DFW data information

Monte Lunacek

Contents

The data available for DFW is aggregated in three frequencies: hourly, daily, and monthly. The files are formatted `parquet` files for size and speed of reading. Each section below describes the currently available data for each segment.

Please note: *Some data is missing* so consider this in your analysis.

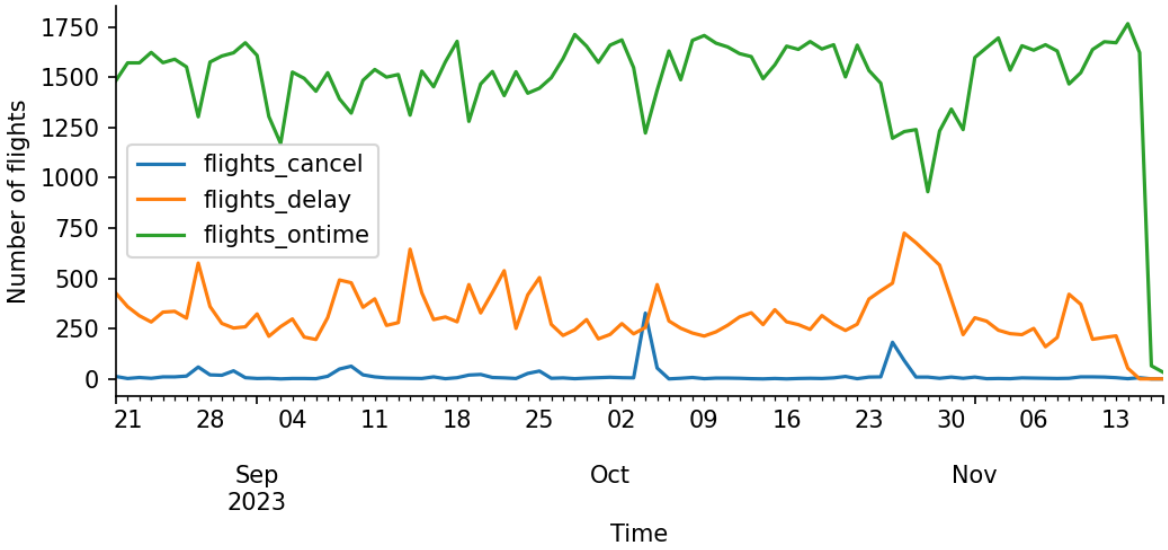
Flights

	value
filename	daily/daily_flights.parquet
start_date	2018-01-01 00:00:00
end_date	2023-11-17 00:00:00
rows	2147
columns	21

	mean	std	min	max
flights_cancel	40	108	0	1330
flights_delay	315	191	0	1270
flights_ontime	1326	276	33	1848
flights_arr	840	138	17	1106

	mean	std	min	max
flights_dep	841	138	16	1080
flights_arr_A	154	27	0	227
flights_arr_B	225	38	0	311
flights_arr_C	172	40	0	244
flights_arr_D	108	24	16	169
flights_arr_E	181	48	1	280
flights_dep_A	155	27	0	211
flights_dep_B	224	38	0	319
flights_dep_C	172	39	0	241
flights_dep_D	105	24	16	166
flights_dep_E	184	47	0	271
flights_arr_cancel	21	54	0	643
flights_arr_delay	148	96	0	651
flights_arr_ontime	672	145	17	943
flights_dep_cancel	19	55	0	687
flights_dep_delay	167	99	0	649
flights_dep_ontime	655	134	16	905

```
fig, ax = plt.subplots(figsize=(8,3), dpi=150)
df[['flights_cancel', 'flights_delay', 'flights_ontime']].tail(90).plot(ax=ax)
ax.spines[['right', 'top']].set_visible(False)
ax.set_ylabel("Number of flights")
ax.set_xlabel("Time")
plt.show()
```



