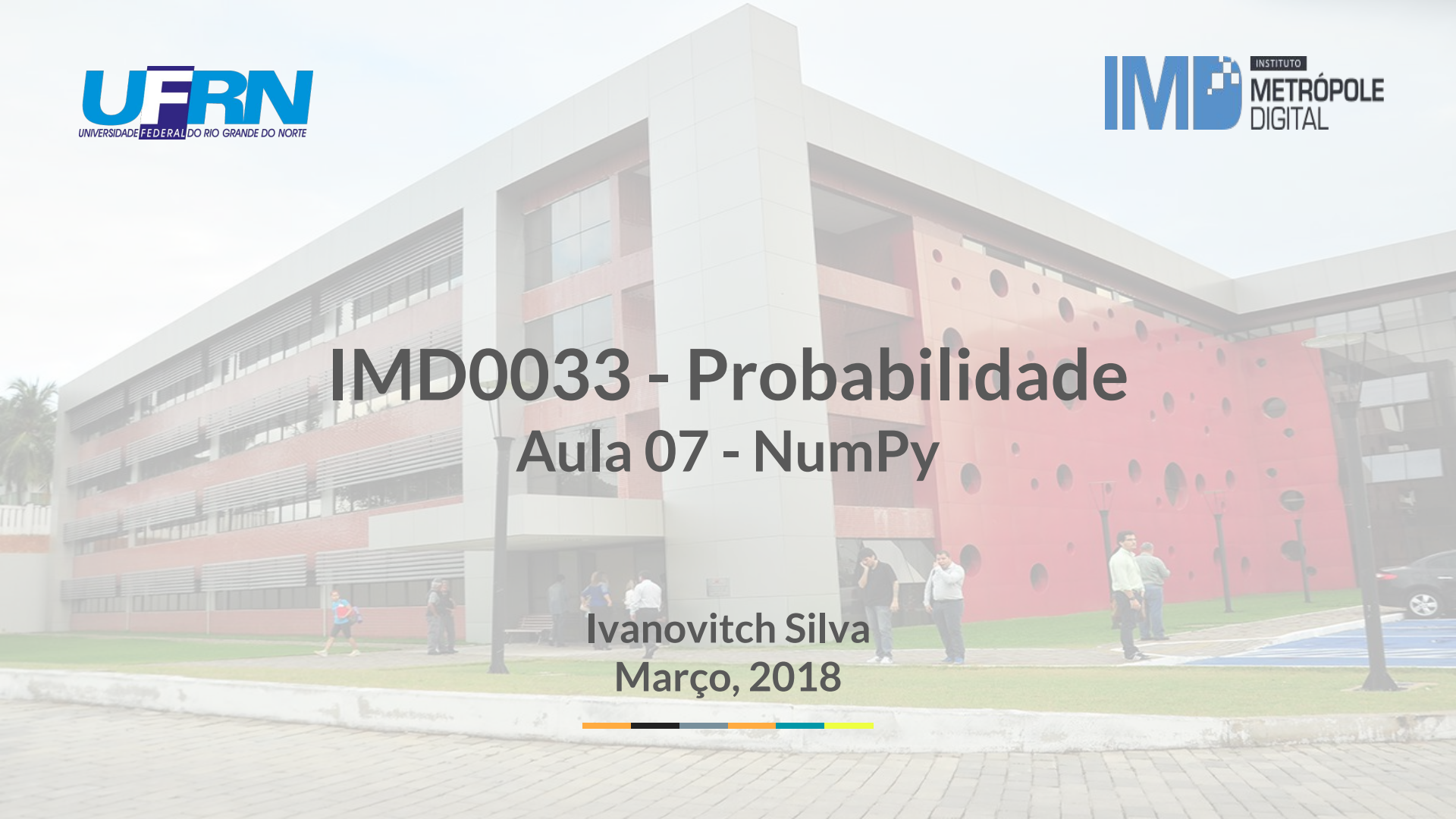


IMD0033 - Probabilidade

Aula 07 - NumPy

Ivanovitch Silva

Março, 2018

Agenda

- Motivação
- Introdução sobre NumPy
- Criação de estruturas multidimensionais
- Leitura de arquivos
- Tipos de dados
- Comparando dados
- Funções agregadas
- Vantagens & Desvantagens

Atualizar o repositório

```
git clone https://github.com/ivanovitchm/imd0033_2018_1.git
```

