Pràctica 2: Neteja i anàlisi de les dades

1. Descripció del dataset

Aquest dataset és el resultat obtingut a la Pràctica 1 – Web scraping, recordem que el web scraper desenvolupat emmagatzemava en un arxiu csv les informacions relatives a totes les operacions, és a dir, tant arribades com sortides, ocorregudes a l'aeroport de Barcelona – El Prat el dia anterior, que és quan es pot assegurar que totes les operacions han tingut lloc. El link on està publicat el codi és el següent:
https://github.com/GuillemTD/BCN_Airport_scraping

Aquesta activitat ha estat realitzada de manera individual per Guillem Tudela Debrigode.

En aquest dataset hi ha emmagatzemades les següents dades:

- Unnamed: Número de fila
- Scheduled Date: Data del vol programada.
- City: Nom de la ciutat d'origen o de destinació.
- IATA: Nom de l'aeroport d'origen o destinació en codi IATA, és un codi de tres lletres associat a cada aeroport.
- Airline: Nom de la companyia que opera la ruta.
- Flight: Número de vol de la ruta. Les dues primeres lletres estan associades a l'aerolínia (p.ex. IB per Ibèria, o VY per Vuèling) i els números posteriors a la ruta (aeroport d'origen, de destinació i horari). Una ruta operada un altre dia conserva el número de vol.
- Operation: Tipus d'operació, és a dir, si és sortida o arribada.
- Scheduled Time: Hora programada de l'operació.
- Terminal: Terminal de l'aeroport on es realitza l'operació, pot ser 1 o 2.
- Status: Status de l'operació, aterrat, en ruta, cancel·lat...
- Real Date: Data real de l'operació (si l'avió aterra abans d'hora o surt amb molt de retard pot ser que no coincideixi el dia programat amb el real).
- Real Time: Hora real de l'operació. La diferència entre aquesta columna i la "Scheduled Time" permeten calcular el retard del vol.
- Gate: Porta de la terminal des d'on s'ha embarcat (informació només disponible per a les operacions de sortides i en cas que l'avió hagi aterrat).

Aquest conjunt de dades és important perquè ens informa sobre el número d'operacions de l'aeroport, les hores previstes de l'operació i les reals, informant alhora sobre la terminal, el tipus d'operació, la companyia aèria i l'aeroport de destinació o origen, segons el cas. Així doncs, aplicant algunes transformacions, aquest mateix dataset permet determinar la puntualitat dels vols a l'aeroport de Barcelona en un dia determinat i fins i tot investigar la relació entre número d'operacions, moment del dia i retards.

La fusió dels datasets diaris durant un període prolongat de temps (per exemple, mesos) ens permetria realitzar un anàlisi sobre els retards més complex, podent arribar a diferenciar entre retards associats al propi aeroport o a factors externs, tals com vagues o mal temps. Per a fer-ho, però, seria interessant la fusió d'aquest conjunt de dades amb algun d'extern amb informació relativa a les dades de vent a l'aeroport de Barcelona per franges horàries. Una altra possibilitat que ofereix aquest dataset és, en l'hipotètic cas que es tingués una sèrie temporal superior a l'any, seria la d'analitzar la temporalitat de les operacions.

Donades les restriccions temporals de la pròpia pràctica la sèrie temporal es limita a una sola setmana d'operacions, degut a la naturalesa del web scraper desenvolupat a la primera pràctica, això implicarà la càrrega de set fitxers.

En aquesta pràctica s'estableixen tres objectius, i per cadascun dels quals es duran a terme transformacions diferents sobre el conjunt de dades per a assolir-los. El primer objectiu és conèixer fins a quin punt les dades de les operacions estan influenciades pel dia de la setmana, és a dir, si hi ha temporalitat, cicles... El segon objectiu és determinar si existeix alguna relació entre el retard i altres característiques del vol, com la terminal, el tipus d'operació, la franja horària, l'acumulació d'operacions... Amb aquestes dades intentarem respondre a les preguntes: "Existeixen diferències significatives entre el retard mitjà a la terminal 1 i la 2?", "A les franges de més afluència hi ha més retards?", "Els retards a les sortides està relacionat amb el retard acumulat a les arribades?". El tercer i darrer objectiu és crear un model de regressió que permeti predir el retard d'una operació basant-nos amb les dades observades durant la setmana. En aquest cas només es tindran en compte els vols operats per l'empresa "Vueling", donat que és l'operadora número 1 de l'aeroport i, per tant, es disposa de més dades que en el cas d'altres empreses.

2. Integració i selecció

Cada dataset, en ser un arxiu depenent del número d'operacions, variarà segons el dia en el número de files però no en el de columnes, que com ja s'ha comentat en l'anterior punt, n'hi ha un total de 13. A continuació es pot veure el número de files de cada arxiu. Veiem com el primer, el cinquè i el darrer són els arxius amb més operacions, que es corresponen als dies dilluns, divendres i diumenge.

```
# Variació del número de files de l'arxiu.
for i in range(0,len(filename)):
    print('Nom arxiu: %s\tNúm files: %d' % (filename[i], len_df[i]))
```

```
Nom arxiu: BCN_Operations_2019-12-09.csv        Núm files: 2267
Nom arxiu: BCN_Operations_2019-12-10.csv        Núm files: 1774
Nom arxiu: BCN_Operations_2019-12-11.csv        Núm files: 1788
Nom arxiu: BCN_Operations_2019-12-12.csv        Núm files: 1937
Nom arxiu: BCN_Operations_2019-12-13.csv        Núm files: 2251
Nom arxiu: BCN_Operations_2019-12-14.csv        Núm files: 1649
Nom arxiu: BCN_Operations_2019-12-15.csv        Núm files: 2129
```

El conjunt de dades carregat té el següent aspecte.

```
# Exploració de les dades.
df_week.head(5)
```

	Unnamed: 0	Scheduled Date	City	IATA	Airline	Flight	Operation	Scheduled Time	Terminal	Status	Real Date	Real Time	Gate
0	0	2019-12-09	Buenos Aires	(EZE)	Iberia	IB2601	Departure	01:30	1.0	Landed - On-time	2019-12-09	01:23	D16
1	1	2019-12-09	Palma Mallorca	(PMI)	Swiftair	WT112	Departure	04:30	NaN	En Route	2019-12-09	04:30	-
2	2	2019-12-09	Algiers	(ALG)	Vueling	VY7476	Departure	06:00	1.0	Landed - On-time	2019-12-09	05:52	E73
3	3	2019-12-09	Algiers	(ALG)	Iberia	IB5682	Departure	06:00	1.0	Landed - On-time	2019-12-09	05:52	E73
4	4	2019-12-09	Amsterdam	(AMS)	KLM	KL1662	Departure	06:00	1.0	Landed - On-time	2019-12-09	06:04	B32

Podem observar amb quin format ha carregat les dades.

```
# Exploració de les dades.
df_week.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 13795 entries, 0 to 2128
Data columns (total 13 columns):
Unnamed: 0        13795 non-null int64
Scheduled Date    13795 non-null object
City              13795 non-null object
IATA              13795 non-null object
Airline           13795 non-null object
Flight            13795 non-null object
Operation         13795 non-null object
Scheduled Time    13795 non-null object
Terminal          13728 non-null float64
Status            13795 non-null object
Real Date         13795 non-null object
Real Time         13795 non-null object
Gate              13795 non-null object
dtypes: float64(1), int64(1), object(11)
memory usage: 1.8+ MB
```

Observem que la columna “Terminal” conté valors absents i que, a més, ha estat convertida erròniament a float, quan hauria de ser un string. La resta de columnes no contenen valors absents i tenen el format desitjat.

Crearem un nou dataset que contingui exclusivament les dades de la companyia “Vueling”, aquest conjunt serà necessari per al model de regressió, serà recuperat més endavant. En total s’han registrat 2002 operacions durant tota la setmana.

```
# Creació d'un arxiu exclusiu amb les dades de Vueling.
df_vueling = df_week[df_week.Airline=='Vueling ']
len(df_vueling)
```

```
2002
```

Abans de procedir cal introduir el concepte d’operació amb codi compartit. A l’aeroport de Barcelona no es van produir al llarg de la setmana 2000 operacions de mitjana, tal i com indica el número de files dels arxius, si això fos així, implicaria, amb la suposició que hi ha tantes operacions de sortida com d’arribada i que l’aeroport opera durant 18 hores al dia, que cada 30 segons s’enlairaria o aterraria un avió ininterrompudament al llarg de tot el dia. Això són números propis d’aeroports tals com Heathrow. Per què tenim tantes operacions? Per l’anomenat codi compartit, és a dir, dues o més aerolínies, normalment membres d’una mateixa aliança o del mateix consorci, venen bitllets per un trajecte, tot i que a l’hora de la veritat només hi ha un sol avió. A continuació es mostra un exemple de codi compartit per múltiples companyies.

