Day 1: Introduction

We talked today about the massive transformation that electricity markets are witnessing, with the rapid growth of renewable power and explicity goal of fully decarbonizing the electricity market in coming years.

In this practice session, we will examine time series data from the Spanish electricity market, which has substantial intermittent renewable power (wind and solar).

The data have been collected from publicly available sources (Red Eléctrica de España and OMIE, among others). The data are from the paper "Measuring the Impact of Wind Power in the Spanish Electricity Market," by Claire Petersen, Mar Reguant, and Lola Segura.

We need to load packages in Python.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import statsmodels.formula.api as smf
from statsmodels.iolib.summary2 import summary_col
from linearmodels.iv import IV2SLS
from scipy.stats import binned_statistic
```

50] ✓ 0.0s Python

We load the data using the CSV syntax (`CSV.read`) into a data frame called `df`. `first(df,5)` gives us a snapshot of the data. Make sure the data is in the same directory as the notebook or specify the full path name.

```
df = pd.read_csv("data_spain.csv")   # adjust path if needed
df.head()
```

✓ 0.0s Python

	year	month	day	hour	dayofweek	demand	demand_forecast	wind	wind_forecast	wholesale_price	system_costs	emis_tCO2
0	2009	5	1	1	5	25.337	25.021000	5.7625	5.427	40.00	0.42	4479.166667
1	2009	5	1	2	5	23.478	23.044001	5.7461	5.441	40.20	0.24	3826.000000
2	2009	5	1	3	5	21.859	21.684999	5.7860	5.487	40.00	0.84	3359.666667
3	2009	5	1	4	5	20.931	20.408001	5.9837	5.532	36.39	2.27	3199.833333
4	2009	5	1	5	5	20.371	19.586000	6.0831	5.551	33.32	4.35	3418.333333

Generate + Code + Markdown

Summary Statistics

We start by displaying some statistics and plot hourly and yearly patterns of wind production and electricity demand.

Variables in Julia are defined by colons (:). eltype determines the type of the variable. Note that Julia differentiates between Integers (1) and Floating-Point (1.0)