Ou ....

```
git pull
```

Motivação

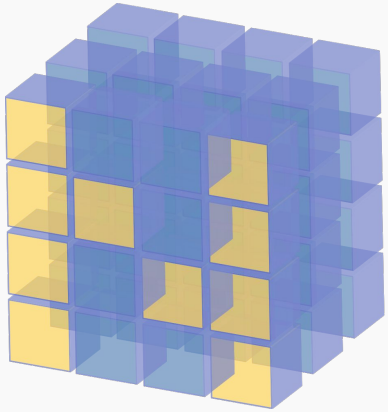
```
areas = ["hallway", 11.25, "kitchen", 18.0, "living room", 20.0]
```

Trabalhar com Listas em Python tem várias vantagens:

- Armazenar tipos diferentes
- Podem diminuir e crescer dinamicamente

Mas.....

- Para suportar essa flexibilidade, Listas consomem mais memória
- Baixo desempenho para lidar com grande volumes de dados
- Não realizam operações matemáticas entre listas



NumPy

A biblioteca NumPy é o núcleo da computação científica em Python. NumPy fornece alto desempenho para operações para estruturas multidimensionais com a flexibilidade de uso encontrada nas Listas.

https://goo.gl/0eWPy6

Criando vetores multidimensionais

1-dimensional array

a

a[0]	0
a[1]	3
a[2]	105
a[3]	30
a[4]	1

2-dimensional array

b

	b[0,0]	b[0,1]	b[0,2]	b[0,3]	b[0,4]
b[0,0]	0	3	105	30	1
b[1,0]	0	3	105	30	1
b[2,0]	0	3	105	30	1

```
import numpy as np
a = np.array([0,3,105,30,1])
b = np.array([[0,3,105,30,1],[0,3,105,30,1],[0,3,105,30,1]])
```

Imprimindo a dimensão das estruturas

```
vector = np.array([1, 2, 3, 4])
print(vector.shape) #output: (4,)
matrix = np.array([[5, 10, 15], [20, 25, 30]])
print(matrix.shape) #output:  (2, 3)
```

Utilizando NumPy para leitura de arquivos

```
import numpy
data = numpy.genfromtxt("data.csv", delimiter=",")
```

World Health Organization

http://apps.who.int/gho/data/view.main.52160

"world_alcohol.csv'

Here's what each column represents:

- Year -- the year the data in the row is for.
- WHO Region -- the region in which the country is located.
- Country -- the country the data is for.
- Beverage Types -- the type of beverage the data is for.
- Display Value -- the number of liters, on average, of the beverage type a citizen of the country drank in the given year.

Analisar o conjunto de dados

world_alcohol

Cabeçalho

```
array([[           nan,            nan,           nan,
                   nan,            nan],
       [  1.98600000e+03,            nan,           nan,
                   nan,   0.00000000e+00],
       [  1.98600000e+03,            nan,           nan,
                   nan,   5.00000000e-01],
       ...,
       [  1.98600000e+03,            nan,           nan,
                   nan,   2.54000000e+00],
       [  1.98700000e+03,            nan,           nan,
                   nan,   0.00000000e+00],
       [  1.98600000e+03,            nan,           nan,
                   nan,   5.15000000e+00]])
```

String

Quando NumPy é incapaz de transformar o tipo do dado para numérico **(nan)**

Lendo o dado corretamente

```
world_alcohol = np.genfromtxt("world_alcohol.csv", delimiter=",", dtype="U75", skip_head
er=1)
[['1986' 'Western Pacific' 'Viet Nam' 'Wine' '0']
 ['1986' 'Americas' 'Uruguay' 'Other' '0.5']
 ['1985' 'Africa' "Cte d'Ivoire" 'Wine' '1.62']
 ...,
 ['1986' 'Europe' 'Switzerland' 'Spirits' '2.54']
 ['1987' 'Western Pacific' 'Papua New Guinea' 'Other' '0']
 ['1986' 'Africa' 'Swaziland' 'Other' '5.15']]
```

Comparando arrays

```
vector = np.array([5, 10, 15, 20])
vector == 10
```

```
array([False,  True, False, False], dtype=bool)
```

```
matrix = np.array([[5, 10, 15],
                   [20, 25, 30],
                   [35, 40, 45]]
                  )
matrix == 25
```

```
array([[False, False, False],
       [False,  True, False],
       [False, False, False]], dtype=bool)
```

Funções agregadas

- sum()
- mean()
- median()
- max()
- min()

Executa funções sobre uma determinada dimensão do array

Funções agregadas

```
vector = np.array([5, 10, 15, 20])
vector.sum()
```

```
50
```

```
matrix = np.array([
                   [5, 10, 15],
                   [20, 25, 30],
                   [35, 40, 45]
                  ])
matrix.sum(axis=1)
```

```
array([ 30,  75, 120])
```

https://goo.gl/0eWPy6

NumPy

The **NumPy** library is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays.