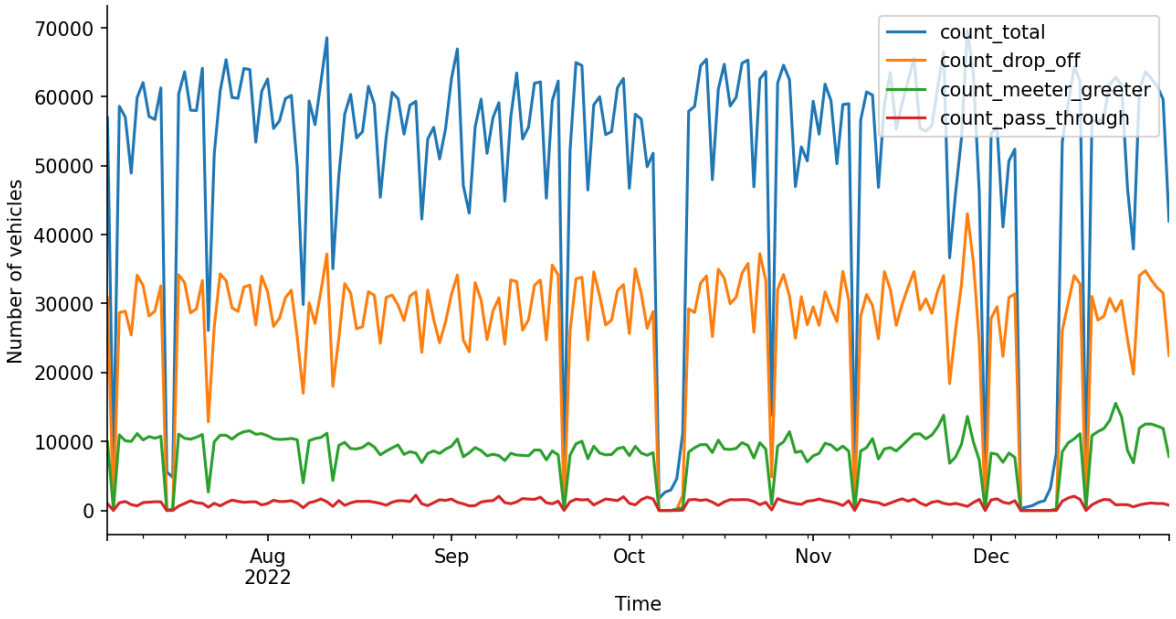
Traffic

	value
filename	daily/daily_traffic.parquet
start_date	2016-01-01 00:00:00
end_date	2022-12-31 00:00:00
rows	2557
columns	27

	mean	std	min	max
count_total	49845	15327	288	80609
count_drop_off	22814	7389	0	45058
count_meeter_greeter	8228	2856	0	15779
count_other_guest	1729	934	54	4490
count_pass_through	2369	1541	0	7265
count_terminal	7116	2627	195	12500
count_unknown_guest	7589	2477	10	13127
count_courtesy	2893	646	0	4357
count_employee	2205	863	0	4142
count_other_class	2814	1298	3	6035
count_standard	18863	5647	89	31214
count_standard_tolltag	21004	7198	184	36037

	mean	std	min	max
count_taxi	1079	780	0	4128
count_unknown_class	987	574	4	2688
duration_drop_off	414	1031	0	24207
duration_meeter_greeter	1617	2837	0	63107
duration_other_guest	78387	19441	19705	291464
duration_pass_through	303	2751	0	71886
duration_terminal	46357	16412	19780	385960
duration_unknown_guest	8641	12501	1870	267635
duration_courtesy	2095	3751	0	73920
duration_employee	6207	6414	0	175325
duration_other_class	8271	8830	1228	112793
duration_standard	42955	18807	15556	370935
duration_standard_tolltag	45321	15452	10053	275826
duration_taxi	768	350	0	7343
duration_unknown_class	21085	14737	2919	289258

```
fig, ax = plt.subplots(figsize=(10,5), dpi=150)
df[['count_total', 'count_drop_off', 'count_meeter_greeter', 'count_pass_through']].tail(1
ax.spines[['right', 'top']].set_visible(False)
ax.set_ylabel("Number of vehicles")
ax.set_xlabel("Time")
plt.show()
```