```
df.describe(include="all")
```

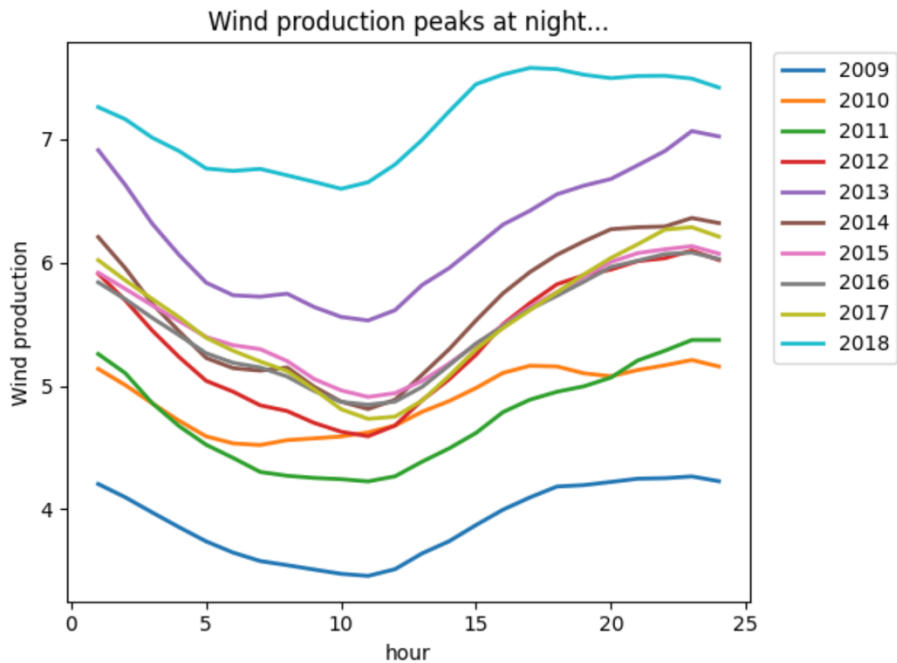
✓ 0.0s Python

	year	month	day	hour	dayofweek	demand	demand_forecast	wind	wind_forecast	wholesale_price	system_costs	emis_tCO2
count	78731.000000	78731.000000	78731.000000	78731.000000	78731.000000	78707.000000	78731.000000	78731.000000	78730.000000	78731.000000	78731.000000	78730.000000
mean	2013.388576	6.543877	15.710914	12.486492	3.003340	28.655597	28.689824	5.383975	5.293887	44.832231	3.968258	7087.110883
std	2.650560	3.412749	8.818906	6.921099	2.000191	4.844160	4.888545	3.090204	2.958359	15.493668	3.173747	2771.432907
min	2009.000000	1.000000	1.000000	1.000000	0.000000	17.096000	14.380000	0.155200	0.398000	0.000000	-1.810000	0.000000
25%	2011.000000	4.000000	8.000000	6.000000	1.000000	24.505000	24.511000	2.958600	2.969000	36.880000	1.960000	4823.500000
50%	2013.000000	7.000000	16.000000	12.000000	3.000000	28.827000	28.853001	4.789900	4.698000	46.500000	3.220000	7204.516221
75%	2016.000000	10.000000	23.000000	18.000000	5.000000	32.341500	32.421001	7.250400	7.064000	54.300000	5.070000	9222.637960
max	2018.000000	12.000000	31.000000	24.000000	6.000000	43.588000	43.568001	17.496300	16.768000	145.000000	99.389999	15772.666667

In order to plot hourly and yearly patterns, we first need to combine the data at those levels. For that, we first define the groups for which the functions will be applied using `groupby`. `agg` is then used to compute the specified summary statistic. Finally, we rename the variable as `wind_mean`.

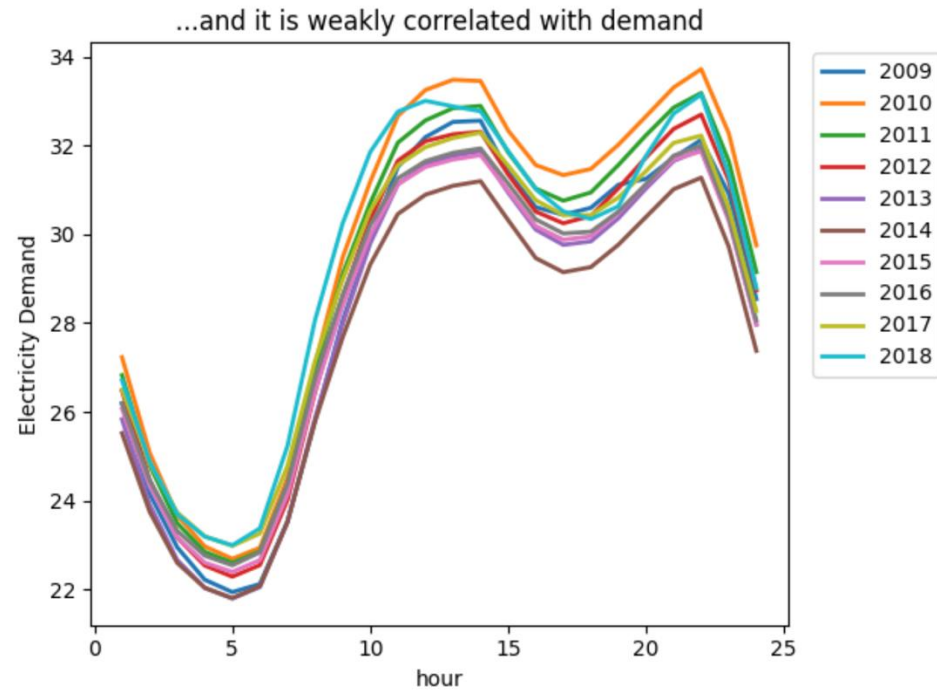
```python
# --- Hourly patterns by year (combine + plot with group=year) ---
df_mean = (
    df.groupby(["hour", "year"], as_index=False)
      .agg(wind_mean=("wind", "mean"),
           demand_mean=("demand", "mean"))
)

# Wind by hour, separate line for each year
plt.figure()
for yr, g in df_mean.groupby("year"):
    plt.plot(g["hour"], g["wind_mean"], linewidth=2, label=str(yr))
plt.title("Wind production peaks at night...")
plt.xlabel("hour")
plt.ylabel("Wind production")
plt.legend(bbox_to_anchor=(1.02, 1), loc="upper left")
plt.tight_layout()
plt.show()
```

✓ 0.0s




