2. Ordenación de datos

June 19, 2022

1 Transformación de datos

1.0.1 Ordenación de datos

Carga de librerías

```
[2]: import pandas as pd
```

Importado de datos

```
[3]: df = pd.read_csv("nycflights.csv")
     #df.info()
     df.head()
```

```
[3]:    year  month  day  dep_time  dep_delay  arr_time  arr_delay carrier tailnum  \
     0  2013      6   30       940         15      1216         -4      VX  N626VA
     1  2013      5    7      1657         -3      2104         10      DL  N3760C
     2  2013     12    8       859         -1      1238         11      DL  N712TW
     3  2013      5   14      1841         -4      2122        -34      DL  N914DL
     4  2013      7   21      1102         -3      1230         -8      9E  N823AY

        flight origin dest  air_time  distance  hour  minute
     0     407    JFK  LAX       313      2475     9      40
     1     329    JFK  SJU       216      1598    16      57
     2     422    JFK  LAX       376      2475     8      59
     3    2391    JFK  TPA       135      1005    18      41
     4    3652    LGA  ORF        50       296    11       2
```

Uso de la función sorted para ordenar una columna Sort by the values along either axis. PARAMETERS: **by**: str or list of str Name or list of names to sort by. - if axis is 0 or 'index' then by may contain index levels and/or column labels. - if axis is 1 or 'columns' then by may contain column levels and/or index labels.

axis: {0 or 'index', 1 or 'columns'}, default 0. - Axis to be sorted.

ascending: bool or list of bool, default True - Sort ascending vs. descending. Specify list for multiple sort orders. If this is a list of bools, must match the length of the by.

inplace: bool, default False - If True, perform operation in-place.

kind: {'quicksort', 'mergesort', 'heapsort', 'stable'}, default 'quicksort' - Choice of sorting algorithm. See also numpy.sort() for more information. mergesort and stable are the only stable algorithms. For DataFrames, this option is only applied when sorting on a single column or label.

na_position: {'first', 'last'}, default 'last' - Puts NaNs at the beginning if first; last puts NaNs at the end.

ignore_index: bool, default False - If True, the resulting axis will be labeled 0, 1, …, n - 1.

[7]:
```
# Orden ascendente
df1 = df.sort_values(by = ['air_time'], inplace = False)
df1.head()
```

[7]:
```
        year  month  day  dep_time  dep_delay  arr_time  arr_delay carrier  \
21986   2013      9   23       718         -7       759        -23      EV
8439    2013      2   11      1313         -2      1422         11      EV
12921   2013      3   21      2138         -7      2247          1      EV
962     2013      6   11      1255         -2      1335        -18      EV
8650    2013      3   17      1324         -4      1425          1      EV

       tailnum  flight origin dest  air_time  distance  hour  minute
21986   N21197    6062    EWR  BDL        22       116     7      18
8439    N14916    4368    EWR  BDL        22       116    13      13
12921   N29917    4276    EWR  BDL        22       116    21      38
962     N12922    5968    EWR  BDL        22       116    12      55
8650    N18557    4118    EWR  BDL        22       116    13      24
```

[8]:
```
# Orden descendente
df1 = df.sort_values(by = ['air_time'], inplace = False, ascending = False)
df1.head()
```

[8]:
```
        year  month  day  dep_time  dep_delay  arr_time  arr_delay carrier  \
20864   2013      3   15      1001          1      1551         21      HA
32210   2013     11   10       957         -3      1625         30      HA
15346   2013      4   10      1431         61      2008        113      UA
15889   2013      1   20      1423         42      2043         62      UA
14152   2013      3   23      1000          0      1529         -1      HA

       tailnum  flight origin dest  air_time  distance  hour  minute
20864   N388HA      51    JFK  HNL       686      4983    10       1
32210   N393HA      51    JFK  HNL       667      4983     9      57
15346   N76065      15    EWR  HNL       666      4963    14      31
15889   N77066      15    EWR  HNL       659      4963    14      23
14152   N380HA      51    JFK  HNL       655      4983    10       0
```

