

Sprint03_Tasca02

October 7, 2022

Sprint 3

Tasca 2: Dataframes i anàlisi estadístic

0.0.1 Exercici 1

Descarrega el data set Airlines Delay: Airline on-time statistics and delay causes i carrega'l a un Pandas Dataframe. Explora les dades que conté, explica breument quines variables hi ha i queda't únicament amb les columnes que consideris rellevants. Justifica la teva elecció.

```
[1]: from pathlib import Path
import numpy as np
import pandas as pd
from scipy import stats
import random
import datetime as dt
import math
```

```
[2]: # file paths
data_path = 'D:/Sistema_Solar/Python/itacademy/sprint03/data/'
data_path = Path(data_path)
```

```
[3]: file_name = 'DelayedFlights.csv'
file = data_path / file_name
delay_df = pd.read_csv(file)
```

```
[4]: pd.set_option('display.max_columns', None)
pd.set_option('display.expand_frame_repr', False)
delay_df.describe(include='all')
```

```
[4]:      Unnamed: 0      Year      Month      DayofMonth      DayOfWeek
DepTime  CRSDepTime      ArrTime  CRSArrTime UniqueCarrier  FlightNum
TailNum  ActualElapsedTime  CRSElapsedTime      AirTime      ArrDelay
DepDelay  Origin      Dest      Distance      TaxiIn      TaxiOut
Cancelled CancellationCode      Diverted  CarrierDelay  WeatherDelay
NASDelay  SecurityDelay  LateAircraftDelay
count  1.936758e+06  1936758.0  1.936758e+06  1.936758e+06  1.936758e+06
1.936758e+06  1.936758e+06  1.929648e+06  1.936758e+06      1936758
```

1.936758e+06	1936753	1.928371e+06	1.936560e+06	1.928371e+06	
1.928371e+06	1.936758e+06	1936758	1936758	1.936758e+06	1.929648e+06
1.936303e+06	1.936758e+06		1936758	1.936758e+06	1.247488e+06
1.247488e+06	1.247488e+06	1.247488e+06		1.247488e+06	
unique	NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	20	NaN
5366	NaN	NaN	NaN	NaN	NaN
NaN	303	304	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN
NaN					
top	NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	WN	NaN
N325SW		NaN	NaN	NaN	NaN
NaN	ATL	ORD	NaN	NaN	NaN
N	NaN	NaN	NaN	NaN	NaN
NaN					
freq	NaN	NaN	NaN	NaN	NaN
NaN	NaN	NaN	NaN	377602	NaN
965		NaN	NaN	NaN	NaN
131613	108984	NaN	NaN	NaN	NaN
1936125	NaN	NaN	NaN	NaN	NaN
NaN					
mean	3.341651e+06	2008.0	6.111106e+00	1.575347e+01	3.984827e+00
1.518534e+03	1.467473e+03	1.610141e+03	1.634225e+03		NaN
2.184263e+03	NaN	1.333059e+02	1.343027e+02	1.082771e+02	
4.219988e+01	4.318518e+01	NaN	NaN	7.656862e+02	6.812975e+00
1.823220e+01	3.268348e-04		NaN	4.003598e-03	1.917940e+01
3.703571e+00	1.502164e+01	9.013714e-02		2.529647e+01	
std	2.066065e+06	0.0	3.482546e+00	8.776272e+00	1.995966e+00
4.504853e+02	4.247668e+02	5.481781e+02	4.646347e+02		NaN
1.944702e+03	NaN	7.206007e+01	7.134144e+01	6.864261e+01	
5.678472e+01	5.340250e+01	NaN	NaN	5.744797e+02	5.273595e+00
1.433853e+01	1.807562e-02		NaN	6.314722e-02	4.354621e+01
2.149290e+01	3.383305e+01	2.022714e+00		4.205486e+01	
min	0.000000e+00	2008.0	1.000000e+00	1.000000e+00	1.000000e+00
1.000000e+00	0.000000e+00	1.000000e+00	0.000000e+00		NaN
1.000000e+00	NaN	1.400000e+01	-2.500000e+01	0.000000e+00	
-1.090000e+02	6.000000e+00	NaN	NaN	1.100000e+01	0.000000e+00
0.000000e+00	0.000000e+00		NaN	0.000000e+00	0.000000e+00
0.000000e+00	0.000000e+00	0.000000e+00		0.000000e+00	
25%	1.517452e+06	2008.0	3.000000e+00	8.000000e+00	2.000000e+00
1.203000e+03	1.135000e+03	1.316000e+03	1.325000e+03		NaN
6.100000e+02	NaN	8.000000e+01	8.200000e+01	5.800000e+01	
9.000000e+00	1.200000e+01	NaN	NaN	3.380000e+02	4.000000e+00
1.000000e+01	0.000000e+00		NaN	0.000000e+00	0.000000e+00
0.000000e+00	0.000000e+00	0.000000e+00		0.000000e+00	
50%	3.242558e+06	2008.0	6.000000e+00	1.600000e+01	4.000000e+00

```

1.545000e+03 1.510000e+03 1.715000e+03 1.705000e+03 NaN
1.543000e+03 NaN 1.160000e+02 1.160000e+02 9.000000e+01
2.400000e+01 2.400000e+01 NaN NaN 6.060000e+02 6.000000e+00
1.400000e+01 0.000000e+00 NaN 0.000000e+00 2.000000e+00
0.000000e+00 2.000000e+00 0.000000e+00 8.000000e+00
75% 4.972467e+06 2008.0 9.000000e+00 2.300000e+01 6.000000e+00
1.900000e+03 1.815000e+03 2.030000e+03 2.014000e+03 NaN
3.422000e+03 NaN 1.650000e+02 1.650000e+02 1.370000e+02
5.600000e+01 5.300000e+01 NaN NaN 9.980000e+02 8.000000e+00
2.100000e+01 0.000000e+00 NaN 0.000000e+00 2.100000e+01
0.000000e+00 1.500000e+01 0.000000e+00 3.300000e+01
max 7.009727e+06 2008.0 1.200000e+01 3.100000e+01 7.000000e+00
2.400000e+03 2.359000e+03 2.400000e+03 2.400000e+03 NaN
9.742000e+03 NaN 1.114000e+03 6.600000e+02 1.091000e+03
2.461000e+03 2.467000e+03 NaN NaN 4.962000e+03 2.400000e+02
4.220000e+02 1.000000e+00 NaN 1.000000e+00 2.436000e+03
1.352000e+03 1.357000e+03 3.920000e+02 1.316000e+03

```

```
[5]: delay_df
```

```

[5]:      Unnamed: 0  Year  Month  DayOfMonth  DayOfWeek  DepTime  CRSDepTime
ArrTime  CRSArrTime UniqueCarrier  FlightNum  TailNum  ActualElapsedTime
CRSElapsedTime  AirTime  ArrDelay  DepDelay  Origin  Dest  Distance  TaxiIn
TaxiOut  Cancelled  CancellationCode  Diverted  CarrierDelay  WeatherDelay
NASDelay  SecurityDelay  LateAircraftDelay
0          0  2008      1          3          4    2003.0          1955
2211.0      2225      WN          335  N712SW          128.0
150.0    116.0    -14.0      8.0    IAD  TPA          810      4.0      8.0
0          N          0          NaN          NaN          NaN
NaN          NaN
1          1  2008      1          3          4    754.0          735
1002.0      1000      WN          3231  N772SW          128.0
145.0    113.0      2.0     19.0    IAD  TPA          810      5.0     10.0
0          N          0          NaN          NaN          NaN
NaN          NaN
2          2  2008      1          3          4    628.0          620
804.0      750      WN          448  N428WN          96.0
90.0     76.0     14.0      8.0    IND  BWI          515      3.0     17.0
0          N          0          NaN          NaN          NaN
NaN          NaN
3          4  2008      1          3          4    1829.0          1755
1959.0      1925      WN          3920  N464WN          90.0
90.0     77.0     34.0     34.0    IND  BWI          515      3.0     10.0
0          N          0          2.0          0.0          0.0
0.0          32.0
4          5  2008      1          3          4    1940.0          1915
2121.0      2110      WN          378  N726SW          101.0

```

115.0	87.0	11.0	25.0	IND	JAX	688	4.0	10.0
0		N	0		NaN	NaN	NaN	
NaN		NaN						
...
...
...
...
...
1936753	7009710	2008	12	13		6	1250.0	1220
1617.0	1552		DL	1621	N938DL			147.0
152.0	120.0	25.0	30.0	MSP	ATL	906	9.0	18.0
0		N	0	3.0		0.0	0.0	
0.0		22.0						
1936754	7009717	2008	12	13		6	657.0	600
904.0	749		DL	1631	N3743H			127.0
109.0	78.0	75.0	57.0	RIC	ATL	481	15.0	34.0
0		N	0	0.0		57.0	18.0	
0.0		0.0						
1936755	7009718	2008	12	13		6	1007.0	847
1149.0	1010		DL	1631	N909DA			162.0
143.0	122.0	99.0	80.0	ATL	IAH	689	8.0	32.0
0		N	0	1.0		0.0	19.0	
0.0		79.0						
1936756	7009726	2008	12	13		6	1251.0	1240
1446.0	1437		DL	1639	N646DL			115.0
117.0	89.0	9.0	11.0	IAD	ATL	533	13.0	13.0
0		N	0	NaN		NaN	NaN	
NaN		NaN						
1936757	7009727	2008	12	13		6	1110.0	1103
1413.0	1418		DL	1641	N908DL			123.0
135.0	104.0	-5.0	7.0	SAT	ATL	874	8.0	11.0
0		N	0	NaN		NaN	NaN	
NaN		NaN						

