Algorítmica y Programación

Enero - Mayo 2020

Dr. Iván S. Razo Zapata
(ivan.razo@itam.mx)

ITAM

Pandas intro

DataFrames

A DataFrame represents a rectangular table of data (spreadsheet-like) and contains an **ordered** collection of columns, each of which can be a different value type (numeric, string, boolean, etc.)

	year	state	pop	debt
one	2000	Ohio	1.5	16.5
two	2001	Ohio	1.7	16.5
three	2002	Ohio	3.6	16.5
four	2001	Nevada	2.4	16.5
five	2002	Nevada	2.9	16.5
six	2003	Nevada	3.2	16.5

Dataframes - creación en base a un diccionario

```python
import pandas as pd

diccionario1 = {'Alcaldia'  : ['Álvaro Obregón', 'Azcapotzalco', 'Benito Juárez'],
                'Poblacion' : [727034, 414711, 385439],
                'Superficie': [96.17, 33.66, 26.63]}

dataFrame1 = pd.DataFrame(diccionario1)

```

```
In [8]: diccionario1
Out[8]:
{'Alcaldia': ['Álvaro Obregón', 'Azcapotzalco', 'Benito Juárez'],
 'Poblacion': [727034, 414711, 385439],
 'Superficie': [96.17, 33.66, 26.63]}

In [9]: dataFrame1
Out[9]:
         Alcaldia  Poblacion  Superficie
0  Álvaro Obregón     727034       96.17
1    Azcapotzalco     414711       33.66
2   Benito Juárez     385439       26.63

In [10]: dataFrame1.index
Out[10]: RangeIndex(start=0, stop=3, step=1)

In [11]: dataFrame1.values
Out[11]:
array([['Álvaro Obregón', 727034, 96.17],
       ['Azcapotzalco', 414711, 33.66],
       ['Benito Juárez', 385439, 26.63]], dtype=object)
```

Dataframes - creación en base a un diccionario

Indicando columnas e índices

```
In [12]: dataFrame2 = pd.DataFrame(diccionario1, columns=['Superficie','Poblacion','Alcaldia'])

In [13]: dataFrame2
Out[13]:
   Superficie  Poblacion        Alcaldia
0       96.17     727034  Álvaro Obregón
1       33.66     414711    Azcapotzalco
2       26.63     385439   Benito Juárez

In [14]: dataFrame3 = pd.DataFrame(diccionario1, columns=['Superficie','Poblacion','Alcaldia'],
    ...: index=['Uno','Dos','Tres'])

In [15]: dataFrame3
Out[15]:
      Superficie  Poblacion        Alcaldia
Uno        96.17     727034  Álvaro Obregón
Dos        33.66     414711    Azcapotzalco
Tres       26.63     385439   Benito Juárez
```

Dataframes – propiedades básicas

```
In [15]: dataFrame3
Out[15]:
      Superficie  Poblacion        Alcaldia
Uno        96.17     727034  Álvaro Obregón
Dos        33.66     414711    Azcapotzalco
Tres       26.63     385439   Benito Juárez

In [16]: dataFrame3.columns
Out[16]: Index(['Superficie', 'Poblacion', 'Alcaldia'], dtype='object')

In [17]: dataFrame3.index
Out[17]: Index(['Uno', 'Dos', 'Tres'], dtype='object')

In [18]: dataFrame3.values
Out[18]:
array([[96.17, 727034, 'Álvaro Obregón'],
       [33.66, 414711, 'Azcapotzalco'],
       [26.63, 385439, 'Benito Juárez']], dtype=object)
```