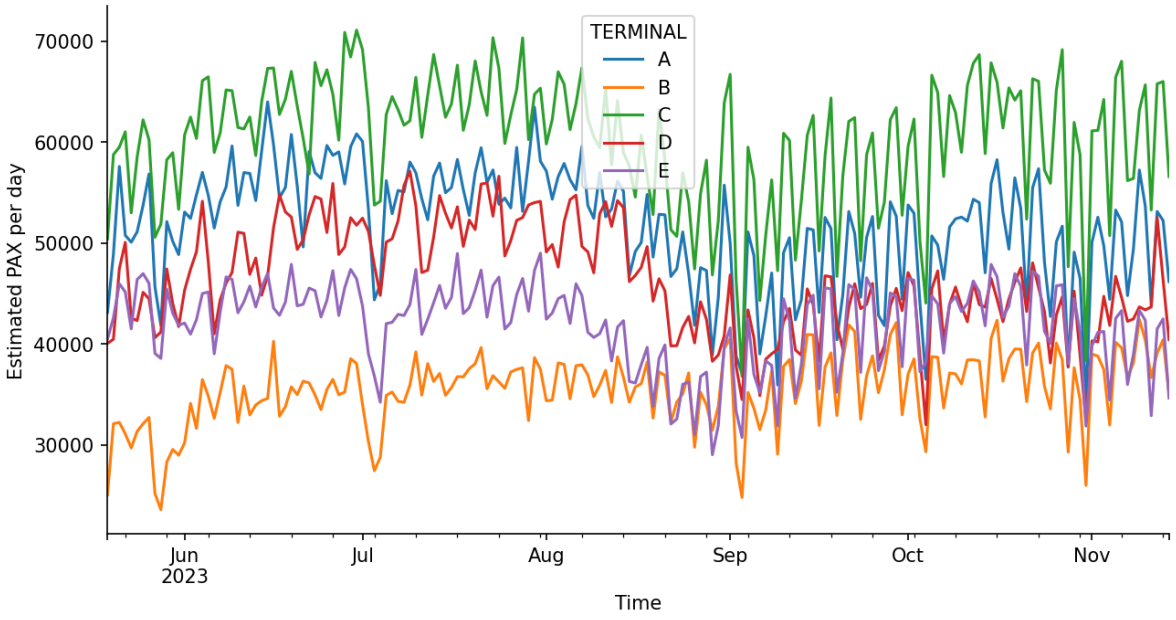
Passengers by Terminal

	value
filename	daily/daily_pax_by_terminal.parquet
start_date	2018-07-01 00:00:00
end_date	2023-11-14 00:00:00
rows	1963
columns	5

TERMINAL	mean	std	min	max
A	42049	13015	0	66074
B	27329	8172	0	44110
C	44839	15484	0	71341
D	27547	12819	0	57088
E	30843	10853	0	48980

```
fig, ax = plt.subplots(figsize=(10,5), dpi=150)
df[['A', 'B', 'C', 'D', 'E']].tail(180).plot(ax=ax)
ax.spines[['right', 'top']].set_visible(False)
```

```
ax.set_ylabel("Estimated PAX per day")
ax.set_xlabel("Time")
plt.show()
```