2019-12-09	Frankfurt	(FRA)	Lufthansa	LH1139	Departure	06:00	1.0	Landed - On-time	2019-12-09	05:58	C72
2019-12-09	Frankfurt	(FRA)	Air Canada	AC9183	Departure	06:00	1.0	Landed - On-time	2019-12-09	05:58	C72
2019-12-09	Frankfurt	(FRA)	SAS	SK3391	Departure	06:00	1.0	Landed - On-time	2019-12-09	05:58	C72
2019-12-09	Frankfurt	(FRA)	United Airlines	UA9194	Departure	06:00	1.0	Landed - On-time	2019-12-09	05:58	C72
2019-12-09	Frankfurt	(FRA)	Oman Air	WY5539	Departure	06:00	1.0	Landed - On-time	2019-12-09	05:58	C72

Aquests tipus d’operacions són fàcils de detectar perquè comparteixen el dia i hora de vol (tant programada com real), el número de terminal, la destinació i la porta. Així doncs, el primer pas serà considerar que estem davant de casos de duplicació de dades i ens quedarem amb una sola operació, la primera, que de fet és la que realitza l’operació realment. Per a fer-ho eliminarem les columnes “Unnamed”, “Airline” i “Flight” que igualment per al tipus d’estudi que volem realitzar no seran necessàries. Això ens permet reduir el dataset de 13795 a 5328 components (dels quals gairebé un 40% operats per Vueling).

```
# Codi compartit - eliminació duplicats
columnes_drop = ['Unnamed: 0', 'Airline', 'Flight']
df_week_ = df_week.drop(columnes_drop, axis=1)
df_week_ = pd.DataFrame.drop_duplicates(df_week_, keep='first').reset_index(drop=True)
df_week_.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5328 entries, 0 to 5327
Data columns (total 10 columns):
Scheduled Date    5328 non-null object
City              5328 non-null object
IATA              5328 non-null object
Operation         5328 non-null object
Scheduled Time    5328 non-null object
Terminal          5273 non-null float64
Status            5328 non-null object
Real Date         5328 non-null object
Real Time         5328 non-null object
Gate              5328 non-null object
dtypes: float64(1), object(9)
memory usage: 416.3+ KB
```

La primera transformació a realitzar consisteix en calcular la diferència entre l'hora real i l'hora programada per tal d'obtenir la variable "Delay", la qual és la base per a la majoria dels càlculs d'aquesta pràctica. Per a fer-ho transformem la data i l'hora a timestamp i calculem la diferència entre ambdós valors. Per tal de facilitar la interpretació convertirem el valor de segons a minuts i ho guardem en una nova columna anomenada "Delay".

```
# Càlcul delay - Diferència entre schedule i real.
# llista on guardar el delay.
delay = []
for i in range(0,len(df_week_)):
    # obtenció del dia i hora de l'operació.
    sch_date_i = df_week_.iloc[i, df_week_.columns.get_loc("Scheduled Date")]
    sch_time_i = df_week_.iloc[i, df_week_.columns.get_loc("Scheduled Time")]
    real_date_i = df_week_.iloc[i, df_week_.columns.get_loc("Real Date")]
    real_time_i = df_week_.iloc[i, df_week_.columns.get_loc("Real Time")]
    #timestamp de l'hora.
    sch_i = datetime.timestamp(datetime.strptime(sch_date_i + ' ' + sch_time_i, '%Y-%m-%d %H:%M'))
    real_i = datetime.timestamp(datetime.strptime(real_date_i + ' ' + real_time_i, '%Y-%m-%d %H:%M'))

    if (sch_i==real_i):
        delay.append(0)
    else:
        delay.append(-((sch_i-real_i)/60))

# adjuntem els valors de delay al dataframe.
df_week_['Delay'] = delay
df_week_.head()
```

	Scheduled Date	City	IATA	Operation	Scheduled Time	Terminal	Status	Real Date	Real Time	Gate	Delay
0	2019-12-09	Buenos Aires	(EZE)	Departure	01:30	1.0	Landed - On-time	2019-12-09	01:23	D16	-7.0
1	2019-12-09	Palma Mallorca	(PMI)	Departure	04:30	NaN	En Route	2019-12-09	04:30	-	0.0
2	2019-12-09	Algiers	(ALG)	Departure	06:00	1.0	Landed - On-time	2019-12-09	05:52	E73	-8.0
3	2019-12-09	Amsterdam	(AMS)	Departure	06:00	1.0	Landed - On-time	2019-12-09	06:04	B32	4.0
4	2019-12-09	Frankfurt	(FRA)	Departure	06:00	1.0	Landed - On-time	2019-12-09	05:58	C72	-2.0

Observem els resultats obtinguts.

```
# Exploració de les dades.
df_week_.Delay.describe()
```

```
count    5328.000000
mean        8.073386
std        53.655116
min      -185.000000
25%        -6.000000
50%         0.000000
75%        11.000000
max      1410.000000
Name: Delay, dtype: float64
```

Els resultats negatius impliquen que l'avió va aterrar o es va enlairar abans de l'hora programada, mentre que els valors positius impliquen un retard en l'operació. En el següent apartat, durant l'anàlisi de valors extrems, analitzarem amb més detall els valors obtinguts per si hi hagués alguna observació que pogués entrar dintre d'aquesta categoria.

La segona transformació consistirà en convertir les hores en franges horàries, així agruparem els vols i els assignarem una variable comuna. S'ha decidit que cada hora tindrà la seva pròpia franja, p.ex. els vols programats per a les 13 tindran assignada la franja número 13. El resultat és el següent.

```
# Divisió en blocs horaris segons l'schedule
# llista on guardar el bloc horari.
bloc = []
for i in range(0,len(df_week_)):
    bloc.append(int((df_week_.iloc[i, df_week_.columns.get_loc("Scheduled Time")])[0:2]))

# adjuntem els valors del bloc horari al dataframe.
df_week_['Time block'] = bloc
df_week_.head()
```

	Scheduled Date	City	IATA	Operation	Scheduled Time	Terminal	Status	Real Date	Real Time	Gate	Delay	Time block
0	2019-12-09	Buenos Aires	(EZE)	Departure	01:30	1.0	Landed - On-time	2019-12-09	01:23	D16	-7.0	1
1	2019-12-09	Palma Mallorca	(PMI)	Departure	04:30	NaN	En Route	2019-12-09	04:30	-	0.0	4
2	2019-12-09	Algiers	(ALG)	Departure	06:00	1.0	Landed - On-time	2019-12-09	05:52	E73	-8.0	6
3	2019-12-09	Amsterdam	(AMS)	Departure	06:00	1.0	Landed - On-time	2019-12-09	06:04	B32	4.0	6
4	2019-12-09	Frankfurt	(FRA)	Departure	06:00	1.0	Landed - On-time	2019-12-09	05:58	C72	-2.0	6

Crearem una variable auxiliar, anomenada Time, que ens ajudarà a distribuir les dades temporals en un gràfic. Aquesta variable es basa en el bloc horari i en el dia de l'operació, si es tracta del primer arxiu temporal Time és idèntic al valor de bloc, un cop hagi passat un dia afegirà el valor 24 a bloc, quant n'hagin passat dos, afegirà 48 i així successivament fins que al setè dia Time serà equivalent a afegir 144 al bloc. D'aquesta manera tenim ordenades les operacions de la setmana.

```
# Creació variable Time
Time = []

# Obtenció dels valors únics com a llista dels dies de la setmana.
time=np.unique(list((df_week_['Scheduled Date'])))
for i in range(0,len(df_week_)):
    for j in range(0,len(time)):
        if (df_week_.iloc[i, df_week_.columns.get_loc("Scheduled Date")]==time[j]):
            Time.append(df_week_.iloc[i, df_week_.columns.get_loc("Time block")] + j*24)

# adjuntem els valors al dataframe
df_week_['Time'] = Time
df_week_.tail()
```

	Scheduled Date	City	IATA	Operation	Scheduled Time	Terminal	Status	Real Date	Real Time	Gate	Delay	Time block	Time
5323	2019-12-15	La Coruna	(LCG)	Arrival	23:59	1.0	Landed - Delayed	2019-12-16	00:27	-	28.0	23	167
5324	2019-12-15	London	(LGW)	Arrival	23:59	1.0	Landed - Delayed	2019-12-16	00:15	-	16.0	23	167
5325	2019-12-15	Naples	(NAP)	Arrival	23:59	1.0	Landed - On-time	2019-12-16	00:01	-	2.0	23	167
5326	2019-12-15	Rennes	(RNS)	Arrival	23:59	1.0	Landed - On-time	2019-12-15	23:56	-	-3.0	23	167
5327	2019-12-15	Venice	(VCE)	Arrival	23:59	2.0	Landed - Delayed	2019-12-16	00:14	-	15.0	23	167

La resta de transformacions pendents, com la de convertir els valors de Terminal a string o la d'eliminar les columnes que ja no ens interessen seran realitzades un cop es dugui a terme la neteja de dades.