[1936758 rows x 30 columns]

Redueix la dimensió del dataset de manera aleatòria per tal d'obtenir un dataset de només 200.000 registres. Tots els exercicis s'han de fer amb aquest dataset reduït.

```
[6]: delay_df = delay_df.sample(n=200000, random_state=1) # El mostreig serà
      ↪ aleatori però sempre el mateix per random_state=1
```

```
[7]: delay_df
```

```
[7]:      Unnamed: 0  Year  Month  DayOfMonth  DayOfWeek  DepTime  CRSDepTime
      ArrTime  CRSArrTime  UniqueCarrier  FlightNum  TailNum  ActualElapsedTime
      CRSElapsedTime  AirTime  ArrDelay  DepDelay  Origin  Dest  Distance  TaxiIn
      TaxiOut  Cancelled  CancellationCode  Diverted  CarrierDelay  WeatherDelay
```

NASDelay	SecurityDelay	LateAircraftDelay						
1750755	6516830	2008	12	19	5	1332.0	1320	
1436.0	1430		WN	3858 N700GS			64.0	
70.0	49.0	6.0	12.0	CLE BWI	314	7.0	8.0	
0	N	0		NaN	NaN	NaN		
NaN	NaN							
1074983	3577214	2008	6	17	2	1731.0	1720	
2013.0	1955		CO	268 N54241			342.0	
335.0	316.0	18.0	11.0	EWL LAS	2227	8.0	18.0	
0	N	0		11.0	0.0	7.0		
0.0	0.0							
46126	148177	2008	1	27	7	735.0	645	
918.0	815		YV	7343 N27185			103.0	
90.0	54.0	63.0	50.0	ORD BNA	409	4.0	45.0	
0	N	0		63.0	0.0	0.0		
0.0	0.0							
327441	1035260	2008	2	13	3	1333.0	1230	
1515.0	1438		9E	2927 87189E			102.0	
128.0	80.0	37.0	63.0	ORF DTW	529	7.0	15.0	
0	N	0		0.0	0.0	0.0		
0.0	37.0							
1363177	4599044	2008	8	12	2	1359.0	1346	
1517.0	1447		FL	301 N267AT			198.0	
181.0	178.0	30.0	13.0	ATL DEN	1199	5.0	15.0	
0	N	0		13.0	0.0	17.0		
0.0	0.0							
...		
...		
...		
...		
...		
1595898	5763672	2008	10	27	1	2159.0	1920	
112.0	2225		MQ	4205 N643MQ			133.0	
125.0	109.0	167.0	159.0	ORD BDL	783	4.0	20.0	
0	N	0		0.0	0.0	8.0		
0.0	159.0							
1503090	5275042	2008	9	25	4	2052.0	1930	
115.0	2245		AA	346 N433AA			203.0	
135.0	115.0	150.0	82.0	ORD LGA	733	34.0	54.0	
0	N	0		0.0	0.0	150.0		
0.0	0.0							
742602	2433686	2008	5	16	5	1335.0	1325	
1422.0	1415		WN	1902 N316SW			47.0	
50.0	37.0	7.0	10.0	OKC DAL	181	3.0	7.0	
0	N	0		NaN	NaN	NaN		
NaN	NaN							
1840527	6749307	2008	12	23	2	720.0	710	

839.0	831		DL	1653	N996DL		79.0	
81.0	50.0	8.0	10.0	CLT	ATL	227	9.0	20.0
0	N		0	NaN		NaN	NaN	
NaN		NaN						
1526456	5395875	2008	10	6		1	1657.0	1650
2217.0	2205		WN	748	N675AA		200.0	
195.0	185.0	12.0	7.0	PHX	MDW	1444	3.0	12.0
0	N		0	NaN		NaN	NaN	
NaN		NaN						

[200000 rows x 30 columns]

El dataset està conformat per 29 variables:

0. **Unnamed 0**: Camp que ennumera els registres.
1. **Year**: L'any.
2. **Month**: El mes.
3. **DayofMonth**: Dia del mes.
4. **DayofWeek**: Dia de la setmana.
5. **DepTime**: L'hora de sortida real, en hora local (format hhmm).
6. **CRSDepTime**: L'hora de sortida programada.
7. **ArrTime**: L'hora d'arribada real, en hora local.
8. **CRSArrTime**: L'hora d'arribada programada.
9. **UniqueCarrier**: Codi IATA de la companyia.
10. **FlightNum**: Número del vol.
11. **TailNum**: número de la cola del avió que identifica un avió.
12. **ActualElapsedTime**: Durada real del vol, en minuts.
13. **CRSElapsedTime**: Durada del vol programat.
14. **AirTime**: Temps en el qual l'avió es troba enlairat.
15. **ArrDelay**: Retard en l'arribada, en minuts. Sols es considera un vol ha arribat amb retard si ho fa en 15 o més minuts del programat.
16. **DepDelay**: Retard en la sortida.
17. **Origin**: Codi IATA de l'aeroport de sortida.
18. **Dest**: Codi IATA de l'aeroport d'arribada.
19. **Distance**: Distància del vol, en milles.
20. **TaxiIn**: Temps en el qual l'avió circula per pista en direcció a la pista d'enlairament.
21. **TaxiOut**: Temps en el qual l'avió circula per pista des de l'aterratge fins el lloc de desembarcament.
22. **Cancelled**: 1 si el vol ha sigut cancel·lat o 0 si no ho ha estat.
23. **CancellationCode**: Codi de cancel·lació indicant la raó (A = Aerolinia; B = Temps; C = Retards en el Sistema d'espai aeri nacional que inclou condicions meteorològiques no extremes, operacions en l'aeroport, alt volum de tràfic, etc...; D = Seguretat).
24. **Diverted**: Indica si el vol ha sigut desviat (1 = Sí, 0 = No).
25. **CarrierDelay**: El temps de retard degut a motius de l'aerolínia.
26. **WeatherDelay**: El temps de retard degut a motius del temps.
27. **NASDelay**: El temps de retard degut al NAS, el Sistema d'espai aeri nacional.
28. **SecurityDelay**: El temps de retard degut a motius de seguretat.
29. **LateAircraftDelay**: El temps de retard en un aeroport degut al retard en l'aeroport anterior.

d'on ve un avió.

Selecció de variables i creació de noves

La columna 0 s'elimina. Les columnes 1 a 3 ens indiquen el dia i l'hora del vol, les mantenim però eliminem (4) el dia de la setmana.

Mentre que les de la 5 a la 8, ens quedem simplement amb l'hora programada ja que ja obtenim els retards en un altre camp. Però mantenim l'hora d'arribada real per fer comprovacions posteriors.

```
[8]: delay_df = delay_df.drop(columns=['Unnamed: 0', 'DayOfWeek', 'DepTime',  
    ↪ 'CRSDepTime'])
```

Podem calcular el número de vols diferents que hi han a partir del número de vol

```
[9]: print(len(delay_df['FlightNum'].unique()))
```

7284

Com també quin són els vols més freqüents:

```
[10]: flight_num_count = delay_df['FlightNum'].value_counts()  
print(flight_num_count)
```

```
44      172  
16      168  
50      164  
511     154  
47      154
```

```
...  
7680     1  
7404     1  
7751     1  
6837     1  
6050     1
```

Name: FlightNum, Length: 7284, dtype: int64

Quin són aquests trajectes? Necessitem les columnes Origin i Dest que són les mateixes per a cada número de vol. Ho comprovem amb el número de vol més nombrós.

```
[11]: search = delay_df['FlightNum'] == flight_num_count.iloc[0]  
busiest_flight_df = delay_df[search]
```

```
[12]: busiest_flight_df['Origin']
```

```
[12]: 1148884    SFO  
      293499    TUS  
      1704202   MEM  
      1404151   JNU  
      255291    SFO  
      1682529   SAN  
      1544090   OAK
```

445	MCO
629430	SFO
318361	MEM
1211421	MEM
660595	TUS
1259845	SEA
1591231	LIH
1347596	SEA
600485	BOI
1259111	SEA
360117	SMF
1835687	SAN
1081777	SEA
1879932	OMA
1265303	SEA
1879925	MEM
629428	SFO
1408768	SMF
1259471	SEA
1065670	SMF
488276	TUS
1406660	SMF
823233	LIH
130753	MEM
1577578	SEA
1536771	OAK
1933962	SAN
1412610	SMF
68322	SFO
1412342	SMF
1211419	MEM
787566	SFO
554411	SMF
1731267	SAN
1811336	SFO
1081326	SEA
1666007	SFO
9711	MDW
600490	BOI
1241955	JNU
823232	LIH
1242467	JNU
1062350	SMF
1921765	SMF