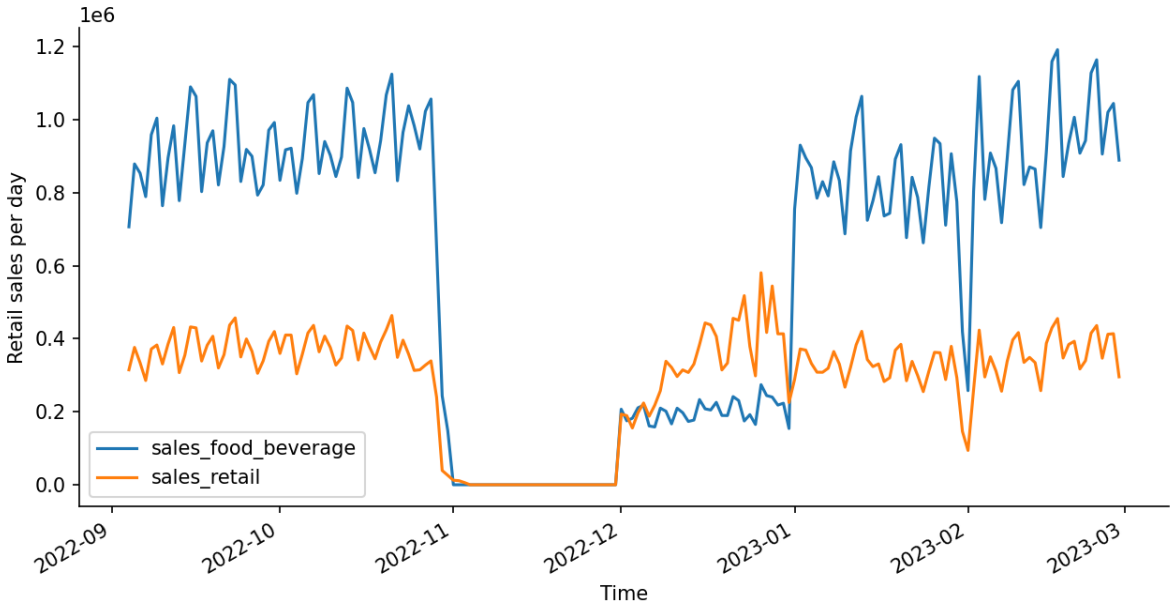
Sales

	value
filename	daily/daily_sales.parquet
start_date	2010-01-01 00:00:00
end_date	2023-03-01 00:00:00
rows	4809
columns	2

	mean	std	min	max
sales_food_beverage	574962	176417	0	1213674
sales_retail	282325	122214	-936950	1604131

```
fig, ax = plt.subplots(figsize=(10,5), dpi=150)
df[['sales_food_beverage', 'sales_retail']].tail(180).plot(ax=ax)
```

```
ax.spines[['right', 'top']].set_visible(False)
ax.set_ylabel("Retail sales per day")
ax.set_xlabel("Time")
plt.show()
```



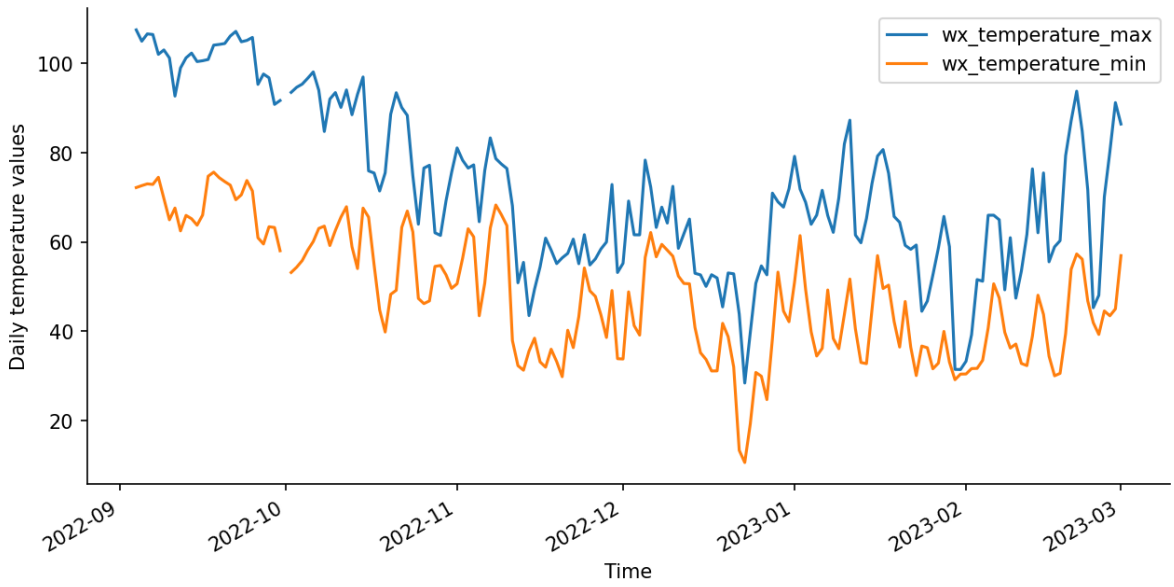
High Resolution Rapid Refresh (HRRR) weather data

	value
filename	daily/daily_weather.parquet
start_date	2010-01-01 00:00:00
end_date	2023-03-01 00:00:00
rows	4809
columns	13

	mean	std	min	max
wx_temperature_max	86	21	21	129
wx_temperature_min	57	17	-6	86
wx_apcp	0	1	0	6
wx_prate	0	0	0	3
wx_asnow	0	0	0	6