3. Neteja de les dades

Hem vist que el nostre dataset conté una columna amb valors absents.

```
# Valors NaN
df_week_[pd.isnull(df_week_.Terminal)].head(10)
```

	Scheduled Date	City	IATA	Operation	Scheduled Time	Terminal	Status	Real Date	Real Time	Gate	Delay	Time block	Time
1	2019-12-09	Palma Mallorca	(PMI)	Departure	04:30	NaN	En Route	2019-12-09	04:30	-	0.0	4	4
65	2019-12-09	Palma Mallorca	(PMI)	Departure	07:45	NaN	Unknown	2019-12-09	07:45	-	0.0	7	7
128	2019-12-09	Bilbao	(BIO)	Departure	10:10	NaN	Unknown	2019-12-09	10:10	B25	0.0	10	10
285	2019-12-09	Palma Mallorca	(PMI)	Departure	16:10	NaN	Landed - Delayed	2019-12-09	18:05	B41	115.0	16	16
292	2019-12-09	Palma Mallorca	(PMI)	Departure	16:30	NaN	Landed - On-time	2019-12-09	16:33	-	3.0	16	16
429	2019-12-09	Marseille	(MRS)	Departure	21:50	NaN	Landed	2019-12-09	21:50	-	0.0	21	21
438	2019-12-09	Ibiza	(IBZ)	Departure	22:55	NaN	Unknown	2019-12-09	22:55	-	0.0	22	22
458	2019-12-09	Liege	(LGG)	Arrival	04:05	NaN	Landed	2019-12-09	04:05	-	0.0	4	4
475	2019-12-09	Barcelona	(BCN)	Arrival	08:15	NaN	Diverted	2019-12-09	09:13	-	58.0	8	8
665	2019-12-09	London	(LGW)	Arrival	14:45	NaN	Diverted	2019-12-09	17:26	-	161.0	14	14

Observem que les posicions tenen diferents Status i afecta tant a sortides com arribades. Ens és impossible determinar amb exactitud l'origen de l'error però tot apunta a que prové del propi scraping que fa la pàgina d'on s'han extret les dades quan duu a terme la pròpia captura de dades. Per aquest motiu, i tenint en compte que hi ha poques observacions afectades (5273 de 5328 observacions a la variable Terminal són no-nuls, per tant, 55, que equival a un 1% de les mostres), es decideix eliminar-los del dataset i no tenir-los en compte.

```
# Eliminació dels valors absents
index_nan = list(df_week_[pd.isnull(df_week_.Terminal)].index)
df_week_ = df_week_.drop(index=index_nan).reset_index(drop=True)
df_week_[pd.isnull(df_week_.Terminal)]
```

Scheduled Date	City	IATA	Operation	Scheduled Time	Terminal	Status	Real Date	Real Time	Gate	Delay	Time block	Time

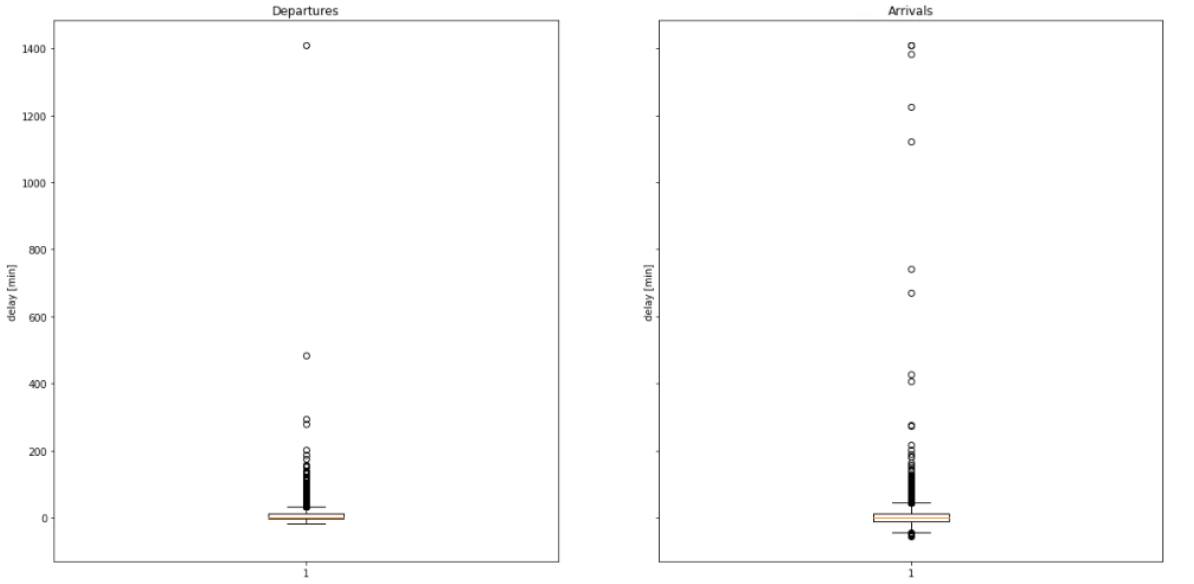
Un cop eliminats aquests registres transformem la columna "Terminal" a string.

```
# Canvi Terminal float to string
for i in range(0, len(df_week_)):
    df_week_.iloc[i,df_week_.columns.get_loc("Terminal")] = \
    str(df_week_.iloc[i,df_week_.columns.get_loc("Terminal")])[0]
df_week_.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5273 entries, 0 to 5272
Data columns (total 13 columns):
Scheduled Date    5273 non-null object
City              5273 non-null object
IATA              5273 non-null object
Operation         5273 non-null object
Scheduled Time    5273 non-null object
Terminal          5273 non-null object
Status            5273 non-null object
Real Date         5273 non-null object
Real Time         5273 non-null object
Gate              5273 non-null object
Delay             5273 non-null float64
Time block        5273 non-null int64
Time              5273 non-null int64
dtypes: float64(1), int64(2), object(10)
memory usage: 535.6+ KB
```

Anem a tractar ara els valors extrems. Els únics valors numèrics del dataset són els del "Delay". Comprovarem mitjançant un boxplot la presència d'aquest tipus de valors, diferenciant entre les operacions de sortida i les d'arribada.

```
# Valors outliers.
fig, (ax1, ax2) = plt.subplots(1, 2, sharey = True, figsize=(20,10))
ax1.boxplot(x = df_week_['Delay'][df_week_.Operation == 'Departure'])
ax1.set(title='Departures', ylabel='delay [min]')
ax2.boxplot(x = df_week_['Delay'][df_week_.Operation == 'Arrival'])
ax2.set(title='Arrivals', ylabel='delay [min]')
plt.show()
```



Es fa difícil de determinar el valor exacte, però sembla que tot el que sigui per sobre de 40 minuts ho considera com si es tractés de valors extrems. Això no hauria de sorprendre perquè ens indica que la majoria dels vols són puntuals. Observem que hi ha valors molt extrems, de 1400 minuts de retard, que equival a 23h entre hora programada i hora real, però que la majoria dels "outliers" es mouen en el rang de valors menors a 180 minuts.

Considerant que les dades són reals i que no hi ha hagut cap error durant el procés d'scraping , no es veu cap necessitat de prescindir d'aquestes observacions per a l'anàlisi. Els orígens dels retards en els avions són múltiples, poden ser degut a problemes mecànics de l'aeronau, de congestió de l'espai aeri, problemes de capacitat a pista, vagues de controladors, meteorologia adversa... per tant no es considera que sigui un error de la captura ni tampoc adient eliminar aquests valors, així doncs, s'inclouran en l'anàlisi.

Analitzem els valors de "Status" per si ens indiqués la presència de valors extrems o inconsistències i no els hàgim sabut interpretar.

```
# Comptem per Status.
Counter(df_week_.Status)
```

```
Counter({' Landed - On-time': 4067,
         ' Landed': 23,
         ' Landed - Delayed': 1087,
         ' Diverted': 8,
         ' Unknown': 30,
         ' En Route - Delayed': 6,
         ' Canceled': 45,
         ' En Route - On-time': 5,
         ' Scheduled - Delayed': 2})
```

Nota: Només hi ha Landed perquè en el cas dels vols d'operació "Departure" el dataset ens indica si aquest ha arribat a la seva destinació o no.

Analitzarem els casos "Unknown", "Canceled" i "Diverted", que són els que presenten a priori una possible presència d'inconsistències.