Dataframes – propiedades básicas

Accediendo a columnas -> Serie

```
In [20]: dataFrame3['Superficie']
Out[20]:
Uno     96.17
Dos     33.66
Tres    26.63
Name: Superficie, dtype: float64

In [21]: dataFrame3['Alcaldia']
Out[21]:
Uno     Álvaro Obregón
Dos       Azcapotzalco
Tres     Benito Juárez
Name: Alcaldia, dtype: object

In [22]: dataFrame3['Poblacion']
Out[22]:
Uno     727034
Dos     414711
Tres    385439
Name: Poblacion, dtype: int64
```

```
In [23]: dataFrame3.Superficie
Out[23]:
Uno     96.17
Dos     33.66
Tres    26.63
Name: Superficie, dtype: float64

In [24]: dataFrame3.Alcaldia
Out[24]:
Uno     Álvaro Obregón
Dos       Azcapotzalco
Tres     Benito Juárez
Name: Alcaldia, dtype: object

In [25]: dataFrame3.Poblacion
Out[25]:
Uno     727034
Dos     414711
Tres    385439
Name: Poblacion, dtype: int64

In [26]:
```

Dataframes – propiedades básicas

Accediendo a renglones -> Serie

```
In [27]: dataFrame3.loc['Uno']
Out[27]:
Superficie                96.17
Poblacion                727034
Alcaldia       Álvaro Obregón
Name: Uno, dtype: object

In [28]: dataFrame3.iloc[0]
Out[28]:
Superficie                96.17
Poblacion                727034
Alcaldia       Álvaro Obregón
Name: Uno, dtype: object
```

Dataframes – propiedades básicas

Modificando contenido

```
In [72]: dataFrame3
Out[72]:
      Superficie  Poblacion        Alcaldia
Uno        96.17     727034  Álvaro Obregón
Dos        33.66     414711    Azcapotzalco
Tres       26.63     385439   Benito Juárez

In [73]: sup = dataFrame3['Superficie']

In [74]: sup
Out[74]:
Uno     96.17
Dos     33.66
Tres    26.63
Name: Superficie, dtype: float64

In [75]: dataFrame3['Superficie'] = 567

In [76]: dataFrame3
Out[76]:
      Superficie  Poblacion        Alcaldia
Uno          567     727034  Álvaro Obregón
Dos          567     414711    Azcapotzalco
Tres         567     385439   Benito Juárez

In [77]: dataFrame3['Superficie'] = sup

In [78]: dataFrame3
Out[78]:
      Superficie  Poblacion        Alcaldia
Uno        96.17     727034  Álvaro Obregón
Dos        33.66     414711    Azcapotzalco
Tres       26.63     385439   Benito Juárez
```

Modificando con una lista y range

```
In [79]: dataFrame3['tmp'] = [1,2,3]

In [80]: dataFrame3
Out[80]:
      Superficie  Poblacion        Alcaldia  tmp
Uno        96.17     727034  Álvaro Obregón    1
Dos        33.66     414711    Azcapotzalco    2
Tres       26.63     385439   Benito Juárez    3

In [81]: dataFrame3['tmp'] = range(5,8)

In [82]: dataFrame3
Out[82]:
      Superficie  Poblacion        Alcaldia  tmp
Uno        96.17     727034  Álvaro Obregón    5
Dos        33.66     414711    Azcapotzalco    6
Tres       26.63     385439   Benito Juárez    7

In [83]:
```

Dataframes – propiedades básicas

Modificando contenido con una serie y uso de **del**

```
In [87]: valores = pd.Series([5.6, 8.9], index=['Uno', 'Tres'])

In [88]: dataFrame3['tmp'] = valores

In [89]: dataFrame3
Out[89]:
      Superficie  Poblacion        Alcaldia  tmp
Uno        96.17     727034  Álvaro Obregón  5.6
Dos        33.66     414711    Azcapotzalco  NaN
Tres       26.63     385439   Benito Juárez  8.9

In [90]: del dataFrame3['tmp']

In [91]: dataFrame3
Out[91]:
      Superficie  Poblacion        Alcaldia
Uno        96.17     727034  Álvaro Obregón
Dos        33.66     414711    Azcapotzalco
Tres       26.63     385439   Benito Juárez
```