```python
# Demand by hour, separate line for each year
plt.figure()
for yr, g in df_mean.groupby("year"):
    plt.plot(g["hour"], g["demand_mean"], linewidth=2, label=str(yr))
plt.title("...and it is weakly correlated with demand")
plt.xlabel("hour")
plt.ylabel("Electricity Demand")
plt.legend(bbox_to_anchor=(1.02, 1), loc="upper left")
plt.tight_layout()
plt.show()
```

✓ 0.0s



The impacts of wind: a visual exploration

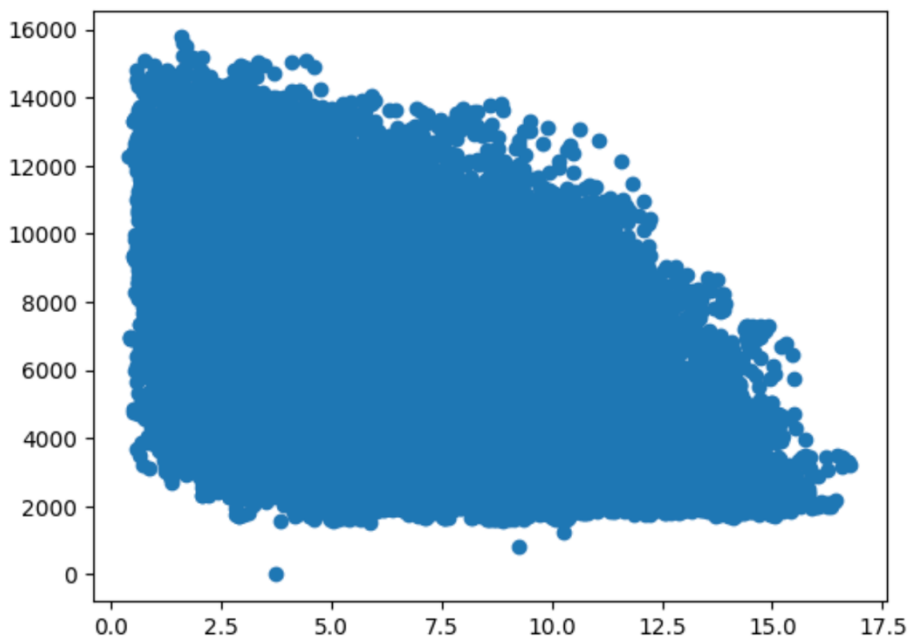
We will be plotting the **impacts of wind** on several outcomes of interest:

- Emissions
- Wholesale prices
- System costs
- Wholesale prices + system costs