Amb "Status" Unknown no detectem cap problema amb el "Delay".

```
# Tractament altres valors
df_week_[df_week_.Status == " Unknown"].head(10)
```

	Scheduled Date	City	IATA	Operation	Scheduled Time	Terminal	Status	Real Date	Real Time	Gate	Delay	Time block	Time
75	2019-12-09	Timisoara	(TSR)	Departure	08:00	2	Unknown	2019-12-09	08:00	W46	0.0	8	8
95	2019-12-09	Algiers	(ALG)	Departure	09:20	1	Unknown	2019-12-09	09:28	D11	8.0	9	9
103	2019-12-09	Debrecen	(DEB)	Departure	09:35	2	Unknown	2019-12-09	09:42	S25	7.0	9	9
312	2019-12-09	Cluj-Napoca	(CLJ)	Departure	17:35	2	Unknown	2019-12-09	17:41	Y58	6.0	17	17
331	2019-12-09	Istanbul	(ISL)	Departure	18:10	1	Unknown	2019-12-09	18:25	D4	15.0	18	18
332	2019-12-09	Kutaisi	(KUT)	Departure	18:10	2	Unknown	2019-12-09	18:26	W46	16.0	18	18
410	2019-12-09	Skopje	(SKP)	Departure	21:15	2	Unknown	2019-12-09	21:17	W46	2.0	21	21
1175	2019-12-10	Katowice	(KTW)	Departure	20:40	2	Unknown	2019-12-10	20:42	U36	2.0	20	44
1184	2019-12-10	Casablanca	(CMN)	Departure	21:20	2	Unknown	2019-12-10	21:29	Y54	9.0	21	45
1622	2019-12-11	Casablanca	(CMN)	Departure	10:10	2	Unknown	2019-12-11	10:09	Y55	-1.0	10	58

Amb Canceled, veiem que el “Delay” és 0, un valor que no sembla concordar amb la realitat.

```
# Tractament altres valors
df_week_[df_week_.Status == " Canceled"].head(10)
```

	Scheduled Date	City	IATA	Operation	Scheduled Time	Terminal	Status	Real Date	Real Time	Gate	Delay	Time block	Time
522	2019-12-09	Bilbao	(BIO)	Arrival	10:05	1	Canceled	2019-12-09	10:05	-	0.0	10	10
850	2019-12-09	Ibiza	(IBZ)	Arrival	23:05	1	Canceled	2019-12-09	23:05	-	0.0	23	23
974	2019-12-10	London	(LHR)	Departure	10:10	1	Canceled	2019-12-10	10:10	-	0.0	10	34
1024	2019-12-10	London	(LHR)	Departure	12:20	1	Canceled	2019-12-10	12:20	-	0.0	12	36
1049	2019-12-10	London	(LGW)	Departure	14:10	2	Canceled	2019-12-10	14:10	-	0.0	14	38
1095	2019-12-10	Milan	(MXP)	Departure	16:50	2	Canceled	2019-12-10	16:50	-	0.0	16	40
1124	2019-12-10	Paris	(CDG)	Departure	18:10	2	Canceled	2019-12-10	18:10	-	0.0	18	42
1135	2019-12-10	Basel, Switzerland / Mulhouse	(BSL)	Departure	18:45	2	Canceled	2019-12-10	18:45	-	0.0	18	42
1142	2019-12-10	Lyon	(LYS)	Departure	19:00	2	Canceled	2019-12-10	19:00	-	0.0	19	43
1148	2019-12-10	Paris	(ORY)	Departure	19:20	1	Canceled	2019-12-10	19:20	-	0.0	19	43

I amb Diverted tampoc detectem cap problema amb el “Delay”.

```
# Tractament altres valors
df_week_[df_week_.Status == " Diverted"].head(10)
```

	Scheduled Date	City	IATA	Operation	Scheduled Time	Terminal	Status	Real Date	Real Time	Gate	Delay	Time block	Time
31	2019-12-09	Bilbao	(BIO)	Departure	07:00	1	Diverted	2019-12-09	07:03	A1	3.0	7	7
237	2019-12-09	Palma Mallorca	(PMI)	Departure	14:15	1	Diverted	2019-12-09	14:14	B43	-1.0	14	14
392	2019-12-09	Ibiza	(IBZ)	Departure	20:30	1	Diverted	2019-12-09	20:38	B47	8.0	20	20
1867	2019-12-11	Buenos Aires	(EZE)	Arrival	05:30	1	Diverted	2019-12-11	05:30	-	0.0	5	53
2029	2019-12-11	London	(LTN)	Arrival	15:35	2	Diverted	2019-12-11	16:53	-	78.0	15	63
2964	2019-12-13	Vigo	(VGO)	Departure	07:20	1	Diverted	2019-12-13	07:23	B51	3.0	7	103
3052	2019-12-13	Newark	(EWR)	Departure	11:00	1	Diverted	2019-12-13	11:12	D8	12.0	11	107
3900	2019-12-14	Bristol	(BRS)	Departure	10:55	2	Diverted	2019-12-14	10:50	M6	-5.0	10	130

Dels tres “Status” analitzats, el de Canceled és l’únic que requereix prendre una decisió. Una de les possibilitats és trobar quan es repeteix el mateix vol i assignar com a retard la diferència de dies entre ambdues dates. Normalment els vols tenen una freqüència diària, però n’hi ha que són setmanals i això implicaria assignar com a retard valors que van des

dels 1440 fins als 10000 minuts. Una altra possibilitat és eliminar aquests vols i no tenir-los en compte en l'anàlisi.

Tenint en compte la complexitat per determinar la freqüència de cada vol (per exemple, caldria recuperar el conjunt de dades original, on encara hi havia la columna de l'aerolínia i, eliminant la data, comptar quants cops apareix cada vol), que caldria suposar que cada vol que no es repeteix té una freqüència setmanal (i per tant un retard de 10000 minuts) i que només tenen aquesta condició 45 observacions (inferior al 1%), es decideix que aquests vols seran eliminats i no es tindran en compte per l'anàlisi.

```
# Eliminació de les observacions amb Status Canceled
index_can = list(df_week_[df_week_.Status == " Canceled"].index)
df_week_ = df_week_.drop(index=index_can).reset_index(drop=True)
df_week_[df_week_.Status == " Canceled"]
```

Scheduled Date	City	IATA	Operation	Scheduled Time	Terminal	Status	Real Date	Real Time	Gate	Delay	Time block	Time

Recuperem en aquest punt el dataframe amb les dades de Vueling per aplicar els mateixos processos de transformació i neteja que hem aplicat en el dataframe amb totes les dades. Calcularem el retard i el bloc horari, enlloc de la variable "Time" en crearem una que s'anomeni "Day" on assignarem quin dia de la setmana (dilluns 1, diumenge 7) ha operat el vol. També s'eliminaran les observacions amb valors absents i aquelles amb "Status" Canceled.

Finalment, el darrer pas és eliminar les columnes innecessàries segons l'anàlisi a realitzar. Per al primer anàlisi, el de la temporalitat, sols en fa falta la columna de "Operation", "Delay" i "Time". Per al segon anàlisi, requerirem de les columnes "Operation", "Time block", "Delay" i "Terminal". Per al darrer anàlisi, requerirem les columnes "City", "Operation", "Time block", "Day" i "Delay".

```
df_analisi_1 = pd.DataFrame(data = df_week_[['Time', 'Operation', 'Delay']])
df_analisi_2 = pd.DataFrame(data = df_week_[['Time block', 'Operation', 'Terminal', 'Delay']])
df_analisi_3 = pd.DataFrame(data = df_vueling[['City', 'Operation', 'Time block', 'Delay']])
```

Procedim a emmagatzemar les dades dels tres conjunts.

4. Anàlisi de les dades

Per a cada conjunt de dades, a excepció del tercer, aplicarem algun tipus de transformació extra.

Per al primer anàlisi cal que es vegi per a cada moment del dia quantes operacions hi ha hagut, tenint sempre en compte el dia de la setmana. Per a fer-ho caldrà aplicar un groupby.

```
# Apliquem el groupby al df_analisi_1
df_analisi_1_gb = df_analisi_1.groupby(['Time'])\
                              .agg({'Delay': ('mean', 'count')}).reset_index()
df_analisi_1_gb.columns = ['Time', 'Mean', 'Total op']
df_analisi_1_gb.head()
```

	Time	Mean	Total op
0	0	97.533333	15
1	1	-17.666667	3
2	5	-24.000000	1
3	6	22.000000	32
4	7	0.369565	46

Donat que hi ha hores del dia sense operacions cal aplicar una altra transformació per a afegir aquestes posicions.

```
# Afegim les hores sense operacions.
time_nou = np.arange(0,24*7,1).tolist()
time_nou_df = pd.DataFrame(data=time_nou,columns=['Time'])
df_analisi_1_gb_ = pd.merge(time_nou_df, df_analisi_1_gb, how='outer', on='Time')
df_analisi_1_gb_ = df_analisi_1_gb_.fillna(0)
df_analisi_1_gb_.head()
```

	Time	Mean	Total op
0	0	97.533333	15.0
1	1	-17.666667	3.0
2	2	0.000000	0.0
3	3	0.000000	0.0
4	4	0.000000	0.0

Com que per al segon anàlisi es desitja determinar les influències del retard amb la resta de valors és necessari agrupar les dades basant-nos en les franges horàries, així es pot obtenir per cada combinació de terminal-operació-franja el número d'operacions i el retard mitjà.

Per tant, s'aplicarà un groupby per a agrupar les operacions segons aquestes característiques.

```
# Apliquem el groupby al df_analisi_2
df_analisi_2_gb = df_analisi_2.groupby(['Time block', 'Operation', 'Terminal'])\
                              .agg({'Delay': ('sum', 'mean'),
                               'Time block': 'count'}).reset_index()
df_analisi_2_gb.columns = ['Time block', 'Operation', 'Terminal', 'Sum delay', 'Mean', 'Total op']
df_analisi_2_gb.head()
```

	Time block	Operation	Terminal	Sum delay	Mean	Total op
0	0	Arrival	1	1440.0	36.923077	39
1	0	Arrival	2	3921.0	392.100000	10
2	1	Arrival	1	556.0	37.066667	15
3	1	Departure	1	-13.0	-1.857143	7
4	2	Arrival	1	-31.0	-31.000000	1

Guardem les dues transformacions.