Dataframes – propiedades básicas

Modificando contenido en renglones

```
In [95]: dataFrame3.loc['Uno'] = [1,2,3]

In [96]: dataFrame3
Out[96]:
      Superficie  Poblacion       Alcaldia
Uno         1.00          2              3
Dos        33.66     414711   Azcapotzalco
Tres       26.63     385439  Benito Juárez

In [97]: dataFrame3.iloc[0] = [4,5,6]

In [98]: dataFrame3
Out[98]:
      Superficie  Poblacion       Alcaldia
Uno         4.00          5              6
Dos        33.66     414711   Azcapotzalco
Tres       26.63     385439  Benito Juárez

In [99]: dataFrame3.iloc[0] = ['ABC',8,9]

In [100]: dataFrame3
Out[100]:
     Superficie  Poblacion       Alcaldia
Uno         ABC          8              9
Dos       33.66     414711   Azcapotzalco
Tres      26.63     385439  Benito Juárez
```

Dataframes – propiedades básicas

Invirtiendo índices y columnas

```
In [122]: dataFrame3
Out[122]:
     Superficie  Poblacion       Alcaldia
Uno         ABC        8.0              9
Dos       33.66   414711.0   Azcapotzalco
Tres      26.63   385439.0  Benito Juárez

In [123]: dataFrame3.T
Out[123]:
            Uno           Dos           Tres
Superficie  ABC         33.66          26.63
Poblacion     8        414711         385439
Alcaldia      9  Azcapotzalco  Benito Juárez

In [124]: dataFrame4 = dataFrame3.T

In [125]: dataFrame4
Out[125]:
            Uno           Dos           Tres
Superficie  ABC         33.66          26.63
Poblacion     8        414711         385439
Alcaldia      9  Azcapotzalco  Benito Juárez

In [126]: valores2 = pd.Series([5.6, 8.9], index=['Alcaldia', 'Superficie'])

In [127]: dataFrame4['Uno'] = valores2

In [128]: dataFrame4
Out[128]:
            Uno           Dos           Tres
Superficie  8.9         33.66          26.63
Poblacion   NaN        414711         385439
Alcaldia    5.6  Azcapotzalco  Benito Juárez
```

Dataframes – Análisis básico

```python
import pandas as pd

dataCO2 = pd.read_csv("DataSets/CO2.csv", encoding='latin1')
dataCO2.set_index('Country', inplace=True)

```

Dataframes – Análisis básico

```
In [4]: dataCO2.sum()
Out[4]:
1990    26467.677
2005    33283.021
2017    39410.623
dtype: float64

In [5]: dataCO2.mean()
Out[5]:
1990    126.639603
2005    159.248904
2017    188.567574
dtype: float64

In [6]: dataCO2.max()
Out[6]:
1990     5085.90
2005     6263.06
2017    10877.22
dtype: float64

In [7]: dataCO2.min()
Out[7]:
1990    0.001
2005    0.002
2017    0.002
dtype: float64
```

```
In [12]: dataCO2.idxmax()
Out[12]:
1990     United States
2005             China
2017             China
dtype: object

In [13]: dataCO2.idxmin()
Out[13]:
1990     Faroe Islands
2005     Faroe Islands
2017     Faroe Islands
dtype: object

In [14]: dataCO2.head()
Out[14]:
                 1990    2005     2017
Country
 Afghanistan    2.546   1.063   11.422
 Albania        6.583   4.196    5.026
 Algeria       65.677  98.197  159.929
 Angola         5.851  15.975   30.876
 Anguilla       0.006   0.014    0.028

In [15]: dataCO2.tail()
Out[15]:
                   1990    2005     2017
Country
 Vietnam          20.182  99.231  218.729
 Western Sahara    0.144   0.227    0.276
 Yemen             6.887  21.768   12.503
 Zambia            2.955   2.457    4.967
 Zimbabwe         17.178  11.388   12.087
```