We will be using bin scatters for plotting.

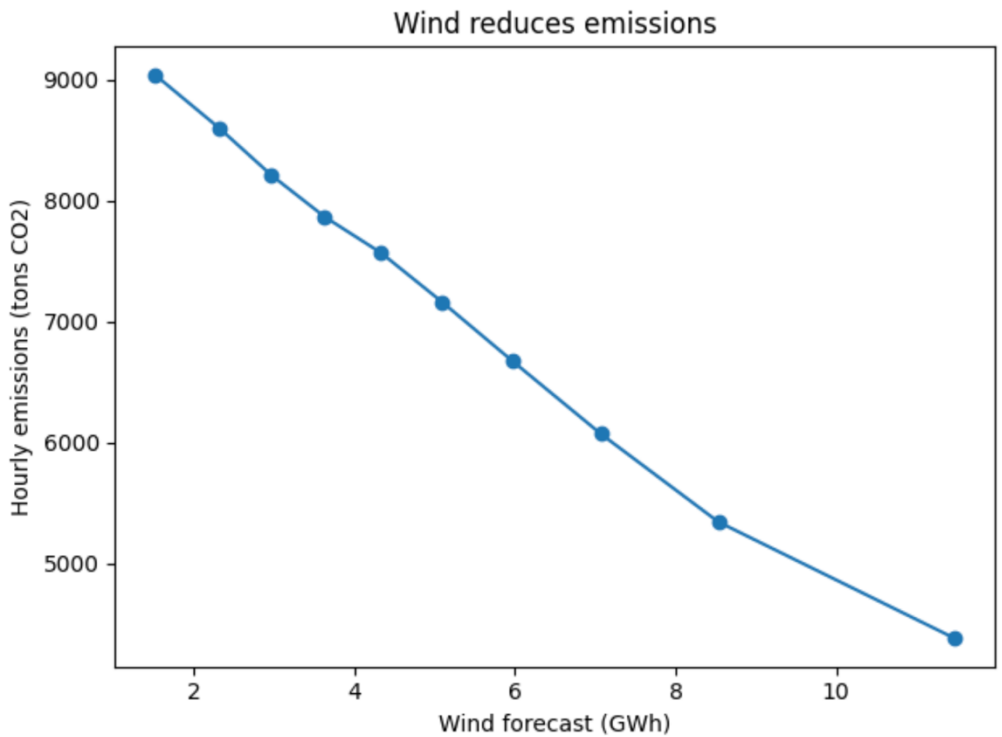
```
# This is impossible to parse!
plt.scatter(df.wind_forecast, df.emis_tCO2)
plt.show()
```

✓ 0.0s Python



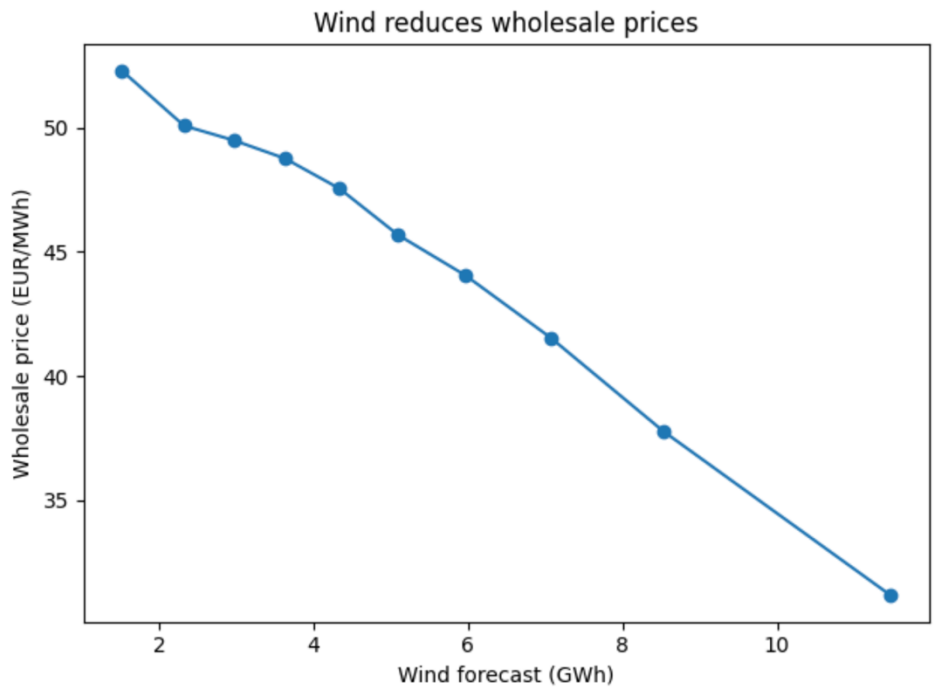
```python
# Example: wind_forecast vs emissions
fig, ax = plt.subplots()
out = binscatter_scipy(df, x="wind_forecast", y="emis_tCO2", bins=10, ax=ax, connect=True)
ax.set_title("Wind reduces emissions")
ax.set_xlabel("Wind forecast (GWh)")
ax.set_ylabel("Hourly emissions (tons CO2)")
plt.tight_layout()
plt.show()
```

7] ✓ 0.0s Python



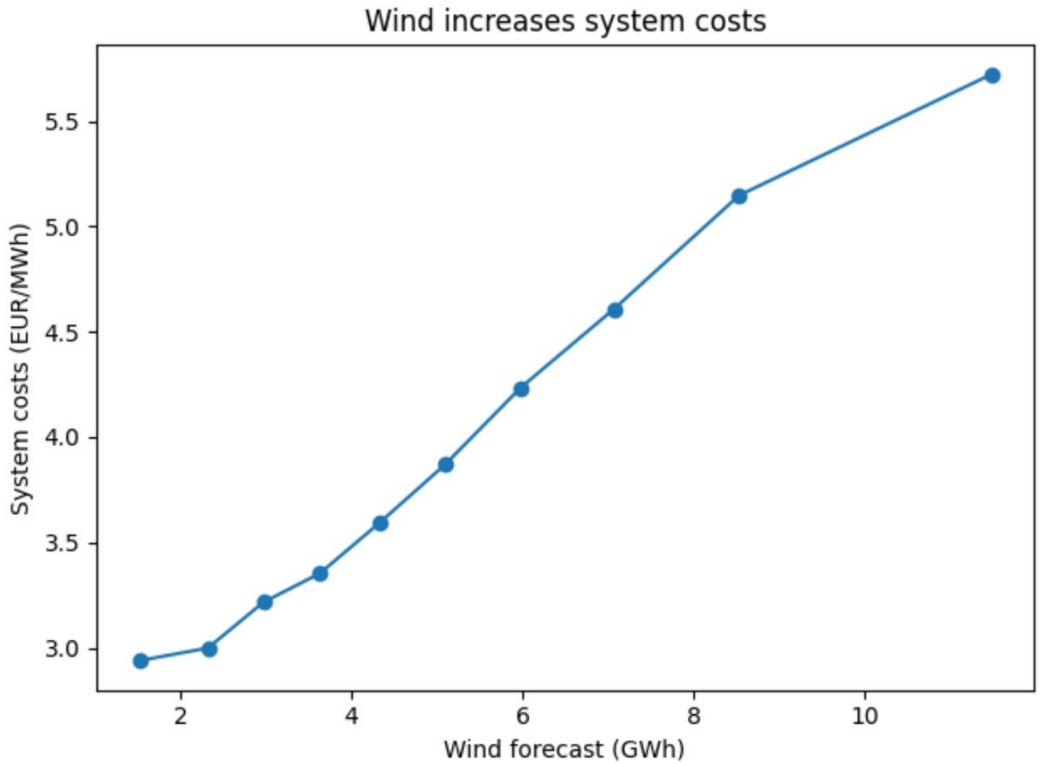
```python
# Wind reduces wholesale prices
fig, ax = plt.subplots()