Use the following import convention:

```
>>> import numpy as np
```

NumPy Arrays

Creating Arrays

```
>>> a = np.array([1,2,3])
>>> b = np.array([(1.5,2,3), (4,5,6)], dtype = float)
>>> c = np.array([[(1.5,2,3), (4,5,6)], [(3,2,1), (4,5,6)]],
                 dtype = float)
```

Initial Placeholders

```
>>> np.zeros((3,4))                    Create an array of zeros
>>> np.ones((2,3,4),dtype=np.int16)    Create an array of ones
>>> d = np.arange(10,25,5)             Create an array of evenly
                                       spaced values (step value)
>>> np.linspace(0,2,9)                 Create an array of evenly
                                       spaced values (number of samples)
>>> e = np.full((2,2),7)               Create a constant array
>>> f = np.eye(2)                      Create a 2X2 identity matrix
>>> np.random.random((2,2))            Create an array with random values
>>> np.empty((3,2))                    Create an empty array
```

I/O

Saving & Loading On Disk

```
>>> np.save('my_array', a)
>>> np.savez('array.npz', a, b)
>>> np.load('my_array.npy')
```

Saving & Loading Text Files

```
>>> np.loadtxt("myfile.txt")
>>> np.genfromtxt("my_file.csv", delimiter=',')
>>> np.savetxt("myarray.txt", a, delimiter=" ")
```

Data Types

```
>>> np.int64        Signed 64-bit integer types
>>> np.float32      Standard double-precision floating point
>>> np.complex      Complex numbers represented by 128 floats
>>> np.bool         Boolean type storing TRUE and FALSE values
>>> np.object       Python object type
>>> np.string_      Fixed-length string type
>>> np.unicode_     Fixed-length unicode type
```

Inspecting Your Array

```
>>> a.shape          Array dimensions
>>> len(a)           Length of array
>>> b.ndim           Number of array dimensions
>>> e.size           Number of array elements
>>> b.dtype          Data type of array elements
>>> b.dtype.name     Name of data type
>>> b.astype(int)    Convert an array to a different type
```

Asking For Help

```
>>> np.info(np.ndarray.dtype)
```

Array Mathematics

Arithmetic Operations

```
>>> g = a - b                          Subtraction
  array([[-0.5,  0. ,  0. ],
         [-3. , -3. , -3. ]])
>>> np.subtract(a,b)                   Subtraction
>>> b + a                              Addition
  array([[ 2.5,  4. ,  6. ],
         [ 5. ,  7. ,  9. ]])
>>> np.add(b,a)                        Addition
>>> a / b                              Division
  array([[ 0.66666667,  1.        ,  1.        ],
         [ 0.25      ,  0.4       ,  0.5       ]])
>>> np.divide(a,b)                     Division
>>> a * b                              Multiplication
  array([[  1.5,   4. ,   9. ],
         [  4. ,  10. ,  18. ]])
>>> np.multiply(a,b)                   Multiplication
>>> np.exp(b)                          Exponentiation
>>> np.sqrt(b)                         Square root
>>> np.sin(a)                          Print sines of an array
>>> np.cos(b)                          Element-wise cosine
>>> np.log(a)                          Element-wise natural logarithm
>>> e.dot(f)                           Dot product
  array([[ 7.,  7.],
         [ 7.,  7.]])
```

Comparison

```
>>> a == b                             Element-wise comparison
  array([[False,  True,  True],
         [False, False, False]], dtype=bool)
>>> a < 2                              Element-wise comparison
  array([True, False, False], dtype=bool)
>>> np.array_equal(a, b)               Array-wise comparison
```