Dataframes – Análisis básico

```
In [16]: dataCO2.describe()
Out[16]:
                1990          2005          2017
count    209.000000    209.000000    209.000000
mean     126.639603    159.248904    188.567574
std      531.019059    686.057879    892.167195
min        0.001000      0.002000      0.002000
25%        0.644000      1.063000      1.572000
50%        5.474000      7.204000     10.562000
75%       51.919000     59.747000     62.487000
max     5085.900000   6263.060000  10877.220000
```

MovieLens Básico

https://grouplens.org/datasets/movielens/

grouplens.org/datasets/movielens/

recommended for education and development

MovieLens Latest Datasets

These datasets will change over time, and are not appropriate for reporting research results. We will keep the download links stable for automated downloads. We will not archive or make available previously released versions.

Small: 100,000 ratings and 3,600 tag applications applied to 9,000 movies by 600 users. Last updated 9/2018.

- README.html
- ml-latest-small.zip (size: 1 MB)

Full: 27,000,000 ratings and 1,100,000 tag applications applied to 58,000 movies by 280,000 users. Includes tag genome data with 14 million relevance scores across 1,100 tags. Last updated 9/2018.

- README.html
- ml-latest.zip (size: 265 MB)

Permalink: https://grouplens.org/datasets/movielens/latest/

MovieLens

No seguro | files.grouplens.org/datasets/movielens/ml-latest-small-README.html

Summary

This dataset (ml-latest-small) describes 5-star rating and free-text tagging activity from MovieLens, a movie recommendation service. It contains 100836 ratings and 3683 tag applications across 9742 movies. These data were created by 610 users between March 29, 1996 and September 24, 2018. This dataset was generated on September 26, 2018.

Users were selected at random for inclusion. All selected users had rated at least 20 movies. No demographic information is included. Each user is represented by an id, and no other information is provided.

The data are contained in the files `links.csv`, `movies.csv`, `ratings.csv` and `tags.csv`. More details about the contents and use of all these files follows.

This is a *development* dataset. As such, it may change over time and is not an appropriate dataset for shared research results. See available *benchmark* datasets if that is your intent.

This and other GroupLens data sets are publicly available for download at http://grouplens.org/datasets/.

Usage License

Neither the University of Minnesota nor any of the researchers involved can guarantee the correctness of the data, its suitability for any particular purpose, or the validity of results based on the use of the data set. The data set may be used for any research purposes under the following conditions:

- The user may not state or imply any endorsement from the University of Minnesota or the GroupLens Research Group.
- The user must acknowledge the use of the data set in publications resulting from the use of the data set (see below for citation information).
- The user may redistribute the data set, including transformations, so long as it is distributed under these same license conditions.
- The user may not use this information for any commercial or revenue-bearing purposes without first obtaining permission from a faculty member of the GroupLens Research Project at the University of Minnesota.
- The executable software scripts are provided "as is" without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the quality and performance of them is with you. Should the program prove defective, you assume the cost of all necessary servicing, repair or correction.

In no event shall the University of Minnesota, its affiliates or employees be liable to you for any damages arising out of the use or inability to use these programs (including but not limited to loss of data or data being rendered inaccurate).

If you have any further questions or comments, please email grouplens-info@umn.edu

Citation

To acknowledge use of the dataset in publications, please cite the following paper:

> F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4: 19:1–19:19. https://doi.org/10.1145/2827872

Further Information About GroupLens

GroupLens is a research group in the Department of Computer Science and Engineering at the University of Minnesota. Since its inception in 1992, GroupLens's research projects have explored a variety of fields including:

- recommender systems
- online communities
- mobile and ubiquitious technologies
- digital libraries
- local geographic information systems

MovieLens

```python

import pandas as pd

dataRatings = pd.read_csv("DataSets/ml-latest-small/ratings.csv")
dataRatings.set_index('userId', inplace=True)

dataMovies = pd.read_csv("DataSets/ml-latest-small/movies.csv")
dataMovies.set_index('movieId', inplace=True)

dataLinks = pd.read_csv("DataSets/ml-latest-small/links.csv")
dataLinks.set_index('movieId', inplace=True)

dataTags = pd.read_csv("DataSets/ml-latest-small/tags.csv")