binscatter_scipy(df, x="wind_forecast", y="wholesale_price", bins=10, ax=ax, connect=True)
ax.set_title("Wind reduces wholesale prices")
ax.set_xlabel("Wind forecast (GWh)")
ax.set_ylabel("Wholesale price (EUR/MWh)")
plt.tight_layout()
plt.show()
```

✓ 0.0s Python



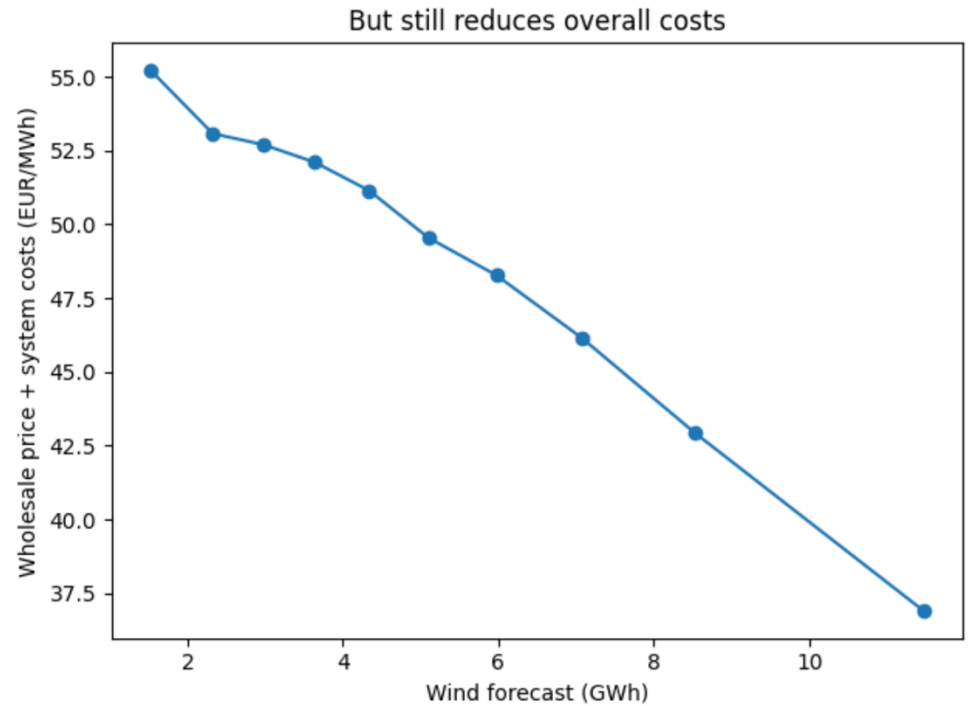
```python
# Wind increases system costs
fig, ax = plt.subplots()
binscatter_scipy(df, x="wind_forecast", y="system_costs", bins=10, ax=ax, connect=True)
ax.set_title("Wind increases system costs")
ax.set_xlabel("Wind forecast (GWh)")
ax.set_ylabel("System costs (EUR/MWh)")
plt.tight_layout()
plt.show()
```

✓ 0.0s