Name: Origin, dtype: object

0.0.2 Continuem

Mantenim (9) UniqueCarrier per identificar l'aerolinia i (11) el número de cola. Descartem el (10) número de vol. Del (12) i (13), la diferència és el retard però mantenim (14) per calcular posteriorment la velocitat mitjana. Retindrem (15) ArrDelay i (16) DepDelay.

```
[13]: delay_df = delay_df.drop(columns=['CRSElapsedTime', 'FlightNum',  
    ↪ 'ActualElapsedTime'])
```

Mantenim els codis IATA dels aeroports d'origen i de sortida (17 i 18). Mantenim la distància per calcular la velocitat (19) però descartem el temps que està l'avió circulant a l'aeroport (20 i 21)

```
[14]: #delay_df = delay_df.drop(columns=['TaxiIn', 'TaxiOut'])
```

Retenim (22) i (23) per saber quins vol són cancel·lats, i els motius. i també els retards atribuïts als diferents motius (25-29).

```
[15]: delay_df = delay_df.drop(columns=['Diverted'])
```

0.0.3 Exercici 2

Fes un informe complet del dataset:

- Resumeix estadísticament el dataset i les columnes d'interès.
- Fes una anàlisi estadístic del que consideris rellevant.
- Troba quantes dades faltants hi ha per columna.
- Crea columnes noves (velocitat mitjana del vol, si ha arribat tard o no...).
- Fes una taula de les aerolínies amb més endarreriments acumulats.
- Quins són els vols més llargs? I els més endarrerits? Busca les rutes més llargues i les que acumulen més retards.
- Aporta allò que consideris rellevant.

Creem una nova columna 'Date' que uneix la informació de les columnes 1 a 4 en una sola. Aquestes columnes les eliminem i aprofitem per reordenar-les

```
[16]: delay_df['Date'] = pd.to_datetime(pd.DataFrame({'year':delay_df['Year'],  
    ↪ 'month':delay_df['Month'], 'day':delay_df['DayofMonth']}))
```

```
delay_df = delay_df.drop(columns=['Year', 'Month', 'DayofMonth'])
```

```
[17]: cols = delay_df.columns.tolist()  
cols = cols[-1:] + cols[:-1]  
delay_df = delay_df[cols]
```

1. Date
2. CRSArrTime
3. UniqueCarrier

4. FlightNum
5. TailNum
6. ArrDelay
7. DepDelay
8. Origin
9. Dest
10. Cancelled
11. CancellationCode
12. CarrierDelay
13. WeatherDelay
14. NASDelay
15. SecurityDelay
16. LateAircraftDelay

[18]: delay_df

```
[18]:      Date  ArrTime  CRSArrTime UniqueCarrier TailNum  AirTime  ArrDelay
DepDelay Origin Dest  Distance  TaxiIn  TaxiOut  Cancelled CancellationCode
CarrierDelay WeatherDelay NASDelay SecurityDelay LateAircraftDelay
1750755 2008-12-19  1436.0      1430      WN  N700GS    49.0      6.0
12.0    CLE  BWI      314      7.0      8.0      0      N
NaN      NaN      NaN      NaN      NaN      NaN
1074983 2008-06-17  2013.0      1955      CO  N54241    316.0     18.0
11.0    EWR  LAS      2227      8.0     18.0      0      N
11.0      0.0      7.0      0.0      0.0
46126   2008-01-27  918.0      815      YV  N27185    54.0     63.0
50.0    ORD  BNA      409      4.0     45.0      0      N
63.0      0.0      0.0      0.0      0.0
327441  2008-02-13  1515.0      1438      9E  87189E    80.0     37.0
63.0    ORF  DTW      529      7.0     15.0      0      N
0.0      0.0      0.0      0.0      37.0
1363177 2008-08-12  1517.0      1447      FL  N267AT    178.0     30.0
13.0    ATL  DEN      1199      5.0     15.0      0      N
13.0      0.0     17.0      0.0      0.0
...      ...      ...      ...      ...      ...      ...      ...
```

```

...      ...      ...      ...      ...      ...      ...
...
1595898 2008-10-27      112.0      2225      MQ  N643MQ      109.0      167.0
159.0   ORD  BDL      783      4.0      20.0      0      N
0.0      0.0      8.0      0.0      159.0
1503090 2008-09-25      115.0      2245      AA  N433AA      115.0      150.0
82.0   ORD  LGA      733      34.0      54.0      0      N
0.0      0.0      150.0      0.0      0.0
742602 2008-05-16      1422.0      1415      WN  N316SW      37.0      7.0
10.0   OKC  DAL      181      3.0      7.0      0      N
NaN      NaN      NaN      NaN      NaN
1840527 2008-12-23      839.0      831      DL  N996DL      50.0      8.0
10.0   CLT  ATL      227      9.0      20.0      0      N
NaN      NaN      NaN      NaN      NaN
1526456 2008-10-06      2217.0      2205      WN  N675AA      185.0      12.0
7.0   PHX  MDW      1444      3.0      12.0      0      N
NaN      NaN      NaN      NaN      NaN

```

[200000 rows x 20 columns]

Quina és la completitud de les dades disponibles? Cerquem els NA de les diferents variables?

```
[19]: delay_df.isna().sum()
```

```

[19]: Date                0
ArrTime                792
CRSArrTime             0
UniqueCarrier          0
TailNum                1
AirTime               920
ArrDelay              920
DepDelay              0
Origin                0
Dest                  0
Distance              0
TaxiIn                792
TaxiOut               58
Cancelled              0
CancellationCode       0
CarrierDelay          71210
WeatherDelay          71210
NASDelay              71210
SecurityDelay         71210
LateAircraftDelay     71210
dtype: int64

```

```
[20]: delay_df.isna().sum() / len(delay_df) * 100
```

```
[20]: Date            0.0000
      ArrTime         0.3960
      CRSArrTime      0.0000
      UniqueCarrier   0.0000
      TailNum         0.0005
      AirTime         0.4600
      ArrDelay        0.4600
      DepDelay        0.0000
      Origin          0.0000
      Dest            0.0000
      Distance        0.0000
      TaxiIn          0.3960
      TaxiOut         0.0290
      Cancelled       0.0000
      CancellationCode 0.0000
      CarrierDelay    35.6050
      WeatherDelay    35.6050
      NASDelay        35.6050
      SecurityDelay   35.6050
      LateAircraftDelay 35.6050
      dtype: float64
```

1. Date
2. ArrTime
3. CRSArrTime
4. UniqueCarrier
5. TailNum
6. AirTime
7. ArrDelay
8. DepDelay
9. Origin
10. Dest
11. Distance
12. TaxiIn
13. TaxiOut
14. Cancelled
15. CancellationCode
16. CarrierDelay
17. WeatherDelay

18. NASDelay

19. SecurityDelay

20. LateAircraftDelay

1. Data

Però açò pot estar relacionat a que falten dades o bé perquè, en els casos on apareixen NA, relacionat amb que no s'ha registrat retards. Anem a inspeccionar les diferents variables.

La data de les dades és de l'any 2008

```
[21]: delay_df['Date'].min()
```

```
[21]: Timestamp('2008-01-01 00:00:00')
```

```
[22]: delay_df['Date'].max()
```

```
[22]: Timestamp('2008-12-31 00:00:00')
```

2. CRSArrTime: El temps d'arribada esperada

```
[23]: delay_df['CRSArrTime'] = delay_df['CRSArrTime']/100
```

```
[24]: delay_df['CRSArrTime']
```

```
[24]: 1750755    14.30
      1074983    19.55
      46126     8.15
      327441    14.38
      1363177   14.47
      ...
      1595898    22.25
      1503090    22.45
      742602    14.15
      1840527     8.31
      1526456    22.05
      Name: CRSArrTime, Length: 200000, dtype: float64
```

3. UniqueCarrier.

Les diferents companyies que conformen les dades són:

```
[25]: np.sort(delay_df['UniqueCarrier'].unique())
```

```
[25]: array(['9E', 'AA', 'AQ', 'AS', 'B6', 'CO', 'DL', 'EV', 'F9', 'FL', 'HA',
      'MQ', 'NW', 'OH', 'OO', 'UA', 'US', 'WN', 'XE', 'YV'], dtype=object)
```

Aquest és el codi IATA (International Air Transport Association). Perquè quede més clar de quines companyies parlem, anem a incloure-les a la nostra dataframe a partir de la informació

recabada de les següents pàgines webs. https://en.wikipedia.org/wiki/List_of_airline_codes

https://aspm.faa.gov/aspmhelp/index/ASQP____Carrier_Codes_And_Names.html

<https://www.tvlon.com/resources/airlinecodes.htm>

Confeccionem un excel i l'apugem.

```
[26]: file_name = 'UniqueCarrier.csv'
      file = data_path / file_name
      unique_carrier_df = pd.read_csv(file, sep = ';')
```

L'excel conté la relació dels codis i el nom de la companyia

```
[27]: unique_carrier_df.iloc[0:2]
```

```
[27]:  IATA Code  Air Carrier Name
      0      9E      Endeavor Air
      1      AA  American Airlines
```

```
[28]: air_carrier_names = []
      for code in delay_df['UniqueCarrier']:
          filter = unique_carrier_df['IATA Code'] == code
          name = unique_carrier_df[filter]['Air Carrier Name']
          try:
              air_carrier_names.append(name.values[0])
          except IndexError:
              air_carrier_names.append(np.nan)
```

```
[29]: delay_df['air_carrier_names'] = air_carrier_names
```

4. NumTail

```
[30]: print(delay_df['TailNum'].value_counts())
      print(len(delay_df['TailNum'].unique()))
```

```
N676SW    115
N665WN    108
N325SW    107
N328SW    103
N683SW    103
```

...

```
N603NW     1
N1501P     1
N186DN     1
N173DZ     1
N823AL     1
```

```
Name: TailNum, Length: 5269, dtype: int64
5270
```

5. Rutes

Quins són els trajectes més freqüents?