4.1. Anàlisi de la normalitat i l'homoscedasticitat:

Ha arribat el moment d'analitzar l'homogeneïtat de les dades. Tal i com s'ha construït el model podem considerar que existeixen quatre grups, en funció de la combinació de tipus d'operació i de terminal i l'atribut que caldrà analitzar si segueix una distribució normal o no és el del retard, en conseqüència, cal dur a terme aquesta prova per a cadascun d'aquests grups.

S'aplicaran dues proves, una inspecció visual (les gràfiques sobre la distribució del retard segons cada grup estan ubicades a l'apartat 5.) i el test de Shapiro-Wilk. La hipòtesi nul·la consisteix en afirmar que la població segueix una distribució normal, en cas que el p-valor sigui menor al nivell de significació (amb valor estàndard 0.05), rebutgem la hipòtesi nul·la i no podem afirmar que les dades es distribueixen seguint la normal, en cas contrari, sí que ho podrem afirmar.

```
# Comprovació de la normalitat de les dades
# Test de Shapiro-Wilk
resultat = stats.shapiro(df_analisi_2['Delay'][(df_analisi_2['Operation']=='Departure')
                                               & (df_analisi_2['Terminal']=='1')])
print("Shapiro-Wilk, Departures T1: %.15f %.2E" % resultat)
resultat = stats.shapiro(df_analisi_2['Delay'][(df_analisi_2['Operation']=='Departure')
                                               & (df_analisi_2['Terminal']=='2')])
print("Shapiro-Wilk, Departures T2: %.15f %.2E" % resultat)
resultat = stats.shapiro(df_analisi_2['Delay'][(df_analisi_2['Operation']=='Arrival')
                                               & (df_analisi_2['Terminal']=='1')])
print("Shapiro-Wilk, Arrival T1: %.15f %.2E" % resultat)
resultat = stats.shapiro(df_analisi_2['Delay'][(df_analisi_2['Operation']=='Arrival')
                                               & (df_analisi_2['Terminal']=='2')])
print("Shapiro-Wilk, Arrival T2: %.15f %.2E" % resultat)
```

```
Shapiro-Wilk, Departures T1: 0.544930458068848 0.00E+00
Shapiro-Wilk, Departures T2: 0.240927159786224 0.00E+00
Shapiro-Wilk, Arrival T1: 0.320438981056213 0.00E+00
Shapiro-Wilk, Arrival T2: 0.220502853393555 0.00E+00
```

En els quatre grups obtenim un p-valor de 0, així doncs, segons el test de Shapiro-Wilk les dades no segueixen la distribució normal.

Si no tinguéssim en compte els valors amb un retard superior a 60 minuts, el resultat tampoc varia.

```
# Eliminem els valors superiors a 60
resultat = stats.shapiro(df_analisi_2['Delay'][(df_analisi_2['Operation']=='Departure')
                                               & (df_analisi_2['Terminal']=='1') & (df_analisi_2['Delay']<60)])
print("Shapiro-Wilk, Departures T1: %.15f %.2E" % resultat)
resultat = stats.shapiro(df_analisi_2['Delay'][(df_analisi_2['Operation']=='Departure')
                                               & (df_analisi_2['Terminal']=='2') & (df_analisi_2['Delay']<60)])
print("Shapiro-Wilk, Departures T2: %.15f %.2E" % resultat)
resultat = stats.shapiro(df_analisi_2['Delay'][(df_analisi_2['Operation']=='Arrival')
                                               & (df_analisi_2['Terminal']=='1') & (df_analisi_2['Delay']<60)])
print("Shapiro-Wilk, Arrival T1: %.15f %.2E" % resultat)
resultat = stats.shapiro(df_analisi_2['Delay'][(df_analisi_2['Operation']=='Arrival')
                                               & (df_analisi_2['Terminal']=='2') & (df_analisi_2['Delay']<60)])
print("Shapiro-Wilk, Arrival T2: %.15f %.2E" % resultat)
```

```
Shapiro-Wilk, Departures T1: 0.839293301105499 1.52E-38
Shapiro-Wilk, Departures T2: 0.870436728000641 2.61E-25
Shapiro-Wilk, Arrival T1: 0.963418483734131 2.78E-20
Shapiro-Wilk, Arrival T2: 0.936766326427460 5.72E-18
```

L'alternativa que ens queda és la de normalitzar les dades del retard i aplicar el test, però com es veu a continuació, tampoc varia el resultat.

```
# Comprovació de la normalitat amb valors normalitzats
from sklearn import preprocessing
normalitzat = preprocessing.normalize([np.array(df_analisi_2['Delay'][(df_analisi_2['Operation']=='Departure')
                                               & (df_analisi_2['Terminal']=='1') & (df_analisi_2['Delay']<60)])])
resultat = stats.shapiro(normalitzat)
print("Shapiro-Wilk, Departures T1: %.15f %.2E" % resultat)
normalitzat = preprocessing.normalize([np.array(df_analisi_2['Delay'][(df_analisi_2['Operation']=='Departure')
                                               & (df_analisi_2['Terminal']=='2') & (df_analisi_2['Delay']<60)])])
resultat = stats.shapiro(normalitzat)
print("Shapiro-Wilk, Departures T2: %.15f %.2E" % resultat)
normalitzat = preprocessing.normalize([np.array(df_analisi_2['Delay'][(df_analisi_2['Operation']=='Arrival')
                                               & (df_analisi_2['Terminal']=='1') & (df_analisi_2['Delay']<60)])])
resultat = stats.shapiro(normalitzat)
print("Shapiro-Wilk, Arrival T1: %.15f %.2E" % resultat)
normalitzat = preprocessing.normalize([np.array(df_analisi_2['Delay'][(df_analisi_2['Operation']=='Arrival')
                                               & (df_analisi_2['Terminal']=='2') & (df_analisi_2['Delay']<60)])])
resultat = stats.shapiro(normalitzat)
print("Shapiro-Wilk, Arrival T2: %.15f %.2E" % resultat)
```

```
Shapiro-Wilk, Departures T1: 0.839293301105499 1.52E-38
Shapiro-Wilk, Departures T2: 0.870436728000641 2.61E-25
Shapiro-Wilk, Arrival T1: 0.963418483734131 2.78E-20
Shapiro-Wilk, Arrival T2: 0.936766326427460 5.72E-18
```

Tot i el resultat del test en les seves diferents variants, aplicant el teorema del límit central podem considerar que el retard segueix una distribució normal perquè tenim un conjunt de mostres suficientment gran.

Per a comprovar l'homoscedasticitat de la variable "Delay" emprarem el test de Fligner-Killen perquè la distribució de les dades no és normal. Cal comparar els quatre grups, és a dir, a

igual tipus d'operació si hi ha diferències entre els subconjunts de terminals i a igual terminal si hi ha diferències entre els subconjunts de tipus d'operació. La hipòtesi nul·la afirma que els grups comparats tenen la mateixa variància, és a dir, homoscedasticitat.

```
# Test de Fligner-Killeen comparant terminals quan l'operació és sortida
stats.fligner(df_analisi_2['Delay'][(df_analisi_2['Operation']=='Departure')
                                    & (df_analisi_2['Terminal']=='1')],
              df_analisi_2['Delay'][(df_analisi_2['Operation']=='Departure')
                                    & (df_analisi_2['Terminal']=='2')])
```

```
FlignerResult(statistic=39.39034430916152, pvalue=3.470086377626322e-10)
```

```
# Test de Fligner-Killeen comparant terminals quan l'operació és arribada
stats.fligner(df_analisi_2['Delay'][(df_analisi_2['Operation']=='Arrival')
                                    & (df_analisi_2['Terminal']=='1')],
              df_analisi_2['Delay'][(df_analisi_2['Operation']=='Arrival')
                                    & (df_analisi_2['Terminal']=='2')])
```

```
FlignerResult(statistic=8.160807775135117, pvalue=0.0042805301969872835)
```

```
# Test de Fligner-Killeen comparant operacions quan la terminal és la 1
stats.fligner(df_analisi_2['Delay'][(df_analisi_2['Operation']=='Departure')
                                    & (df_analisi_2['Terminal']=='1')],
              df_analisi_2['Delay'][(df_analisi_2['Operation']=='Arrival')
                                    & (df_analisi_2['Terminal']=='1')])
```

```
FlignerResult(statistic=156.6036857414893, pvalue=6.248136656660037e-36)
```

```
# Test de Fligner-Killeen comparant operacions quan la terminal és la 2
stats.fligner(df_analisi_2['Delay'][(df_analisi_2['Operation']=='Departure')
                                    & (df_analisi_2['Terminal']=='2')],
              df_analisi_2['Delay'][(df_analisi_2['Operation']=='Arrival')
                                    & (df_analisi_2['Terminal']=='2')])
```

```
FlignerResult(statistic=33.53588251049654, pvalue=6.996132855837335e-09)
```

El resultat en els quatre casos és un p-valor menor al nivell de significació, de 0.05 així doncs, rebutgem la hipòtesi nul·la i determinem que els grups tenen heteroscedasticitat.
els quatre grups no presenten homoscedasticitat i no hi ha homogeneïtat entre les variàncies.