```python
# But still reduces overall costs (define total_price first)
df["total_price"] = df["wholesale_price"] + df["system_costs"]

fig, ax = plt.subplots()
binscatter_scipy(df, x="wind_forecast", y="total_price", bins=10, ax=ax, connect=True)
ax.set_title("But still reduces overall costs")
ax.set_xlabel("Wind forecast (GWh)")
ax.set_ylabel("Wholesale price + system costs (EUR/MWh)")
plt.tight_layout()
plt.show()
```

✓ 0.0s



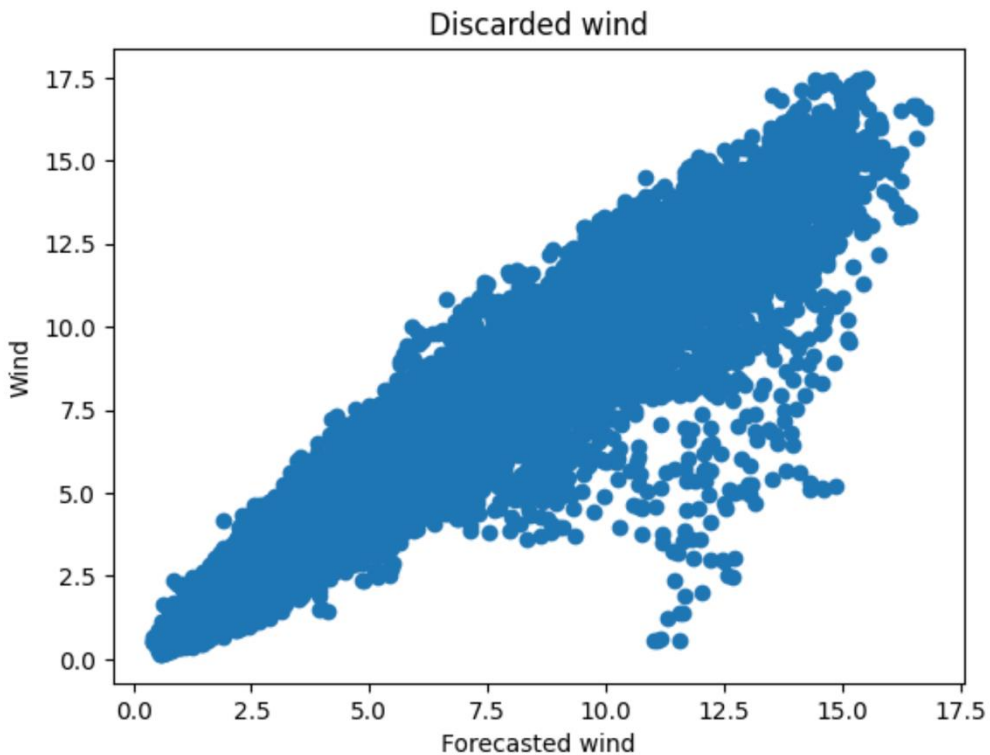
Wind endogeneity

One can estimate the effects of wind using a regression framework. However, it is important to keep in mind that wind production can be endogenous.

In moments of very high forecasted wind, it is often the case that wind is discarded. This can create an endogeneity problem.

