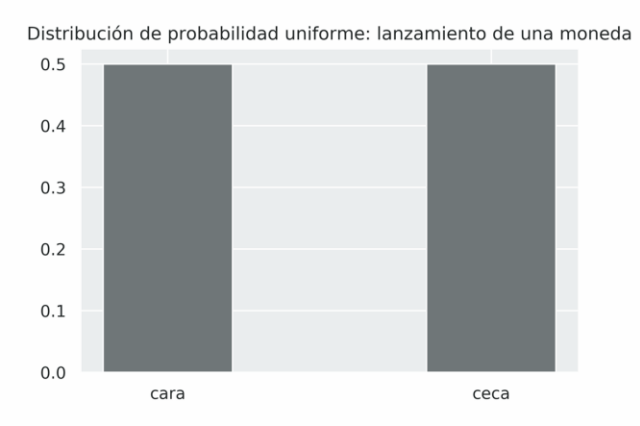
Variables discretas: Distribución uniforme

La distribución de probabilidad **uniforme** asigna la misma probabilidad para todo un rango de valores.

Ejemplos: moneda, dado.

X_{moneda}: {cara, ceca}

P(X = cara, ceca) = 1/2



Variables discretas: Distribución uniforme

La distribución de probabilidad **uniforme** asigna la misma probabilidad para todo un rango de valores.

Ejemplos: moneda, dado.

X_{dado}: {1,2,3,4,5,6}

P(X = 1,2,3,4,5,6) = 1/6

Variables discretas: Distribución binomial

La distribución de probabilidad **binomial** nos ayuda a responder los siguientes tipos de pregunta:

Si tiro 10 veces una moneda, ¿cuál es la probabilidad de obtener tres caras?
¿Y cuatro?

Variables discretas: Distribución binomial

La distribución de probabilidad **binomial** nos ayuda a responder los siguientes tipos de pregunta:

Si tiro 10 veces una moneda, ¿cuál es la probabilidad de obtener tres caras?
¿Y cuatro?

¿Y si en lugar de tirar 10 veces la tiro 100?

Variables discretas: Distribución binomial

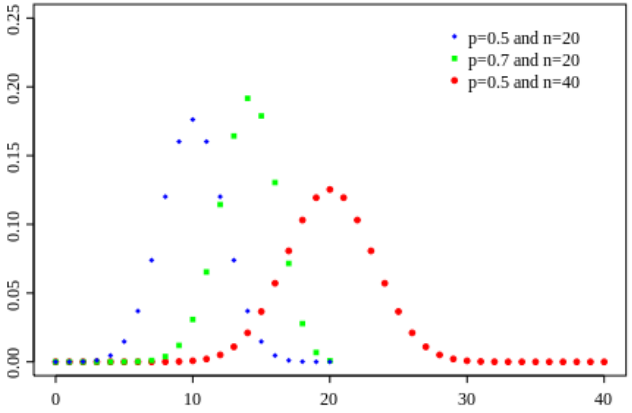
De forma más general, si tiro *n* veces una moneda con probabilidad *p* de sacar cara (o ceca), ¿cuál es la probabilidad de sacar x caras (o cecas)?

Variables discretas: Distribución binomial

$$f(x) = \binom{n}{x} p^x (1-p)^{n-x}, \quad 0 \leq p \leq 1$$

donde $x = \{0, 1, 2, \ldots, n\}$

$$\binom{n}{x} = \frac{n!}{x!(n-x)!}$$ **(combinatorio)**



Fuente: Wikipedia

Variables continuas: Densidad

PROBABILIDAD

Variables discretas

- Son aquellas que se *cuentan*
- Pueden estar acotadas o no

Ejemplo: edades (en años), número de hijos, cantidad de dormitorios en una casa, etc.

Variables continuas

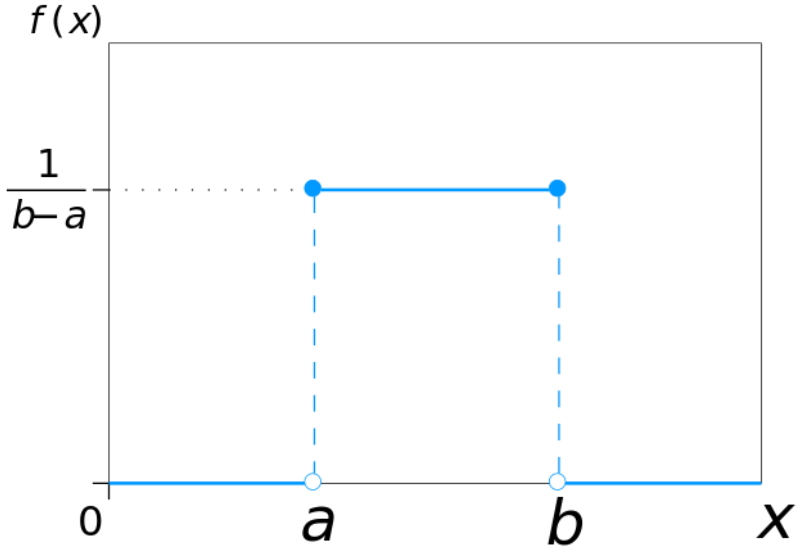
- Son aquellas que se *miden*
- Pueden estar acotadas o no

Ejemplo: altura de una persona, temperaturas, edades (medidas en tiempo transcurrido desde el nacimiento), etc.