Amb els camps d'origen i destí, crearem un nou, la ruta.

```
[31]: delay_df['Route'] = delay_df['Origin'] + '-' + delay_df['Dest']
```

```
[32]: delay_df['Route'].value_counts()
```

```
[32]: ORD-LGA      478
      LAX-SFO      473
      SFO-LAX      423
      ATL-LGA      400
      LGA-ATL      381
      ...
      LAS-ROC       1
      AKN-DLG       1
      ABE-LGA       1
      OKC-SAT       1
      GRR-MCO       1
      Name: Route, Length: 4805, dtype: int64
```

Estes sigles corresponen al codi de la IATA i les podem traduir als aeroports per fer més fàcil reconèixer què rutes són, utilitzem aquesta relació dels codis:

<https://datahub.io/core/airport-codes#resource-airport-codes>

```
[33]: file_name = 'airport-codes_csv.csv'
      file = data_path / file_name
      IATA_airport_df = pd.read_csv(file, sep = ',')
```

```
[34]: IATA_airport_df
```

```
[34]:
```

	ident	type	name	elevation_ft			
	continent	iso_country	iso_region	municipality	gps_code	iata_code	local_code
0	00A	heliport	Total Rf Helipoint				11.0
NaN	US	US-PA	Bensalem	00A	NaN	00A	
-74.93360137939453, 40.07080078125							
1	00AA	small_airport	Aero B Ranch Airport				3435.0
NaN	US	US-KS	Leoti	00AA	NaN	00AA	
-101.473911, 38.704022							
2	00AK	small_airport	Lowell Field				450.0
NaN	US	US-AK	Anchor Point	00AK	NaN	00AK	
-151.695999146, 59.94919968							
3	00AL	small_airport	Epps Airpark				820.0
NaN	US	US-AL	Harvest	00AL	NaN	00AL	
-86.77030181884766, 34.86479949951172							
4	00AR	closed	Newport Hospital & Clinic Helipoint				237.0
NaN	US	US-AR	Newport	NaN	NaN	NaN	
-91.254898, 35.6087							

```

...      ...      ...      ...      ...      ...
...
57416      ZYYK      medium_airport      Yingkou Lanqi Airport      0.0
AS      CN      CN-21      Yingkou      ZYYK      YKH      NaN
122.3586, 40.542524
57417      ZYYY      medium_airport      Shenyang Dongta Airport      NaN
AS      CN      CN-21      Shenyang      ZYYY      NaN      NaN
123.49600219726562, 41.784400939941406
57418      ZZ-0001      heliport      Sealand Helipad      40.0
EU      GB      GB-ENG      Sealand      NaN      NaN      NaN
1.4825, 51.894444
57419      ZZ-0002      small_airport      Glorioso Islands Airstrip      11.0
AF      TF      TF-U-A      Grande Glorieuse      NaN      NaN      NaN
47.296388888900005, -11.584277777799999
57420      ZZZZ      small_airport      Satsuma IÅ jima Airport      338.0
AS      JP      JP-46      Mishima-Mura      RJX7      NaN      RJX7
130.270556, 30.784722

[57421 rows x 12 columns]

```

```

[35]: # Creem una llista amb les rutes amb més vols i afegim el números de vols
busiest_routes = delay_df['Route'].value_counts()
busiest_routes_names = []
length = 5

for i in range(length):
    code = busiest_routes.index[i]
    route_name = []
    code = code.split('-') # Separem la ruta en els dos codis individuals per
    fer la conversió al nom
    search = IATA_airport_df['iata_code'] == code[0]
    origin = IATA_airport_df[search]['name'].values[0]
    search = IATA_airport_df['iata_code'] == code[1]
    dest = IATA_airport_df[search]['name'].values[0]
    busiest_routes_names.append(origin + ' - ' + dest + ' (' +
    str(busiest_routes.values[i]) + ')')

```

```

[36]: busiest_routes_names

```

```

[36]: ["Chicago O'Hare International Airport - La Guardia Airport (478)",
'Los Angeles International Airport - San Francisco International Airport
(473)',
'San Francisco International Airport - Los Angeles International Airport
(423)',
'Hartsfield Jackson Atlanta International Airport - La Guardia Airport (400)',
'La Guardia Airport - Hartsfield Jackson Atlanta International Airport (381)']

```


Aquí estem considerant de forma separada viatges d'anada i tornada i les podriem intentar comptar com les mateixes. Per exemple, la primera ruta és. ORD-LGA. Per tant, ha d'existir també la ruta LGA - ORD. Anem a buscar-ho.

```
[37]: search = delay_df['Route'] == 'LGA-ORD'
      delay_df['Route'][search]
```

```
[37]: 1566991    LGA-ORD
      1468307    LGA-ORD
      260761    LGA-ORD
      1038324    LGA-ORD
      1816177    LGA-ORD
      ...
      454743    LGA-ORD
      261023    LGA-ORD
      456196    LGA-ORD
      634465    LGA-ORD
      1386942    LGA-ORD
      Name: Route, Length: 348, dtype: object
```

```
[38]: len(delay_df['Route'][search])
```

```
[38]: 348
```

```
[39]: # Creem un nou llistat que continga sols un sentit de cada ruta
      route_preunique = delay_df['Route'].unique()
      route_unique = []

      for route in route_preunique:
          route_inv = route.split('-')
          route_inv = route_inv[1] + '-' + route_inv[0]
          if (route not in route_unique and route_inv not in route_unique):
              route_unique.append(route)
```

```
[40]: len(route_preunique)
```

```
[40]: 4805
```

```
[41]: len(route_unique) # Les rutes realment úniques són més o menys la meitat que si
      ↪ les comptem les anades i tornades de forma separada
```

```
[41]: 2476
```

```
[42]: # Fem el compteig del número de vegades que es fa cada ruta única
      route_dict = {}
      for route in route_unique:
          search = delay_df['Route'] == route
          count = len(delay_df['Route'][search])
```

```

route = route.split('-')
route_2 = route[1] + '-' + route[0]
search = delay_df['Route'] == route_2
count += len(delay_df['Route'][search])
route_dict[route_2] = count

route_df = pd.DataFrame.from_dict(data=route_dict, orient='index',
    ↪columns=['count'])

```

```
[43]: busiest_routes_2 = route_df.sort_values(by=['count'], ascending = False)
```

```
[44]: busiest_routes_names_2 = []
length = 10

for i in range(length):
    code = busiest_routes_2.index[i]
    route_name = []
    code = code.split('-')
    search = IATA_airport_df['iata_code'] == code[0]
    origin = IATA_airport_df[search]['name'].values[0]
    search = IATA_airport_df['iata_code'] == code[1]
    dest = IATA_airport_df[search]['name'].values[0]
    busiest_routes_names_2.append(origin + ' - ' + dest + ' (' +
    ↪str(busiest_routes_2.values[i]) + ')')

```

```
[45]: busiest_routes_names_2
```

```
[45]: ['San Francisco International Airport - Los Angeles International Airport
([896])',
'La Guardia Airport - Chicago O'Hare International Airport ([826])',
'Hartsfield Jackson Atlanta International Airport - La Guardia Airport
([781])',
'William P Hobby Airport - Dallas Love Field ([656])',
'Newark Liberty International Airport - Chicago O'Hare International Airport
([642])',
'Chicago O'Hare International Airport - Dallas Fort Worth International Airport
([634])',
'Los Angeles International Airport - McCarran International Airport ([625])',
'Newark Liberty International Airport - Hartsfield Jackson Atlanta
International Airport ([620])',
'Chicago O'Hare International Airport - Minneapolis-St Paul International/Wold-
Chamberlain Airport ([611])',
'Dallas Fort Worth International Airport - Hartsfield Jackson Atlanta
International Airport ([608])']

```

6. Retards:

En el cas dels retards, ens trobem que hi han dades NA, aquestes per què ocorren? Poden ser vols

que no hagen tingut endarreriment, s'hagin cancel·lat? Ens calen les dades reals d'arribada.

```
[46]: # Seleccionarem primer els vols amb NA en el retard i després, sí també tenen
      ↪ Nan en el temps real d'arribada
```

```
nan_delay = delay_df[np.isnan(delay_df['ArrDelay'])]
nan_delay[np.isnan(nan_delay['ArrTime'])]
```

```
[46]:
```