```
plt.scatter(df.wind_forecast, df.wind)
plt.xlabel("Forecasted wind")
plt.ylabel("Wind")
plt.title("Discarded wind")
plt.show()
```

✓ 0.0s



We can examine the endogeneity problem in the context of assessing the impact of wind on reliability and other congestion costs ("system costs").

On days of very high wind, measured wind production could be lower than expected, leading to a downward bias in our estimates: a difficult day with lots of wind appears as a day with low levels of wind i the data.

To address this issue, one can use forecasted wind as an exogenous variable.

We will be running these regressions using the stats models library.

```
reg_w  = smf.ols("system_costs ~ wind + C(year) + C(month)", data=df).fit()
print(reg_w.summary())
```

✓ 0.2s

C(year)[T.2010]	0.6876	0.048	14.315	0.000	0.593	0.782
C(year)[T.2011]	0.3594	0.048	7.536	0.000	0.266	0.453
C(year)[T.2012]	1.8192	0.048	38.080	0.000	1.726	1.913
C(year)[T.2013]	2.5999	0.048	53.967	0.000	2.505	2.694
C(year)[T.2014]	2.8978	0.048	60.308	0.000	2.804	2.992
C(year)[T.2015]	1.6140	0.048	33.760	0.000	1.520	1.708
C(year)[T.2016]	0.3445	0.048	7.213	0.000	0.251	0.438
C(year)[T.2017]	-0.4292	0.048	-8.979	0.000	-0.523	-0.336
C(year)[T.2018]	-0.4703	0.062	-7.540	0.000	-0.593	-0.348
C(month)[T.2]	-0.3599	0.051	-7.101	0.000	-0.459	-0.261
C(month)[T.3]	0.6367	0.049	12.878	0.000	0.540	0.734
C(month)[T.4]	0.7471	0.050	14.954	0.000	0.649	0.845
C(month)[T.5]	-0.3796	0.049	-7.788	0.000	-0.475	-0.284
C(month)[T.6]	-1.1558	0.051	-22.712	0.000	-1.256	-1.056
C(month)[T.7]	-1.1463	0.051	-22.605	0.000	-1.246	-1.047
C(month)[T.8]	-0.7329	0.051	-14.416	0.000	-0.833	-0.633
C(month)[T.9]	-0.6473	0.051	-12.628	0.000	-0.748	-0.547
C(month)[T.10]	0.6735	0.050	13.345	0.000	0.575	0.772
C(month)[T.11]	0.3447	0.050	6.850	0.000	0.246	0.443
C(month)[T.12]	0.1088	0.050	2.168	0.030	0.010	0.207
wind	0.1872	0.004	52.561	0.000	0.180	0.194

Omnibus:	83989.327	Durbin-Watson:	0.593
Prob(Omnibus):	0.000	Jarque-Bera (JB):	19895973.331
Skew:	4.973	Prob(JB):	0.00
Kurtosis:	80.240	Cond. No.	87.2

```
reg_wf = smf.ols("system_costs ~ wind_forecast + C(year) + C(month)", data=df).fit()
print(reg_wf.summary())
```

✓ 0.2s

C(year)[T.2010]	0.7202	0.048	15.071	0.000	0.627	0.814
C(year)[T.2011]	0.3924	0.047	8.269	0.000	0.299	0.485
C(year)[T.2012]	1.8063	0.048	38.000	0.000	1.713	1.899
C(year)[T.2013]	2.4847	0.048	51.640	0.000	2.390	2.579
C(year)[T.2014]	2.8245	0.048	58.981	0.000	2.731	2.918
C(year)[T.2015]	1.5625	0.048	32.817	0.000	1.469	1.656
C(year)[T.2016]	0.2735	0.048	5.747	0.000	0.180	0.367
C(year)[T.2017]	-0.4478	0.048	-9.414	0.000	-0.541	-0.355
C(year)[T.2018]	-0.5157	0.062	-8.307	0.000	-0.637	-0.394
C(month)[T.2]	-0.4017	0.050	-7.961	0.000	-0.501	-0.303
C(month)[T.3]	0.6039	0.049	12.274	0.000	0.508	0.700
C(month)[T.4]	0.7751	0.050	15.589	0.000	0.678	0.873
C(month)[T.5]	-0.3046	0.049	-6.270	0.000	-0.400	-0.209
C(month)[T.6]	-1.0673	0.051	-21.036	0.000	-1.167	-0.968
C(month)[T.7]	-1.0743	0.050	-21.274	0.000	-1.173	-0.975
C(month)[T.8]	-0.6595	0.051	-13.027	0.000	-0.759	-0.560
C(month)[T.9]	-0.5486	0.051	-10.731	0.000	-0.649	-0.448
C(month)[T.10]	0.7830	0.050	15.541	0.000	0.684	0.882
C(month)[T.11]	0.3925	0.050	7.835	0.000	0.294	0.491
C(month)[T.12]	0.1581	0.050	3.164	0.002	0.060	0.256
wind_forecast	0.2250	0.004	59.685	0.000	0.218	0.232

Omnibus:	84555.535	Durbin-Watson:	0.597
Prob(Omnibus):	0.000	Jarque-Bera (JB):	20749118.420
Skew:	5.022	Prob(JB):	0.00
Kurtosis:	81.894	Cond. No.	85.4

We might want to instrument wind production with its forecast instead using linearmodels.