Ordenar valores según múltiples columnas Considerar que el orden en el que se describan las columnas dentro del argumento 'by' determinarán la forma de romper el empate en la operación.

```
[11]: # Ordena por año, mes, día y hora de salida
      df1 = df.sort_values(by = ['year','month','day','dep_time'], inplace = False,␣
       ↪ascending = True)
      df1.head()
```

```
[11]:        year  month  day  dep_time  dep_delay  arr_time  arr_delay carrier  \
      20035  2013      1    1       606         -4       837         -8      DL
      14688  2013      1    1       613          3       925          4      B6
      10841  2013      1    1       635          0      1028         48      AA
      21880  2013      1    1       637         -8       930         -5      B6
      8367   2013      1    1       658         -2      1027          2      VX

            tailnum  flight origin dest  air_time  distance  hour  minute
      20035  N3739P    1743    JFK  ATL       128       760     6       6
      14688  N635JB     135    JFK  RSW       175      1074     6      13
      10841  N3GKAA     711    LGA  DFW       248      1389     6      35
      21880  N709JB     389    LGA  MCO       144       950     6      37
      8367   N627VA     399    JFK  LAX       361      2475     6      58
```

Ordenar la posición de columnas alfabéticamente

```
[12]: df1 = df.sort_index(axis=1)
      df1.head()
```

```
[12]:    air_time  arr_delay  arr_time carrier  day  dep_delay  dep_time dest  \
      0       313         -4      1216      VX   30         15       940  LAX
      1       216         10      2104      DL    7         -3      1657  SJU
      2       376         11      1238      DL    8         -1       859  LAX
      3       135        -34      2122      DL   14         -4      1841  TPA
      4        50         -8      1230      9E   21         -3      1102  ORF

         distance  flight  hour  minute  month origin tailnum  year
      0      2475     407     9      40      6    JFK  N626VA  2013
      1      1598     329    16      57      5    JFK  N3760C  2013
      2      2475     422     8      59     12    JFK  N712TW  2013
      3      1005    2391    18      41      5    JFK  N914DL  2013
      4       296    3652    11       2      7    LGA  N823AY  2013
```

```
[13]: df1 = df.sort_index(axis=1, ascending=False, inplace = False)
      df1.head()
```

```
[13]:    year tailnum origin  month  minute  hour  flight  distance dest  dep_time  \
      0  2013  N626VA    JFK      6      40     9     407      2475  LAX       940
      1  2013  N3760C    JFK      5      57    16     329      1598  SJU      1657
      2  2013  N712TW    JFK     12      59     8     422      2475  LAX       859
      3  2013  N914DL    JFK      5      41    18    2391      1005  TPA      1841
      4  2013  N823AY    LGA      7       2    11    3652       296  ORF      1102
```

```
   dep_delay  day carrier  arr_time  arr_delay  air_time
0         15   30      VX      1216         -4       313
1         -3    7      DL      2104         10       216
2         -1    8      DL      1238         11       376
3         -4   14      DL      2122        -34       135
4         -3   21      9E      1230         -8        50
```

[14]:
```
df1 = df.sort_index(axis=1, ascending=True, inplace = False)
df1.head()
```

[14]:
```
   air_time  arr_delay  arr_time carrier  day  dep_delay  dep_time dest  \
0       313         -4      1216      VX   30         15       940  LAX
1       216         10      2104      DL    7         -3      1657  SJU
2       376         11      1238      DL    8         -1       859  LAX
3       135        -34      2122      DL   14         -4      1841  TPA
4        50         -8      1230      9E   21         -3      1102  ORF

   distance  flight  hour  minute  month origin tailnum  year
0      2475     407     9      40      6    JFK  N626VA  2013
1      1598     329    16      57      5    JFK  N3760C  2013
2      2475     422     8      59     12    JFK  N712TW  2013
3      1005    2391    18      41      5    JFK  N914DL  2013
4       296    3652    11       2      7    LGA  N823AY  2013
```

Ordenar la posición de columnas manualmente

[26]:
```
df = pd.DataFrame({'Name'       : ['Carlos', 'Andrés', np.nan, 'Santiago',␣
 ↪"Fernand0", "Marcelo"],
                   'Age'        : [23, 24, 24, 25, None, 27],
                   'University' : ['AA', pd.NA, 'BB', None, 'CC','EE']})
```

[27]:
```
df1 = df[['University', 'Name', 'Age']]
df1.head(6)
```

[27]:
```
  University      Name   Age
0         AA    Carlos  23.0
1       <NA>    Andrés  24.0
2         BB       NaN  24.0
3       None  Santiago  25.0
4         CC  Fernand0   NaN
5         EE   Marcelo  27.0
```

[38]:
```
# Ordenar columnas manualmente
col_var = df.columns.tolist()
my_order = [1,2,0]
col_var = [col_var[i] for i in my_order]
```

```
df1 = df.loc[:,col_var]
df1.head(6)
```

[38]:

```
    Age University      Name
0  23.0         AA    Carlos
1  24.0       <NA>    Andrés
2  24.0         BB       NaN
3  25.0       None  Santiago
4   NaN         CC  Fernand0
5  27.0         EE   Marcelo
```