Comprovem què passaria si eliminéssim els valors superiors a 60.

```
# Test de Fligner-Killeen comparant terminals quan l'operació és sortida eliminant els valors superiors a 60
stats.fligner(df_analisi_2['Delay'][(df_analisi_2['Operation']=='Departure')
                                    & (df_analisi_2['Terminal']=='1') & (df_analisi_2['Delay']<60)],
              df_analisi_2['Delay'][(df_analisi_2['Operation']=='Departure')
                                    & (df_analisi_2['Terminal']=='2') & (df_analisi_2['Delay']<60)])
```

```
FlignerResult(statistic=14.915646852710454, pvalue=0.00011242663476618398)
```

```
# Test de Fligner-Killeen comparant terminals quan l'operació és arribada eliminant els valors superiors a 60
stats.fligner(df_analisi_2['Delay'][(df_analisi_2['Operation']=='Arrival')
                                    & (df_analisi_2['Terminal']=='1') & (df_analisi_2['Delay']<60)],
              df_analisi_2['Delay'][(df_analisi_2['Operation']=='Arrival')
                                    & (df_analisi_2['Terminal']=='2') & (df_analisi_2['Delay']<60)])
```

```
FlignerResult(statistic=2.6158721901969426, pvalue=0.10579933956279498)
```

```
# Test de Fligner-Killeen comparant operacions quan la terminal és la 1 eliminant els valors superiors a 60
stats.fligner(df_analisi_2['Delay'][(df_analisi_2['Operation']=='Departure')
                                    & (df_analisi_2['Terminal']=='1') & (df_analisi_2['Delay']<60)],
              df_analisi_2['Delay'][(df_analisi_2['Operation']=='Arrival')
                                    & (df_analisi_2['Terminal']=='1') & (df_analisi_2['Delay']<60)])
```

```
FlignerResult(statistic=158.18759954038828, pvalue=2.8161105840774073e-36)
```

```
# Test de Fligner-Killeen comparant operacions quan la terminal és la 2 eliminant els valors superiors a 60
stats.fligner(df_analisi_2['Delay'][(df_analisi_2['Operation']=='Departure')
                                    & (df_analisi_2['Terminal']=='2') & (df_analisi_2['Delay']<60)],
              df_analisi_2['Delay'][(df_analisi_2['Operation']=='Arrival')
                                    & (df_analisi_2['Terminal']=='2') & (df_analisi_2['Delay']<60)])
```

```
FlignerResult(statistic=43.32008634922436, pvalue=4.6478770033058414e-11)
```

En aquest cas, sí que hi hauria homogeneïtat de variàncies entre el subconjunt d'arribades de la terminal 1 i de la terminal 2, en els altres es mantindria l'heteroscedasticitat.

La no homoscedasticitat entre els grups tindrà conseqüències en el tipus de test que podrem aplicar durant l'anàlisi 2.

4.2. Anàlisi temporal

Tot i la limitació de les dades a una sola setmana, realitzarem un anàlisi temporal, aplicant el test de Dickey-Fuller per determinar si la sèrie de dades és o no estacionària, addicionalment provarem de trobar un cicle i el seu corresponent patró en cas que les dades depenguin del temps.

La hipòtesi nul·la del test de Dickey-Fuller suggereix que les dades no són estacionàries, és a dir, que la seva estructura depèn del temps. La hipòtesi alternativa, en canvi, suggereix que la sèrie és estacionària, per tant, que no hi ha dependència amb el temps.

```
# ANÀLISI TEMPORAL - Test de Dickey-Fuller
adfuller(df_analisi_1_gb_['Total op'])
```

```
(-6.561665066451608,
 8.35160134044587e-09,
 13,
 154,
 {'1%': -3.473542528196209,
  '5%': -2.880497674144038,
  '10%': -2.576878053634677},
 1083.8533558979905)
```

Aplicant el test de Dickey-Fuller s'obté un p-valor de pràcticament 0, així doncs, les dades no depenen del component temps. Això és degut amb tota probabilitat a que la sèrie és massa curta.

La representació de les operacions al llarg de lla setmana, així com el resultat gràfic de la cerca del cicle i del patró (tasca que tot i el resultat del test s'ha dut a terme) es poden veure a l'apartat 5.

4.3. Contrast d'hipòtesis

En aquest apartat es determinarà si existeixen diferències significatives entre els subconjunts de terminals i entre les operacions. Per a fer-ho s'usarà el dataset de l'anàlisi 2. Com que a l'inici d'aquest apartat s'ha determinat que les dades presenten heteroscedasticitat, no podrem aplicar el test de t-Student, sinó que serà necessari aplicar el de Mann-Whitney.

Compararem els subconjunts diferenciant per tipus d'operació. En aquest cas, la hipòtesi nul·la afirma que no existeixen diferències significatives entre els valors del retard de la sortida i de l'arribada.

```
# CONTRAST D'HIPÒTESI - Test de Mann-Whitney
# Entre Operacions
stats.mannwhitneyu(df_analisi_2['Delay'][(df_analisi_2['Operation']=='Departure')],
            df_analisi_2['Delay'][(df_analisi_2['Operation']=='Arrival')])
```

```
MannwhitneyuResult(statistic=2737461.0, pvalue=7.051422010417602e-36)
```

Obtenim un p-valor de pràcticament 0 i, per tant, rebutgem la hipòtesi nul·la i es conclou que sí que existeixen diferències estadísticament significatives entre els valors del retard e cada subconjunt.

Apliquem per segona vegada el test, però aquest cop per analitzar els subconjunts tenint en compte la terminal.

```
# Entre Terminals
stats.mannwhitneyu(df_analisi_2['Delay'][(df_analisi_2['Terminal']=='1')],
            df_analisi_2['Delay'][(df_analisi_2['Terminal']=='2')])
```

```
MannwhitneyuResult(statistic=2824150.5, pvalue=0.0002145726841019686)
```

En aquest segon cas també rebutgem la hipòtesi nul·la. Com en el cas anterior, sí que hi ha diferències significatives entre ambdós subconjunts. Així doncs, considerar que es tenen quatre grups separats segons el tipus d'operació i la terminal és correcte.

4.3. Correlació

En aquest apartat es vol determinar si existeix una correlació entre el retard mitjà i el número d'operacions, aquesta relació ens permet veure si un aeroport està operant al màxim de la seva capacitat, és a dir, un component del retard seria de tipus intern, mentre que en cas contrari, ens indicaria que el retard és d'origen extern. Com que les dades tenen heteroscedasticitat no s'aplica la correlació de Pearson sinó la de Spearman.

```
# Correlació entre el delay i les operacions.
rho, pval = stats.spearmanr(df_analisi_2_gb)
print(df_analisi_2_gb.columns)
print(rho)
```

```
Index(['Time block', 'Operation', 'Terminal', 'Sum delay', 'Mean', 'Total op'], dtype='object')
[[ 1.          0.08061433  0.11231994  0.20540812  0.21697863  0.21333797]
 [ 0.08061433  1.          0.04375219  0.27299439  0.26271427  0.11367548]
 [ 0.11231994  0.04375219  1.          0.08300447  0.31656187 -0.42369474]
 [ 0.20540812  0.27299439  0.08300447  1.          0.8817891   0.34359942]
 [ 0.21697863  0.26271427  0.31656187  0.8817891   1.          0.01818803]
 [ 0.21333797  0.11367548 -0.42369474  0.34359942  0.01818803  1.        ]]
```

Com podem veure a la taula anterior, la relació entre retard mitjà i número total d'operacions en aquell bloc temporal és de 0.21, és a dir estan directament (poc) relacionades. Per tant, l'origen del retard és degut principalment a factors externs.

4.4. Model de regressió

El darrer anàlisi consistirà en construir un model de regressió, a partir de les dades dels vols de Vueling, que ens permeti predir el retard que patirà una ruta en funció del seu bloc horari i del tipus d'operació. Usarem el conjunt de dades específic amb els vols de Vueling.

```
# Construcció d'un model de regressió.
# Usarem el OneHotEncoder per a convertir les dades categòriques a numèriques.
ohe = OneHotEncoder(categories='auto',sparse=False)

# Apliquem el OneHotEncoder
X_ohe_cat = ohe.fit_transform(np.column_stack((df_analisi_3[['City']],df_analisi_3[['Operation']])))
X_ohe_num = df_analisi_3[['Time block']]
X_ohe = pd.concat([pd.DataFrame(X_ohe_cat), X_ohe_num], axis=1)

# Construïm el model de regressió.
lr = LinearRegression()

lr.fit(X_ohe, df_analisi_3[['Delay']])
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

Construït el model es poden realitzar prediccions, per exemple, un vol que arriba de València a hora prevista d'aterratge a les 23 de la nit arribarà uns 7 minuts abans del planificat.