	Date	ArrTime	CRSArrTime	UniqueCarrier	TailNum	AirTime	ArrDelay
DepDelay	Origin	Dest	Distance	TaxiIn	TaxiOut	Cancelled	CancellationCode
CarrierDelay	WeatherDelay	NASDelay	SecurityDelay	LateAircraftDelay			
air_carrier_names	Route						
1889862	2008-12-21	NaN	23.07	9E	87589E	NaN	NaN
144.0	DTW	LEX	296	NaN	NaN	1	B
NaN	NaN	NaN	NaN	NaN	NaN	NaN	Endeavor
Air	DTW-LEX						
921578	2008-06-05	NaN	20.20	XE	N14959	NaN	NaN
190.0	MSP	IAH	1034	NaN	65.0	0	N
NaN	NaN	NaN	NaN	NaN	NaN	NaN	
JSX	MSP-IAH						
1501260	2008-09-14	NaN	24.00	9E	89289E	NaN	NaN
8.0	MSP	SDF	603	NaN	26.0	0	N
NaN	NaN	NaN	NaN	NaN	NaN	NaN	Endeavor
Air	MSP-SDF						
3179	2008-01-04	NaN	21.50	WN	N368SW	NaN	NaN
106.0	SJC	LAX	308	NaN	17.0	0	N
NaN	NaN	NaN	NaN	NaN	NaN	NaN	Southwest
Airlines	SJC-LAX						
1303762	2008-08-04	NaN	15.50	YV	N75993	NaN	NaN
217.0	MBS	ORD	222	NaN	12.0	0	N
NaN	NaN	NaN	NaN	NaN	NaN	NaN	Mesa Airlines,
Inc.	MBS-ORD						
...
...
...
...
1420318	2008-08-15	NaN	17.59	CO	N73251	NaN	NaN
51.0	IAH	LGA	1416	NaN	83.0	0	N
NaN	NaN	NaN	NaN	NaN	NaN	NaN	Continental Air
Lines	IAH-LGA						
1227495	2008-07-02	NaN	21.20	AA	N271AA	NaN	NaN
92.0	DFW	EWR	1372	NaN	14.0	0	N
NaN	NaN	NaN	NaN	NaN	NaN	NaN	American
Airlines	DFW-EWR						
1092843	2008-07-12	NaN	23.10	WN	N268WN	NaN	NaN
11.0	PHX	HOU	1020	NaN	16.0	0	N
NaN	NaN	NaN	NaN	NaN	NaN	NaN	Southwest
Airlines	PHX-HOU						

351149	2008-02-04		NaN	17.30		AA	N5FEAA	NaN	NaN
231.0	EWR	EGE	1725	NaN	25.0		0	N	
NaN		NaN	NaN		NaN		NaN	American	
Airlines EWR-EGE									
1017381	2008-06-06		NaN	13.20		MQ	N626AE	NaN	NaN
47.0	CLT	ORD	599	NaN	24.0		0	N	
NaN		NaN	NaN		NaN		NaN		Envoy
Air CLT-ORD									

[792 rows x 22 columns]

```
[47]: len(delay_df[delay_df.Cancelled == 1])
```

[47]: 77

Observem que la major part dels Nans es deuen a que falta l'hora d'arribada real (en alguns casos és degut a que s'han cancel·lat), tot i que sí que hi han dades del retard en l'eixida. Així que descartem els registres amb Nans.

```
[48]: delay_df_notna = delay_df[delay_df['ArrDelay'].notna()]
```

Sorpren la elevada mitjana dels retards, que supera els 40 minuts. Tot i que la mediana es de 24 minuts. Ho siga, la mitja es troba influenciada pels vols que arriben a tindre retards superiors a un dia i tot i això, hi han valors inferiors a 0 que indiquen que s'ha arribat abans. (Cosa que pot ocórrer perquè l'avió trobe vents en altura intensos de cua).

```
[49]: print('Mitja: ' + str(round(delay_df_notna['ArrDelay'].mean(), 0)))
```

```
per = [0, 3, 10, 20, 30, 40, 50, 60, 70, 80, 90, 95, 97, 100]
```

```
for p in per:
    print('Percentil ' + str(p) + ': ' + str(np.
        percentile(delay_df_notna['ArrDelay'], p)))
```

```
Mitja: 42.0
Percentil 0: -59.0
Percentil 3: -8.0
Percentil 10: 0.0
Percentil 20: 6.0
Percentil 30: 12.0
Percentil 40: 17.0
Percentil 50: 24.0
Percentil 60: 34.0
Percentil 70: 47.0
Percentil 80: 67.0
Percentil 90: 106.0
Percentil 95: 148.0
Percentil 97: 180.0
Percentil 100: 1951.0
```

Creem una nova columna i classifiquem els endarreriments. Podem considerar que si arriba en menys de 15 minuts no hi ha hagut retard.

Menys de 15 minuts: Sense retard

Entre 15 y 30 minuts: Retard lleuger

Entre 30 y 60 minuts: Retard moderat

Més de 60 minuts: Gran retard

```
[50]: delay_df.loc[delay_df.ArrDelay < 15, 'ArrDelay_Class'] = 'Sense retard'
      delay_df.loc[(delay_df.ArrDelay >= 15) & (delay_df.ArrDelay < 30),
        ↪ 'ArrDelay_Class'] = 'Petit'
      delay_df.loc[(delay_df.ArrDelay >= 30) & (delay_df.ArrDelay < 60),
        ↪ 'ArrDelay_Class'] = 'Mitjà'
      delay_df.loc[delay_df.ArrDelay > 60, 'ArrDelay_Class'] = 'Gran'

      delay_class_count = delay_df['ArrDelay_Class'].value_counts()[['Sense retard',
        ↪ 'Petit', 'Mitjà', 'Gran']]
      delay_class_count
```

```
[50]: Sense retard      70290
      Petit           40933
      Mitjà           41635
      Gran            45354
      Name: ArrDelay_Class, dtype: int64
```

Un 35 % dels vols no tenen retards i més d'un 20 % tenen retards superiors a l'hora

```
[51]: round((delay_class_count / len(delay_df) * 100), 1)
```

```
[51]: Sense retard      35.1
      Petit           20.5
      Mitjà           20.8
      Gran            22.7
      Name: ArrDelay_Class, dtype: float64
```

Quins són els retards segons l'aerolínia?

```
[52]: carrier_delay_dict = {}

      for carrier in delay_df['air_carrier_names'].unique():
          subset = delay_df[delay_df['air_carrier_names'] == carrier]
          carrier_delay_dict[carrier] = subset['ArrDelay_Class'].
            ↪ value_counts()[['Sense retard', 'Petit', 'Mitjà', 'Gran']]

      carrier_delay_df = pd.DataFrame.from_dict(data=carrier_delay_dict,
        ↪ orient='index')
```

```

carrier_delay_df['Total'] = carrier_delay_df['Sense retard'] +
    ↪carrier_delay_df['Petit'] + carrier_delay_df['Mitjà'] +
    ↪carrier_delay_df['Gran']

carrier_delay_df['Sense retard %'] = round(carrier_delay_df['Sense retard'] /
    ↪carrier_delay_df['Total'] * 100, 1)
carrier_delay_df['Petit %'] = round(carrier_delay_df['Petit'] /
    ↪carrier_delay_df['Total'] * 100, 1)
carrier_delay_df['Mitjà %'] = round(carrier_delay_df['Mitjà'] /
    ↪carrier_delay_df['Total'] * 100, 1)
carrier_delay_df['Gran %'] = round(carrier_delay_df['Gran'] /
    ↪carrier_delay_df['Total'] * 100, 1)

```

```
[53]: carrier_delay_df.sort_values(by = 'Sense retard %', ascending = False)
```

```
[53]:
```

	Sense retard	Petit	Mitjà	Gran	Total	Sense
retard %	Petit %	Mitjà %	Gran %			
Mapjet	35	17	8	8	68	
51.5	25.0	11.8	11.8			
Southwest Airlines	17962	8079	7105	5643	38789	
46.3	20.8	18.3	14.5			
Frontier Airlines, Inc.	1259	784	535	387	2965	
42.5	26.4	18.0	13.1			
Hawaiian Airlines Inc.	319	212	139	95	765	
41.7	27.7	18.2	12.4			
Continental Air Lines	4329	1807	1897	2435	10468	
41.4	17.3	18.1	23.3			
US Airways	3872	2099	2089	2005	10065	
38.5	20.9	20.8	19.9			
Alaska Airlines	1537	914	878	707	4036	
38.1	22.6	21.8	17.5			
Delta Air Lines, Inc.	4304	2655	2362	2465	11786	
36.5	22.5	20.0	20.9			
AirTran	2447	1633	1506	1709	7295	
33.5	22.4	20.6	23.4			
United Airlines, Inc.	4668	2614	2990	3953	14225	
32.8	18.4	21.0	27.8			
Skywest Airlines	4405	2779	2747	3587	13518	
32.6	20.6	20.3	26.5			
Envoy Air	4502	2983	3398	3558	14441	
31.2	20.7	23.5	24.6			
Endeavor Air	1649	1086	1214	1368	5317	
31.0	20.4	22.8	25.7			
Northwest Airlines	2474	1817	2032	1687	8010	
30.9	22.7	25.4	21.1			
American Airlines	6130	4081	4501	5116	19828	
30.9	20.6	22.7	25.8			

ExpressJet Airlines					2548	1821	1838	2156	8363
30.5	21.8	22.0	25.8						
Jetblue Airways Corporation					1710	930	1191	1797	5628
30.4	16.5	21.2	31.9						
JSX					3137	2033	2324	3002	10496
29.9	19.4	22.1	28.6						
Jetstream Intl					1371	1163	1347	1484	5365
25.6	21.7	25.1	27.7						
Mesa Airlines, Inc.					1632	1426	1534	2192	6784
24.1	21.0	22.6	32.3						