	mean	std	min	max
wx_frozr	0	0	0	1
wx_vis	10	6	0	41
wx_gust	26	9	7	61
wx_maxref	10	17	0	60
wx_cape	961	1157	0	5580
wx_lftx	1	9	-14	35
wx_wind_speed	8	3	3	22
wx_wind_direction	174	62	9	344

```
fig, ax = plt.subplots(figsize=(10,5), dpi=150)
df[['wx_temperature_max', 'wx_temperature_min']].tail(180).plot(ax=ax)
ax.spines[['right', 'top']].set_visible(False)
ax.set_ylabel("Daily temperature values")
ax.set_xlabel("Time")
plt.show()
```



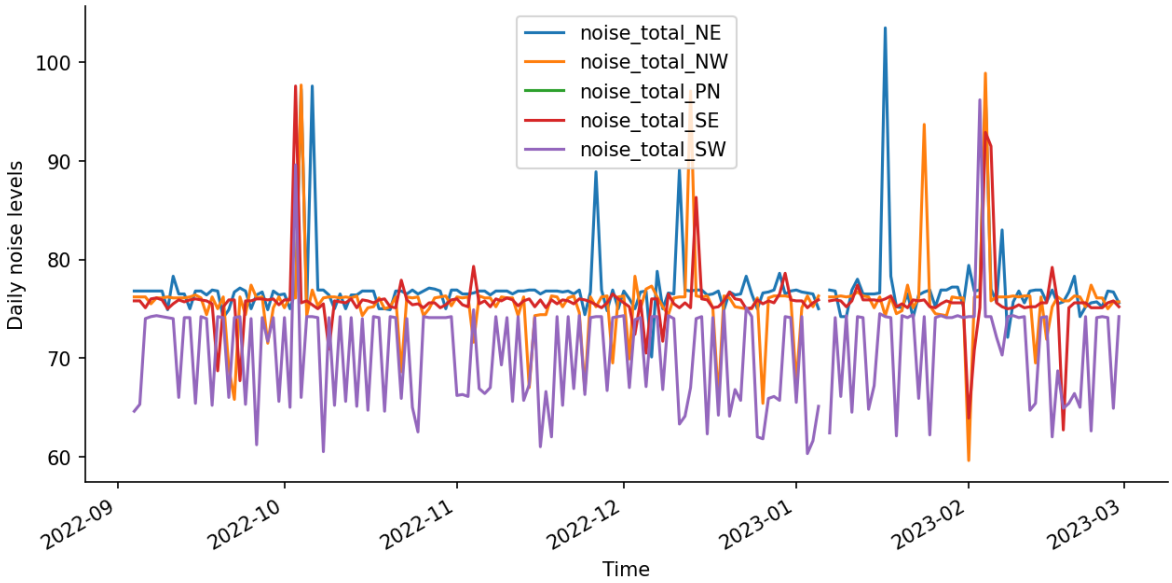
Noise

	value
filename	daily/daily_noise.parquet

	value
start_date	2010-01-01 00:00:00
end_date	2023-03-01 00:00:00
rows	4809
columns	12

	mean	std	min	max
noise_air_NE	64	15	0	97
noise_air_NW	63	15	0	101
noise_air_PN	63	12	0	68
noise_air_SE	69	5	0	100
noise_air_SW	64	12	0	130
noise_air	71	4	0	130
noise_total	76	7	0	132
noise_total_NE	69	17	0	117
noise_total_NW	68	17	0	111
noise_total_PN	67	3	58	69
noise_total_SE	72	5	0	109
noise_total_SW	70	9	0	132

```
fig, ax = plt.subplots(figsize=(10,5), dpi=150)
df[['noise_total_NE', 'noise_total_NW', 'noise_total_PN', 'noise_total_SE', 'noise_total_S
ax.spines[['right', 'top']].set_visible(False)
ax.set_ylabel("Daily noise levels")
ax.set_xlabel("Time")
plt.show()
```


noise_total_NE
noise_total_NW
noise_total_PN
noise_total_SE
noise_total_SW
Daily noise levels
100
90
80
70
60
2022-09
2022-10
2022-11
2022-12
2023-01
2023-02
2023-03
Time