```
# Predicció 1.
PRED = pd.DataFrame()
PRED['HORA'] = [23]

predicio = pd.concat([pd.DataFrame(ohe.transform([['Valencia', 'Arrival']])), PRED], axis=1)

lr.predict(predicio)
```

```
array([[-7.26171875]])
```

El que va a París, en canvi, sortirà 8 minuts tard.

```
# Predicció 2.
PRED = pd.DataFrame()
PRED['HORA'] = [16]

predicio = pd.concat([pd.DataFrame(ohe.transform([['Paris', 'Departure']])), PRED], axis=1)

lr.predict(predicio)
```

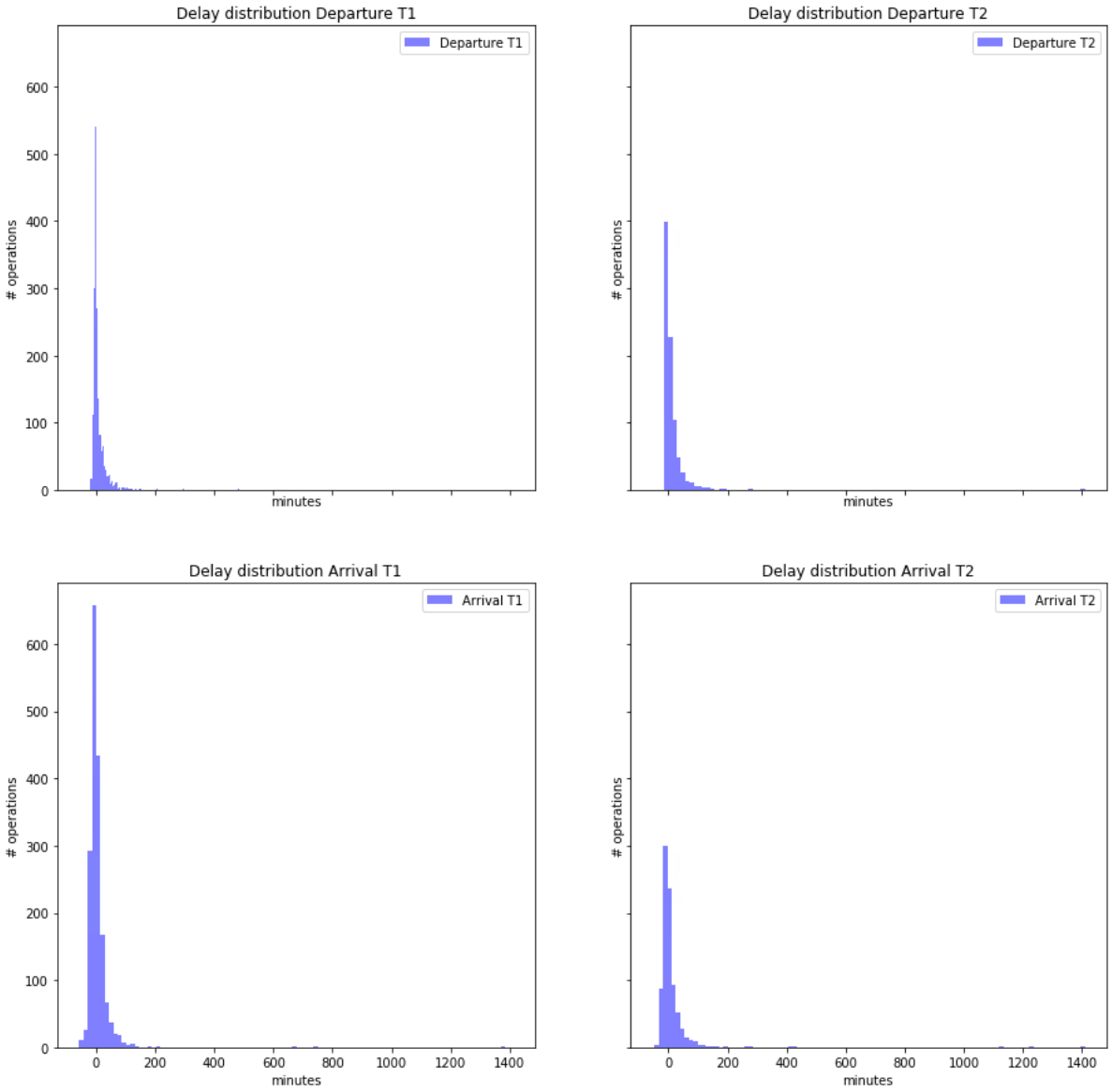
```
array([[8.31933594]])
```

Per descomptat, com més dades es disposi, millor serà el model.

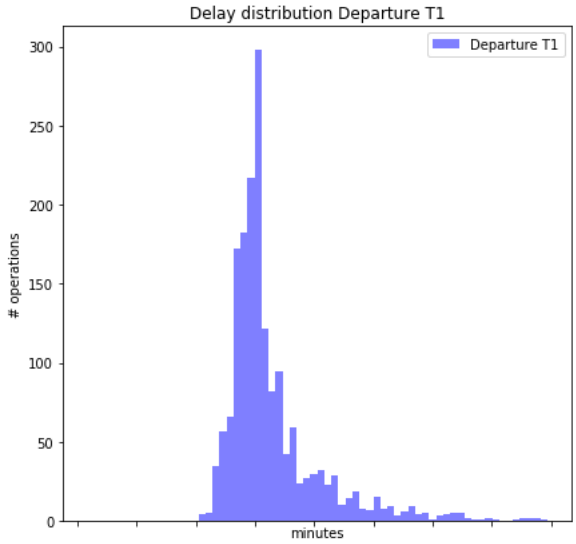
Com a alternativa es podria crear un altre model que enlloc de predir els minuts de retard ho fes amb variables categòriques (arriba puntual, abans d'hora, amb retard menor de 30 minuts...) en aquest projecte, però, no es durà a terme aquesta variant.

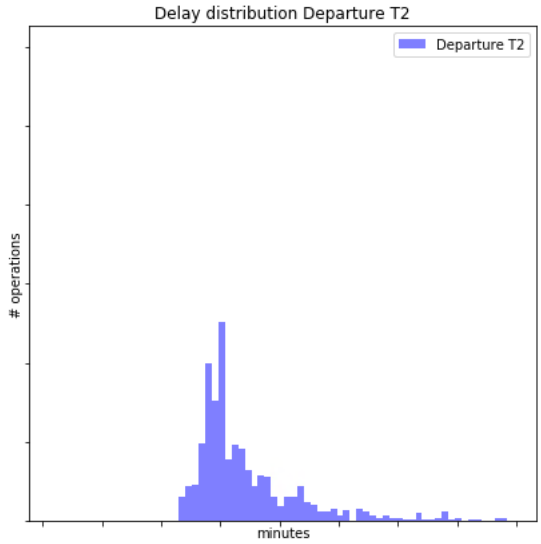
5. Representació dels resultats

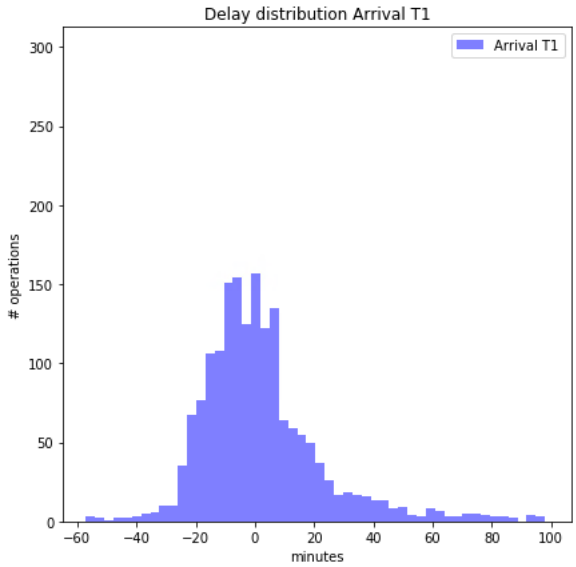
A continuació es mostra la distribució del retard per cadascun dels subgrups.

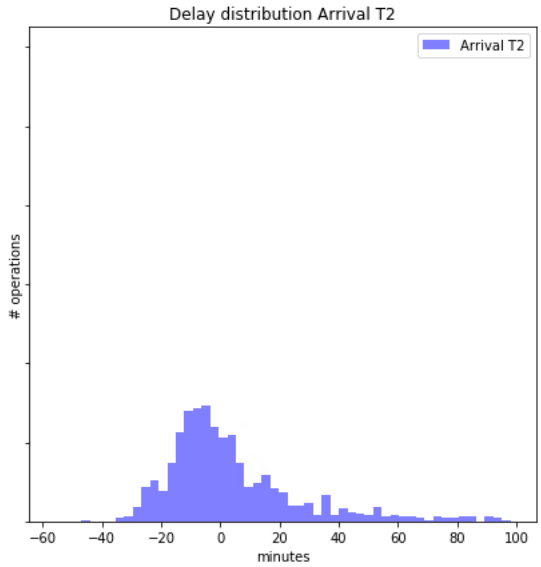


Si no representem els valors superiors a 100 minuts s'aprecia millor la distribució.



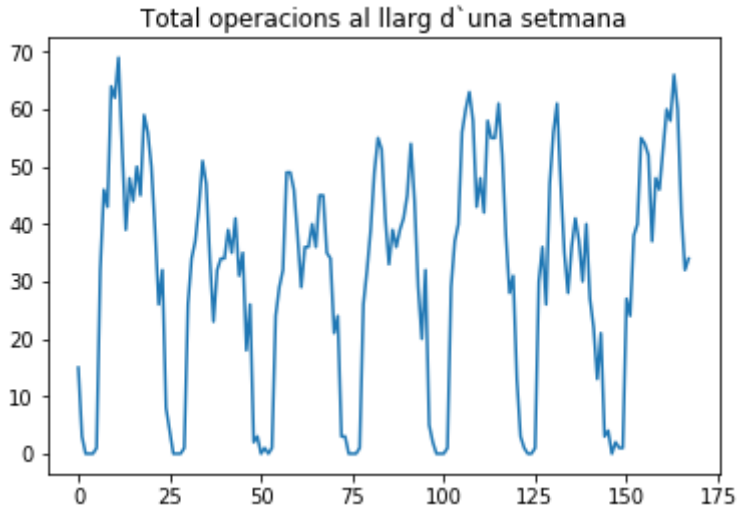






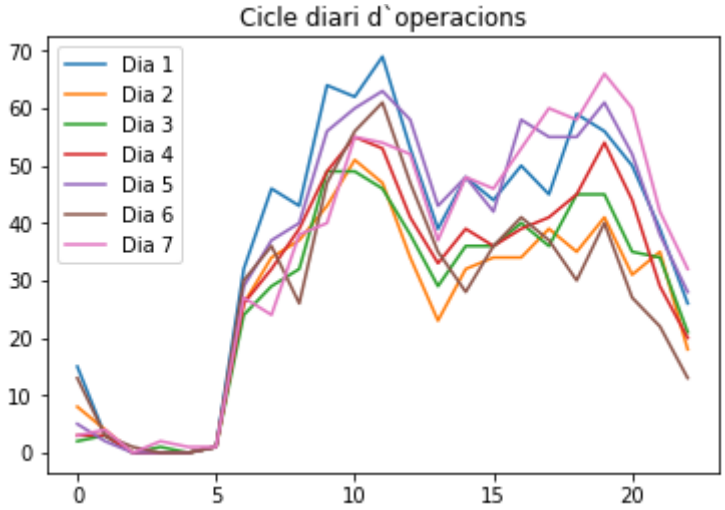
Segons la comprovació visual, i un cop eliminats els valors extrems, no podem afirmar que els valors del retard segueixin la distribució normal. Hi ha un pic a zero, però la quantitat de valors entre esquerra i dreta no s'equilibren.

A continuació es mostra la representació del total d'operacions per hora al llarg de tota la setmana.

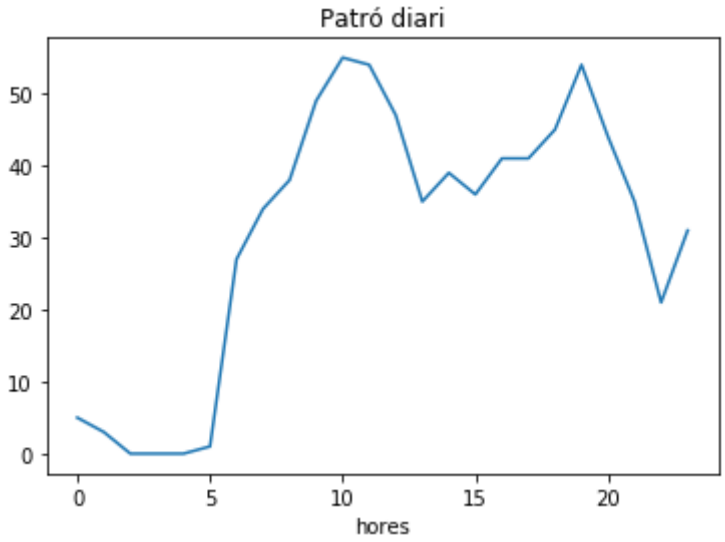


La diferenciació entre dies queda marcada per les valls profundes, moment en el qual no hi ha (gairebé mai) operacions. Són les hores entre la mitjanit i les sis del matí. Llavors acostuma a haver un primer pic abans del migdia, i a la tarda es produeix el segon pic.

A continuació es pot observar el resultat de superposar les operacions diàries de tota la setmana.



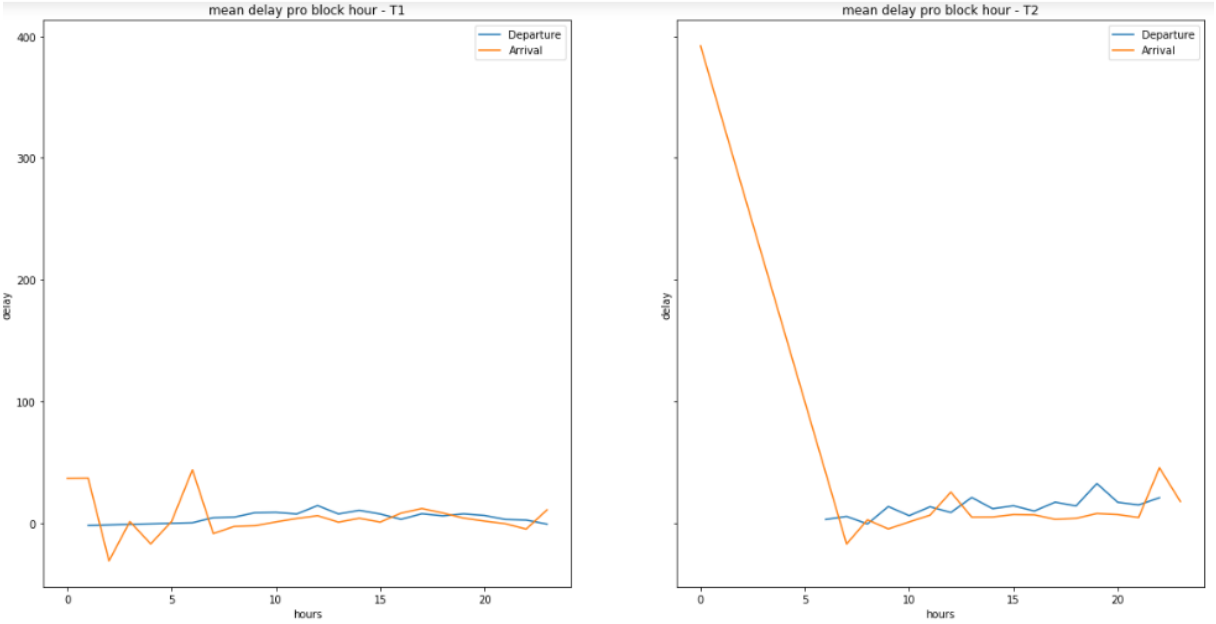
Calculant la mitjana de cada dia obtindrem el patró diari.