Quines rutes tenen més retards?

```
[54]: # Considerarem de nou, les anades i tornades com les mateixes rutes. Ara
      ↪ crearem un nou camp amb les rutes úniques al dataframe
for route in route_unique:
    delay_df.loc[delay_df.Route == route, 'RouteUnique'] = route
    route_split = route.split('-')
    route_2 = route_split[1] + '-' + route_split[0]
    delay_df.loc[delay_df.Route == route_2, 'RouteUnique'] = route

# Fem un diccionari per clasificar els retards en cada una de les rutes.
route_delay_dict = {}
for route in route_unique:
    subset = delay_df[delay_df['RouteUnique'] == route]
    delay_class_count = subset['ArrDelay_Class'].value_counts()
    route_delay_dict[route] = delay_class_count

# Calcularem els percentatges
route_delay_df = pd.DataFrame.from_dict(data=route_delay_dict, orient='index')
route_delay_df['Total'] = route_delay_df['Sense retard'] +
    ↪ route_delay_df['Petit'] + route_delay_df['Mitjà'] + route_delay_df['Gran']
route_delay_df['Sense retard %'] = round(route_delay_df['Sense retard'] /
    ↪ route_delay_df['Total'] * 100, 1)
route_delay_df['Petit %'] = round(route_delay_df['Petit'] /
    ↪ route_delay_df['Total'] * 100, 1)
route_delay_df['Mitjà %'] = round(route_delay_df['Mitjà'] /
    ↪ route_delay_df['Total'] * 100, 1)
route_delay_df['Gran %'] = round(route_delay_df['Gran'] /
    ↪ route_delay_df['Total'] * 100, 1)

route_delay_df.sort_values(by='Total', ascending=False)
```

```
[54]:
```

	Sense retard	Mitjà	Petit	Gran	Total	Sense retard %	Petit %	
Mitjà %	Gran %							
LAX-SFO		263.0	187.0	159.0	281.0	890.0	29.6	17.9
21.0	31.6							

ORD-LGA		191.0	191.0	127.0	307.0	816.0		23.4	15.6
23.4	37.6								
LGA-ATL		245.0	184.0	152.0	190.0	771.0		31.8	19.7
23.9	24.6								
DAL-HOU		251.0	148.0	150.0	93.0	642.0		39.1	23.4
23.1	14.5								
ORD-EWR		163.0	128.0	83.0	262.0	636.0		25.6	13.1
20.1	41.2								
...	
...	...								
MRY-SLC		NaN	NaN	NaN	1.0	NaN		NaN	NaN
NaN	NaN								
TUL-OMA		NaN	NaN	NaN	1.0	NaN		NaN	NaN
NaN	NaN								
IAD-BUF		NaN	NaN	NaN	1.0	NaN		NaN	NaN
NaN	NaN								
LIH-SAN		NaN	NaN	NaN	1.0	NaN		NaN	NaN
NaN	NaN								
TOL-CVG		NaN	NaN	NaN	1.0	NaN		NaN	NaN
NaN	NaN								

[2463 rows x 9 columns]

Abans convé fer algo amb els NaN, ja que afecta per a obtindre el valor Total

```
[55]: route_delay_df = route_delay_df[['Sense retard', 'Mitjà', 'Petit', 'Gran',
↳ 'Total']].fillna(0)
route_delay_df['Total'] = route_delay_df['Sense retard'] +
↳ route_delay_df['Petit'] + route_delay_df['Mitjà'] + route_delay_df['Gran']
route_delay_df['Sense retard %'] = round(route_delay_df['Sense retard'] /
↳ route_delay_df['Total'] * 100, 1)
route_delay_df['Petit %'] = round(route_delay_df['Petit'] /
↳ route_delay_df['Total'] * 100, 1)
route_delay_df['Mitjà %'] = round(route_delay_df['Mitjà'] /
↳ route_delay_df['Total'] * 100, 1)
route_delay_df['Gran %'] = round(route_delay_df['Gran'] /
↳ route_delay_df['Total'] * 100, 1)

route_delay_df.sort_values(by = 'Total', ascending = False)
```

```
[55]:      Sense retard  Mitjà  Petit  Gran  Total  Sense retard %  Petit %
Mitjà %  Gran %
LAX-SFO      263.0  187.0  159.0  281.0  890.0      29.6      17.9
21.0      31.6
ORD-LGA      191.0  191.0  127.0  307.0  816.0      23.4      15.6
23.4      37.6
LGA-ATL      245.0  184.0  152.0  190.0  771.0      31.8      19.7
```


23.9	24.6							
DAL-HOU		251.0	148.0	150.0	93.0	642.0	39.1	23.4
23.1	14.5							
ORD-EWR		163.0	128.0	83.0	262.0	636.0	25.6	13.1
20.1	41.2							
...	
...	...							
XNA-CVG		1.0	0.0	0.0	0.0	1.0	100.0	0.0
0.0	0.0							
CLE-ORF		0.0	1.0	0.0	0.0	1.0	0.0	0.0
100.0	0.0							
LGA-AGS		1.0	0.0	0.0	0.0	1.0	100.0	0.0
0.0	0.0							
KOA-LIH		1.0	0.0	0.0	0.0	1.0	100.0	0.0
0.0	0.0							
TOL-CVG		0.0	0.0	0.0	1.0	1.0	0.0	0.0
0.0	100.0							

[2463 rows x 9 columns]

```
[56]: route_delay_df.sort_values(by='Sense retard %', ascending=False)
```

		Sense retard	Mitjà	Petit	Gran	Total	Sense retard %	Petit %	Mitjà
%	Gran %								
SLC-LGA		1.0	0.0	0.0	0.0	1.0	100.0	0.0	
0.0	0.0								
MSP-BJI		1.0	0.0	0.0	0.0	1.0	100.0	0.0	
0.0	0.0								
CLT-CMH		2.0	0.0	0.0	0.0	2.0	100.0	0.0	
0.0	0.0								
AKN-DLG		1.0	0.0	0.0	0.0	1.0	100.0	0.0	
0.0	0.0								
MYR-MSP		1.0	0.0	0.0	0.0	1.0	100.0	0.0	
0.0	0.0								
...	
...	...								
GSP-MCO		0.0	5.0	1.0	1.0	7.0	0.0	14.3	
71.4	14.3								
MKE-DCA		0.0	1.0	0.0	4.0	5.0	0.0	0.0	
20.0	80.0								
ATL-BOI		0.0	2.0	0.0	0.0	2.0	0.0	0.0	
100.0	0.0								
DCA-CMH		0.0	1.0	2.0	1.0	4.0	0.0	50.0	
25.0	25.0								
TOL-CVG		0.0	0.0	0.0	1.0	1.0	0.0	0.0	
0.0	100.0								

[2463 rows x 9 columns]

I anem a excloure les rutes que tenen pocs viatge (Per davall del percentil 5).

```
[57]: route_delay_df.Total.quantile(0.1)
route_delay_df_q01 = route_delay_df[route_delay_df.Total >= route_delay_df.
    ↪Total.quantile(0.1)]

route_delay_df_q01.sort_values(by='Sense retard %', ascending=False)
```

```
[57]:
```

	Sense retard	Mitjà	Petit	Gran	Total	Sense retard %	Petit %	Mitjà
% Gran %								
PDX-CVG	9.0	0.0	2.0	0.0	11.0	81.8	18.2	
0.0 0.0								
ROC-DFW	6.0	0.0	0.0	2.0	8.0	75.0	0.0	
0.0 25.0								
TUL-SAT	5.0	1.0	1.0	0.0	7.0	71.4	14.3	
14.3 0.0								
SLC-BNA	5.0	0.0	2.0	0.0	7.0	71.4	28.6	
0.0 0.0								
MSP-LIT	5.0	1.0	1.0	0.0	7.0	71.4	14.3	
14.3 0.0								
...
...
STL-AUS	0.0	2.0	2.0	3.0	7.0	0.0	28.6	
28.6 42.9								
OAK-ORD	0.0	2.0	2.0	4.0	8.0	0.0	25.0	
25.0 50.0								
GSP-MCO	0.0	5.0	1.0	1.0	7.0	0.0	14.3	
71.4 14.3								
EWB-JAN	0.0	2.0	3.0	2.0	7.0	0.0	42.9	
28.6 28.6								
JFK-IND	0.0	0.0	2.0	5.0	7.0	0.0	28.6	
0.0 71.4								

[2235 rows x 9 columns]