Aggregate Functions

```
>>> a.sum()                            Array-wise sum
>>> a.min()                            Array-wise minimum value
>>> b.max(axis=0)                      Maximum value of an array row
>>> b.cumsum(axis=1)                   Cumulative sum of the elements
>>> a.mean()                           Mean
>>> b.median()                         Median
>>> a.corrcoef()                       Correlation coefficient
>>> np.std(b)                          Standard deviation
```

Copying Arrays

```
>>> h = a.view()      Create a view of the array with the same data
>>> np.copy(a)        Create a copy of the array
>>> h = a.copy()      Create a deep copy of the array
```

Sorting Arrays

```
>>> a.sort()          Sort an array
>>> c.sort(axis=0)    Sort the elements of an array's axis
```

Subsetting, Slicing, Indexing

Also see Lists

Subsetting

```
>>> a[2]                               Select the element at the 2nd index
  3
>>> b[1,2]                             Select the element at row 0 column 2
  6.0                                  (equivalent to b[1][2])
```

Slicing

```
>>> a[0:2]                             Select items at index 0 and 1
  array([1, 2])
>>> b[0:2,1]                           Select items at rows 0 and 1 in column 1
  array([ 2.,  5.])
>>> b[:1]                              Select all items at row 0
  array([[1.5, 2., 3.]])               (equivalent to b[0:1, :])
>>> c[1,...]                           Same as [1,:,:]
  array([[[ 3.,  2.,  1.],
          [ 4.,  5.,  6.]]])
>>> a[ : :-1]                          Reversed array a
  array([3, 2, 1])
```

Boolean Indexing

```
>>> a[a<2]                             Select elements from a less than 2
  array([1])
```

Fancy Indexing

```
>>> b[[1, 0, 1, 0],[0, 1, 2, 0]]      Select elements (1,0),(0,1),(1,2) and (0,0)
  array([ 4. ,  2. ,  6. ,  1.5])
>>> b[[1, 0, 1, 0]][:,[0,1,2,0]]      Select a subset of the matrix's rows
  array([[ 4. ,  5. ,  6. ,  4. ],     and columns
         [ 1.5,  2. ,  3. ,  1.5],
         [ 4. ,  5. ,  6. ,  4. ],
         [ 1.5,  2. ,  3. ,  1.5]])
```

Array Manipulation

Transposing Array

```
>>> i = np.transpose(b)                Permute array dimensions
>>> i.T                                Permute array dimensions
```

Changing Array Shape

```
>>> b.ravel()                          Flatten the array
>>> g.reshape(3,-2)                    Reshape, but don't change data
```

Adding/Removing Elements

```
>>> h.resize((2,6))                    Return a new array with shape (2,6)
>>> np.append(h,g)                     Append items to an array
>>> np.insert(a, 1, 5)                 Insert items in an array
>>> np.delete(a,[1])                   Delete items from an array
```

Combining Arrays

```
>>> np.concatenate((a,d),axis=0)       Concatenate arrays
  array([ 1,  2,  3, 10, 15, 20])
>>> np.vstack((a,b))                   Stack arrays vertically (row-wise)
  array([[ 1. ,  2. ,  3. ],
         [ 1.5,  2. ,  3. ],
         [ 4. ,  5. ,  6. ]])
>>> np.r_[e,f]                         Stack arrays vertically (row-wise)
>>> np.hstack((e,f))                   Stack arrays horizontally (column-wise)
  array([[ 7.,  7.,  1.,  0.],
         [ 7.,  7.,  0.,  1.]])
>>> np.column_stack((a,d))             Create stacked column-wise arrays
  array([[ 1, 10],
         [ 2, 15],
         [ 3, 20]])
>>> np.c_[a,d]                         Create stacked column-wise arrays
```

Splitting Arrays

```
>>> np.hsplit(a,3)                     Split the array horizontally at the 3rd
  [array([1]),array([2]),array([3])]   index
>>> np.vsplit(c,2)                     Split the array vertically at the 2nd index
[array([[[ 1.5,  2. ,  1. ],
         [ 4. ,  5. ,  6. ]]]),
 array([[[ 3.,  2.,  3.],
         [ 4.,  5.,  6.]]])]
```

NumPy - Pros & Contra

Pros

- Fácil e flexível para computação científica
- Acesso rápido e flexível ao dado (slicing, indexing)
- Conversão de tipos rapidamente

Contra

- Todos os dados devem ser do mesmo tipo
- Colunas e linhas são acessadas por números


Desafio (WHO)?
World Health Organization