A continuació es mostra la superposició del número d'operacions durant tota la setmana amb el corresponent Delay mitjà experimentat durant cada bloc.

Observem visualment que no hi ha cap correlació entre número d'operacions i retard.

A continuació es superposaran els retards mitjans de sortida i d'arribada segons la terminal per analitzar visualment si hi ha alguna relació o patró de similitud.



Es determina que no s'aprecien patrons entre ambdós retards.

6. Conclusions

S'ha comprovat que les dades de retard no segueixen la distribució normal i que tenen heteroscedasticitat.

Hem aplicat el test de Dickey-Fuller i hem determinat que les dades són estacionàries, possiblement per la limitació de la nostra sèrie temporal, tanmateix, s'ha pogut identificar el cicle diari d'operacions i s'ha determinat el seu corresponent patró. Hem vist visualment que entre dies de la setmana hi ha diferències significatives.

Tot seguit hem aplicat el test de Mann-Whitney per a poder dur a terme un contrast d'hipòtesis. Hem comparat els subconjunts per terminal i per tipus d'operació i en ambdós casos hem conclòs que existeixen diferències significatives entre els subconjunts.

A continuació s'ha buscat la correlació entre el retard i el número d'operacions per bloc horari, i s'ha conclòs que gairebé no estaven gaire correlacionades, això implica que l'origen del retard és aliè a la saturació de l'aeroport i és d'origen extern. Visualment hem vist que tampoc s'aprecia cap relació entre el retard de les sortides i el de les arribades.

Finalment s'ha construït un model de regressió a partir de les dades de les operacions de Vueling el qual permet predir el retard d'una determinada ruta en funció de l'aeroport de destinació o origen així com el seu bloc horari.

7. Codi

Els datasets originals, els resultants així com el notebook de Jupyter amb el codi es troben ubicats al següent link:
https://github.com/GuillemTD/BCN_Airport_Operations

8. Recursos

Casas Roma, J. (2019). Introducción al análisis de series temporales. Editorial UOC
Calvo, M., Pérez, D., Subirats, L. (2019). Introducció a la neteja i anàlisi de dades. Editorial UOC