Quines aerolínies pateixen més retards?

```
[58]: # Fem un diccionari per clasificar els retards en cada una de les rutes.
carrier_delay_dict = {}
for carrier in delay_df['air_carrier_names'].unique():
    subset = delay_df[delay_df['air_carrier_names'] == carrier]
    carrier_delay_count = subset['ArrDelay_Class'].value_counts()
    carrier_delay_dict[carrier] = carrier_delay_count

# Calcularem els percentatges
```

```

carrier_delay_df = pd.DataFrame.from_dict(data=carrier_delay_dict,
    ↳orient='index')
carrier_delay_df['Total'] = carrier_delay_df['Sense retard'] +
    ↳carrier_delay_df['Petit'] + carrier_delay_df['Mitjà'] +
    ↳carrier_delay_df['Gran']
carrier_delay_df['Sense retard %'] = round(carrier_delay_df['Sense retard'] /
    ↳carrier_delay_df['Total'] * 100, 1)
carrier_delay_df['Petit %'] = round(carrier_delay_df['Petit'] /
    ↳carrier_delay_df['Total'] * 100, 1)
carrier_delay_df['Mitjà %'] = round(carrier_delay_df['Mitjà'] /
    ↳carrier_delay_df['Total'] * 100, 1)
carrier_delay_df['Gran %'] = round(carrier_delay_df['Gran'] /
    ↳carrier_delay_df['Total'] * 100, 1)

carrier_delay_df.sort_values(by='Sense retard %', ascending=False)

```

[58]:

	Sense retard	Petit	Mitjà	Gran	Total	Sense
retard %	Petit %	Mitjà %	Gran %			
Mapjet	35	17	8	8	68	
51.5	25.0	11.8	11.8			
Southwest Airlines	17962	8079	7105	5643	38789	
46.3	20.8	18.3	14.5			
Frontier Airlines, Inc.	1259	784	535	387	2965	
42.5	26.4	18.0	13.1			
Hawaiian Airlines Inc.	319	212	139	95	765	
41.7	27.7	18.2	12.4			
Continental Air Lines	4329	1807	1897	2435	10468	
41.4	17.3	18.1	23.3			
US Airways	3872	2099	2089	2005	10065	
38.5	20.9	20.8	19.9			
Alaska Airlines	1537	914	878	707	4036	
38.1	22.6	21.8	17.5			
Delta Air Lines, Inc.	4304	2655	2362	2465	11786	
36.5	22.5	20.0	20.9			
AirTran	2447	1633	1506	1709	7295	
33.5	22.4	20.6	23.4			
United Airlines, Inc.	4668	2614	2990	3953	14225	
32.8	18.4	21.0	27.8			
Skywest Airlines	4405	2779	2747	3587	13518	
32.6	20.6	20.3	26.5			
Envoy Air	4502	2983	3398	3558	14441	
31.2	20.7	23.5	24.6			
Endeavor Air	1649	1086	1214	1368	5317	
31.0	20.4	22.8	25.7			
Northwest Airlines	2474	1817	2032	1687	8010	
30.9	22.7	25.4	21.1			
American Airlines	6130	4081	4501	5116	19828	

30.9	20.6	22.7	25.8					
ExpressJet Airlines				2548	1821	1838	2156	8363
30.5	21.8	22.0	25.8					
Jetblue Airways Corporation				1710	930	1191	1797	5628
30.4	16.5	21.2	31.9					
JSX				3137	2033	2324	3002	10496
29.9	19.4	22.1	28.6					
Jetstream Intl				1371	1163	1347	1484	5365
25.6	21.7	25.1	27.7					
Mesa Airlines, Inc.				1632	1426	1534	2192	6784
24.1	21.0	22.6	32.3					

Hi han avions que tenen més retards que d'altres?

```
[59]: tailnum_delay_dict = {}
for tailnum in delay_df['TailNum'].unique():
    subset = delay_df[delay_df['TailNum'] == tailnum]
    tailnum_delay_count = subset['ArrDelay_Class'].value_counts()
    tailnum_delay_dict[tailnum] = tailnum_delay_count

tailnum_delay_df = pd.DataFrame.from_dict(data=tailnum_delay_dict,
    orient='index')
tailnum_delay_df.fillna(0)
tailnum_delay_df['Total'] = tailnum_delay_df['Sense retard'] +
    tailnum_delay_df['Petit'] + tailnum_delay_df['Mitjà'] +
    tailnum_delay_df['Gran']
tailnum_delay_df['Sense retard %'] = round(tailnum_delay_df['Sense retard'] /
    tailnum_delay_df['Total'] * 100, 1)
tailnum_delay_df['Petit %'] = round(tailnum_delay_df['Petit'] /
    tailnum_delay_df['Total'] * 100, 1)
tailnum_delay_df['Mitjà %'] = round(tailnum_delay_df['Mitjà'] /
    tailnum_delay_df['Total'] * 100, 1)
tailnum_delay_df['Gran %'] = round(tailnum_delay_df['Gran'] /
    tailnum_delay_df['Total'] * 100, 1)

tailnum_delay_df.sort_values(by='Sense retard %', ascending=False)
```

```
[59]:
```

	Sense retard	Petit	Mitjà	Gran	Total	Sense retard %	Petit %	Mitjà
N87512	17.0	2.0	2.0	1.0	22.0	77.3	9.1	
N27239	26.0	3.0	1.0	7.0	37.0	70.3	8.1	
N41135	7.0	1.0	1.0	1.0	10.0	70.0	10.0	
N77865	23.0	6.0	3.0	1.0	33.0	69.7	18.2	

9.1	3.0							
N456UW		22.0	3.0	2.0	5.0	32.0	68.8	9.4
6.2	15.6							
...		
...	...							
N792UA		NaN	NaN	NaN	1.0	NaN	NaN	NaN
NaN	NaN							
N808NW		NaN	NaN	NaN	1.0	NaN	NaN	NaN
NaN	NaN							
N270AY		NaN	NaN	NaN	1.0	NaN	NaN	NaN
NaN	NaN							
N278AY		NaN	NaN	NaN	1.0	NaN	NaN	NaN
NaN	NaN							
N823AL		NaN	NaN	NaN	1.0	NaN	NaN	NaN
NaN	NaN							

[5267 rows x 9 columns]

Contem sols els retards degut a l'aerolínia, ja que alguns d'aquests poden ser degut a manteniment, neteja del propi avió.

```
[60]: delay_df.loc[delay_df.CarrierDelay < 15, 'CarrierDelay_Class'] = 'Sense retard'
delay_df.loc[(delay_df.CarrierDelay >= 15) & (delay_df.ArrDelay < 30),
↳ 'CarrierDelay_Class'] = 'Petit'
delay_df.loc[(delay_df.CarrierDelay >= 30) & (delay_df.ArrDelay < 60),
↳ 'CarrierDelay_Class'] = 'Mitjà'
delay_df.loc[delay_df.CarrierDelay > 60, 'CarrierDelay_Class'] = 'Gran'

tailnum_delay_carrier_dict = {}
for tailnum in delay_df['TailNum'].unique():
    subset = delay_df[delay_df['TailNum'] == tailnum]
    tailnum_delay_carrier_count = subset['CarrierDelay_Class'].value_counts()
    tailnum_delay_carrier_dict[tailnum] = tailnum_delay_carrier_count

tailnum_delay_carrier_df = pd.DataFrame.
↳ from_dict(data=tailnum_delay_carrier_dict, orient='index')
tailnum_delay_carrier_df.fillna(0)
tailnum_delay_carrier_df['Total'] = tailnum_delay_carrier_df['Sense retard'] +
↳ tailnum_delay_carrier_df['Petit'] + tailnum_delay_carrier_df['Mitjà'] +
↳ tailnum_delay_carrier_df['Gran']
tailnum_delay_carrier_df['Sense retard %'] =
↳ round(tailnum_delay_carrier_df['Sense retard'] /
↳ tailnum_delay_carrier_df['Total'] * 100, 1)
tailnum_delay_carrier_df['Petit %'] = round(tailnum_delay_carrier_df['Petit'] /
↳ tailnum_delay_carrier_df['Total'] * 100, 1)
tailnum_delay_carrier_df['Mitjà %'] = round(tailnum_delay_carrier_df['Mitjà'] /
↳ tailnum_delay_carrier_df['Total'] * 100, 1)
```

```
tailnum_delay_carrier_df['Gran %'] = round(tailnum_delay_carrier_df['Gran'] /
↳tailnum_delay_carrier_df['Total'] * 100, 1)

tailnum_delay_carrier_df.sort_values(by='Sense retard %', ascending=False)
```

```
[60]:
```

	Sense retard	Gran	Petit	Mitjà	Total	Sense retard %	Petit %	Mitjà
% Gran %								
N399WN	47.0	1.0	1.0	1.0	50.0	94.0	2.0	
2.0	2.0							
N16911	43.0	1.0	1.0	1.0	46.0	93.5	2.2	
2.2	2.2							
N921AT	37.0	1.0	1.0	1.0	40.0	92.5	2.5	
2.5	2.5							
N311SW	49.0	1.0	2.0	1.0	53.0	92.5	3.8	
1.9	1.9							
N952AT	34.0	1.0	1.0	1.0	37.0	91.9	2.7	
2.7	2.7							
...	
...	...							
N1603	NaN	NaN	NaN	1.0	NaN	NaN	NaN	
NaN	NaN							
N185DN	NaN	NaN	NaN	1.0	NaN	NaN	NaN	
NaN	NaN							
N120UA	NaN	NaN	NaN	1.0	NaN	NaN	NaN	
NaN	NaN							
N739AL	NaN	NaN	NaN	1.0	NaN	NaN	NaN	
NaN	NaN							
N273AY	NaN	NaN	NaN	1.0	NaN	NaN	NaN	
NaN	NaN							