Para variables **continuas** se usa el concepto de **densidad** de probabilidad.

Probabilidad: Densidad uniforme

Muy parecida a su versión discreta.



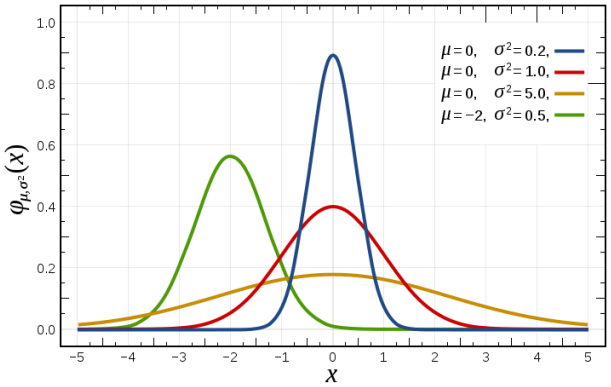
Fuente: Wikipedia

Probabilidad: Densidad normal o Gaussiana

¡La más famosa de las distribuciones!

$$f(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Parámetros:
μ: valor medio
σ: desviación estándar

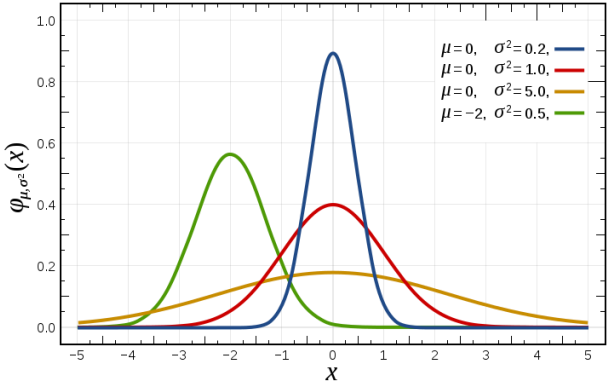


Fuente: Wikipedia

Pero… ¿Para qué sirven esos parámetros?

$$f(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Parámetros:
μ: valor medio
σ: desviación estándar



Si nuestros datos tienen una distribución Gaussiana

Teórico	Calculado
μ	Promedio de los datos
σ	Desviación Estándar Calculada de los datos

Fuente: Wikipedia

Hands-on training

Actividad

DS_Clase_04_Estadistica.ipynb

¡BREAK!

Ph. Credit: Drew Coffmann

Buenas prácticas de un data scientist

Buenas prácticas de un ~~data scientist~~ programador

PEP-20: The Zen of Python

```
Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one -and preferably only one- obvious way to do it.
Although that may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking idea --let's do more of those!
```

Pandas

DATASET

Es el conjunto de datos que utilizaremos en el workflow de data science. Los podemos generar, obtener de terceros o simular.

datasets estructurados	similar a planilla de cálculo. Información pre-procesada. Suelen venir en .txt, .csv, .xlsx, .json, etc.
datasets no estructurados	audio, imágenes, texto en crudo humanos / redes neuronales

DATASET

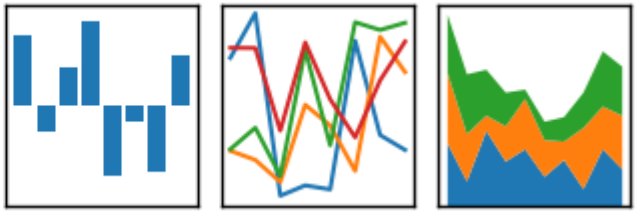
datasets estructurados

similar a planilla de cálculo. Información pre-procesada. Suelen venir en .txt, .csv, .xlsx, .json, etc.

Para trabajar con datasets estructurados (y bueno, más), la librería estándar de Python es:

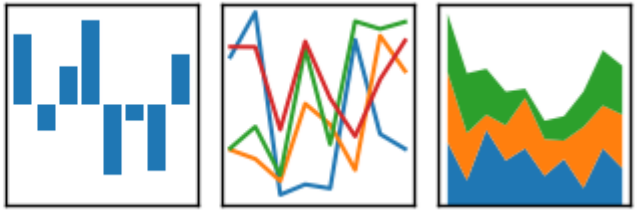
pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



¿Qué aprendieron en los videos de la plataforma?

Pandas: Instalación

1. Activar el ambiente: “*conda activate datascience*”
2. Instalar Pandas: “*conda install pandas*”

ARGENTINA DATASET

División Política, Superficie y Población

IRIS DATASET

Famoso dataset introducido por Ronald Fisher (padre de la estadística) en 1936.

Iris Versicolor **Iris Setosa** **Iris Virginica**

Hands-on training

Hands-on training

DS_Clase_04_Pandas.ipynb

Recursos

Recursos

- Libro de acceso libre (¡en castellano!) de estadística: http://webs.ucm.es/info/Astrof/users/jaz/ESTADISTICA/libro_GCZ2009.pdf

- https://seeing-theory.brown.edu/basic-probability/index.html

- Capítulo 3, “Data Manipulation With Pandas”, de Python Data Science Handbook

Encuesta

¡Queremos escucharte!

ENCUESTA

Para la próxima

1) Ver los videos de la plataforma “Biblioteca: Pandas” y arrancar “Visualización de Datos” si tienen tiempo.

2) Terminar notebooks de hoy y atrasados

1) Googlear: ¿Qué es un cuartil? ¿Y un percentil? ¿Moda y mediana?

ACÁMICA