```

MovieLens

```
In [8]: dataRatings.head()
Out[8]:
        movieId  rating  timestamp
userId
1             1     4.0  964982703
1             3     4.0  964981247
1             6     4.0  964982224
1            47     5.0  964983815
1            50     5.0  964982931

In [9]: dataMovies.head()
Out[9]:
                                    title                                       genres
movieId
1                        Toy Story (1995)  Adventure|Animation|Children|Comedy|Fantasy
2                          Jumanji (1995)                  Adventure|Children|Fantasy
3                  Grumpier Old Men (1995)                              Comedy|Romance
4                 Waiting to Exhale (1995)                        Comedy|Drama|Romance
5        Father of the Bride Part II (1995)                                      Comedy

In [10]: dataLinks.head()
Out[10]:
         imdbId   tmdbId
movieId
1        114709    862.0
2        113497   8844.0
3        113228  15602.0
4        114885  31357.0
5        113041  11862.0

In [11]: dataTags.head()
Out[11]:
   userId  movieId              tag   timestamp
0       2    60756            funny  1445714994
1       2    60756  Highly quotable  1445714996
2       2    60756     will ferrell  1445714992
3       2    89774     Boxing story  1445715207
4       2    89774              MMA  1445715200
```

MovieLens

```
In [4]: dataRatings.describe()
Out[4]:
             movieId         rating     timestamp
count  100836.000000  100836.000000  1.008360e+05
mean    19435.295718       3.501557  1.205946e+09
std     35530.987199       1.042529  2.162610e+08
min         1.000000       0.500000  8.281246e+08
25%      1199.000000       3.000000  1.019124e+09
50%      2991.000000       3.500000  1.186087e+09
75%      8122.000000       4.000000  1.435994e+09
max    193609.000000       5.000000  1.537799e+09
```

```
In [5]: dataMovies.describe()
Out[5]:
              title genres
count          9742   9742
unique         9737    951
top     Eros (2004)  Drama
freq              2   1053
```

```
In [6]: dataLinks.describe()
Out[6]:
             imdbId         tmdbId
count  9.742000e+03    9734.000000
mean   6.771839e+05   55162.123793
std    1.107228e+06   93653.481487
min    4.170000e+02       2.000000
25%    9.518075e+04    9665.500000
50%    1.672605e+05   16529.000000
75%    8.055685e+05   44205.750000
max    8.391976e+06  525662.000000
```

```
In [7]: dataTags.describe()
Out[7]:
            userId        movieId     timestamp
count  3683.000000    3683.000000  3.683000e+03
mean    431.149335   27252.013576  1.320032e+09
std     158.472553   43490.558803  1.721025e+08
min       2.000000       1.000000  1.137179e+09
25%     424.000000    1262.500000  1.137521e+09
50%     474.000000    4454.000000  1.269833e+09
75%     477.000000   39263.000000  1.498457e+09
max     610.000000  193565.000000  1.537099e+09
```

MovieLens – cargando 25M

Probar abrir ratings.csv en Excel

```python
import pandas as pd

dataFrame1 = pd.read_csv("DataSets/ml-25m/ratings.csv")
dataFrame1.set_index('userId', inplace=True)

```

```
In [8]: dataFrame1.size
Out[8]: 75000285

In [9]: dataFrame1.shape
Out[9]: (25000095, 3)

In [10]: dataFrame1.head()
Out[10]:
        movieId  rating   timestamp
userId
1           296     5.0  1147880044
1           306     3.5  1147868817
1           307     5.0  1147868828
1           665     5.0  1147878820
1           899     3.5  1147868510

In [11]: dataFrame1.tail()
Out[11]:
        movieId  rating   timestamp
userId
162541    50872     4.5  1240953372
162541    55768     2.5  1240951998
162541    56176     2.0  1240950697
162541    58559     4.0  1240953434
162541    63876     5.0  1240952515
```