[5196 rows x 9 columns]

I quins són els motius principals dels retards? Cercarem per cada vol, quin ha sigut el motiu més important del retard

```
[61]: # Crearem un diccionari per calcular quantes vegades cada un dels motius ha
↳sigut el que més minuts de retard ha sumat en cada vol.

columns_req = ['CarrierDelay', 'WeatherDelay', 'NASDelay', 'SecurityDelay',
↳'LateAircraftDelay']
main_reason_delay = dict.fromkeys(columns_req, 0)

# Ens caldrà els índexs al que correspon cada columna en el dataframe, per si
↳al llarg del treball canviem l'ordre. Aquí trobem el índex de manera
↳automàtica.
col_req_index = []
for column in columns_req:
```

```
lst = delay_df.columns == column
col_req_index.append([i for i, x in enumerate(lst) if x][0])
```

```
[62]: main_reason_delay
```

```
[62]: {'CarrierDelay': 0,
      'WeatherDelay': 0,
      'NASDelay': 0,
      'SecurityDelay': 0,
      'LateAircraftDelay': 0}
```

```
{'CarrierDelay': 0, 'WeatherDelay': 0, 'NASDelay': 0, 'SecurityDelay': 0, 'LateAircraftDelay': 0}
```

```
[63]: # Iterem per les files cercant quin motiu acumula més retard en cada vol

for flight in delay_df.iterrows():
    min_delays = [] # Llista per guardar els minuts de cada motiu de retard en
    ↪ un vol concret
    for item in col_req_index:
        min_delays.append(flight[1][item])
    max_index = [] # Trobarem quina posició (o quines posicions) té el retard
    ↪ més gran
    max_value = min_delays[0] # max_value guarda el primer valor i els
    ↪ comparará amb la resta per cercar el més alt (o els més alts si coincideix
    ↪ en més d'un motiu)
    for i, val in ((i, val) for i, val in enumerate(min_delays) if val >=
    ↪ max_value):
        if val == max_value:
            max_index.append(i)
        else:
            max_val = max_value
            max_index = [i]
    for maxi in max_index: # Comptem en el diccionari el principal motiu
    ↪ traduint el índex al motiu
        if maxi == 0:
            main_reason_delay['CarrierDelay'] += 1
        elif maxi == 1:
            main_reason_delay['WeatherDelay'] += 1
        elif maxi == 2:
            main_reason_delay['NASDelay'] += 1
        elif maxi == 3:
            main_reason_delay['SecurityDelay'] += 1
        elif maxi == 4:
            main_reason_delay['LateAircraftDelay'] += 1
```

```
[64]: main_reason_delay
```

```
[64]: {'CarrierDelay': 41939,
      'WeatherDelay': 2423,
      'NASDelay': 23403,
      'SecurityDelay': 17141,
      'LateAircraftDelay': 80845}
```

Quina és la velocitat mitjana dels trajectes?

```
[65]: delay_df['MeanSpeed'] = round((delay_df['Distance'] / delay_df['AirTime']) * (1.
    ↪ 609344 * 60), 1)
```

```
[66]: delay_df
```

```
[66]:      Date  ArrTime  CRSArrTime UniqueCarrier TailNum  AirTime  ArrDelay
DepDelay Origin Dest  Distance  TaxiIn  TaxiOut  Cancelled  CancellationCode
CarrierDelay WeatherDelay NASDelay SecurityDelay LateAircraftDelay
air_carrier_names      Route ArrDelay_Class RouteUnique CarrierDelay_Class
MeanSpeed
1750755 2008-12-19   1436.0      14.30          WN  N700GS      49.0        6.0
12.0    CLE  BWI      314      7.0      8.0          0          N
NaN          NaN      NaN          NaN          NaN      Southwest
Airlines  CLE-BWI  Sense retard      CLE-BWI          NaN      618.8
1074983 2008-06-17   2013.0      19.55          CO  N54241     316.0       18.0
11.0    EWR  LAS     2227      8.0     18.0          0          N
11.0          0.0      7.0          0.0          0.0  Continental Air
Lines  EWR-LAS      Petit      EWR-LAS      Sense retard      680.5
46126   2008-01-27   918.0      8.15          YV  N27185     54.0       63.0
50.0    ORD  BNA      409      4.0     45.0          0          N
63.0          0.0      0.0          0.0          0.0  Mesa Airlines,
Inc.  ORD-BNA      Gran      ORD-BNA      Gran      731.4
327441 2008-02-13   1515.0      14.38          9E  87189E     80.0       37.0
63.0    ORF  DTW      529      7.0     15.0          0          N
0.0          0.0      0.0          0.0          37.0      Endeavor
Air  ORF-DTW      Mitjà      ORF-DTW      Sense retard      638.5
1363177 2008-08-12   1517.0      14.47          FL  N267AT     178.0       30.0
13.0    ATL  DEN     1199      5.0     15.0          0          N
13.0          0.0     17.0          0.0          0.0
AirTran  ATL-DEN      Mitjà      ATL-DEN      Sense retard      650.4
...      ...      ...      ...      ...      ...      ...
...      ...      ...      ...      ...      ...      ...
...      ...      ...      ...      ...      ...      ...
...      ...      ...      ...      ...      ...      ...
1595898 2008-10-27   112.0      22.25          MQ  N643MQ     109.0      167.0
159.0    ORD  BDL      783      4.0     20.0          0          N
0.0          0.0      8.0          0.0          159.0      Envoy
Air  ORD-BDL      Gran      BDL-ORD      Sense retard      693.6
1503090 2008-09-25   115.0      22.45          AA  N433AA     115.0      150.0
82.0    ORD  LGA      733     34.0     54.0          0          N
```


0.0	0.0	150.0	0.0	0.0	American
Airlines	ORD-LGA	Gran	ORD-LGA	Sense retard	615.5
742602	2008-05-16	1422.0	14.15	WN N316SW	37.0 7.0
10.0	OKC DAL	181 3.0	7.0	0	N
NaN	NaN	NaN	NaN	NaN	Southwest
Airlines	OKC-DAL	Sense retard	DAL-OKC	NaN	472.4
1840527	2008-12-23	839.0	8.31	DL N996DL	50.0 8.0
10.0	CLT ATL	227 9.0	20.0	0	N
NaN	NaN	NaN	NaN	NaN	Delta Air Lines,
Inc.	CLT-ATL	Sense retard	CLT-ATL	NaN	438.4
1526456	2008-10-06	2217.0	22.05	WN N675AA	185.0 12.0
7.0	PHX MDW	1444 3.0	12.0	0	N
NaN	NaN	NaN	NaN	NaN	Southwest
Airlines	PHX-MDW	Sense retard	PHX-MDW	NaN	753.7

[200000 rows x 26 columns]

Quin és el percentatge i el motiu principal de cancel·lació? (A = Aerolinia; B = Temps; C = Retards en el Sistema d'espai aeri nacional que inclou condicions meteorològiques no extremes, operacions en l'aeroport, alt volum de tràfic, etc.; D = Seguretat)

```
[67]: print('Percentatge de vols cancel·lats: ' + str((len(delay_df[delay_df.
      ↪Cancelled == 1]) / len(delay_df))*100) + '%')

reason_cancel = {'A':'Aerolinia', 'B':'Temps', 'C':'Retards Sistema Espai_
      ↪Nacional (meteo no extrama, tràfic)', 'D':'Seguretat'}
cancod_delay_df = delay_df[delay_df.CancellationCode != 'N']['CancellationCode']
round(cancod_delay_df.value_counts() / len(cancod_delay_df)*100, 1)
cancod_delay_count = cancod_delay_df.value_counts()

for i in range(len(cancod_delay_count)):
    reason = reason_cancel[cancod_delay_count.index[i]]
    perc = cancod_delay_count[i]
    print(reason + ' = ' + str(perc) + '%')
```

Percentatge de vols cancel·lats: 0.0385%

Temps = 35%

Aerolinia = 32%

Retards Sistema Espai Nacional (meteo no extrama, tràfic) = 10%

La distància té un efecte important en els retards? Podem cercar la correlació que hi ha entre el camp distància i el retard a l'arribada. La correlació però és petita, això està relacionat amb el fet amb que els principals motius dels retards venen donats dels retards que es donen a la sortida.

```
[68]: print(np.corrcoef(delay_df_notna['ArrDelay'], delay_df_notna['Distance'])[0][1])
```

-0.028558250136643933

```
[69]: print(np.corrcoef(delay_df_notna['ArrDelay'], delay_df_notna['DepDelay'])[0][1])
```

0.9529689239333934

Comprovem també que hi ha més correlació positiva amb el temps de TaxiOut, o siga el temps que està a l'avió en circulació en terra en l'aeroport abans d'enlairar, que en el TaxiIn. No es estrany que es produïsquen cues en els avions alhora de sortir, sobretot després de una tempesta o situacions de baixa visibilitat que redueix l'operativa de l'aeroport.

```
[70]: print(np.corrcoef(delay_df_notna['ArrDelay'], delay_df_notna['TaxiIn'])[0][1])
```

0.15798720203518202

```
[71]: print(np.corrcoef(delay_df_notna['ArrDelay'], delay_df_notna['TaxiOut'])[0][1])
```

0.28615974071560346