```python
d = df[["system_costs","wind","wind_forecast","year","month"]].dropna()

# 1st stage: wind on instrument + FE
fs = smf.ols("wind ~ wind_forecast + C(year) + C(month)", data=d).fit()
d = d.assign(wind_hat=fs.fittedvalues)

# 2nd stage: outcome on predicted wind + FE
ss = smf.ols("system_costs ~ wind_hat + C(year) + C(month)", data=d).fit()

print(ss.summary())


# Alternative: use linearmodels IV2SLS
# iv = IV2SLS.from_formula("system_costs ~ 1 + C(year) + C(month) + [wind ~ wind_forecast]", data=d).fit(cov_type="robust")   # or "unadjusted", "robust", "clustered"
# print(iv.summary)
```

✓ 0.7s Python

```
C(year)[T.2010]     0.6732      0.048     14.081      0.000       0.579       0.767
C(year)[T.2011]     0.3513      0.047      7.402      0.000       0.258       0.444
C(year)[T.2012]     1.7918      0.048     37.683      0.000       1.699       1.885
C(year)[T.2013]     2.5448      0.048     53.048      0.000       2.451       2.639
C(year)[T.2014]     2.8613      0.048     59.822      0.000       2.768       2.955
C(year)[T.2015]     1.5811      0.048     33.226      0.000       1.488       1.674
C(year)[T.2016]     0.3139      0.048      6.604      0.000       0.221       0.407
C(year)[T.2017]    -0.4616      0.048     -9.701      0.000      -0.555      -0.368
C(year)[T.2018]    -0.5261      0.062     -8.471      0.000      -0.648      -0.404
C(month)[T.2]      -0.3872      0.050     -7.676      0.000      -0.486      -0.288
C(month)[T.3]       0.6309      0.049     12.824      0.000       0.534       0.727
C(month)[T.4]       0.7778      0.050     15.643      0.000       0.680       0.875
C(month)[T.5]      -0.3290      0.049     -6.779      0.000      -0.424      -0.234
C(month)[T.6]      -1.0852      0.051    -21.410      0.000      -1.185      -0.986
C(month)[T.7]      -1.0685      0.051    -21.153      0.000      -1.168      -0.970
C(month)[T.8]      -0.6517      0.051    -12.867      0.000      -0.751      -0.552
C(month)[T.9]      -0.5621      0.051    -11.006      0.000      -0.662      -0.462
C(month)[T.10]      0.7360      0.050     14.643      0.000       0.637       0.834
C(month)[T.11]      0.3536      0.050      7.061      0.000       0.255       0.452
C(month)[T.12]      0.1263      0.050      2.530      0.011       0.028       0.224
wind_hat            0.2187      0.004     59.685      0.000       0.211       0.226
==============================================================================
```

Another possible problem is that system costs from wind production may be realized in hours with no wind. In this case, the hourly regression coefficient will be downward biased. To circumvent this issue, we can estimate the same regression at a daily level.

For that, we compute the total system costs as well as total wind power.

```
df["day_id"] = df["year"].astype(str) + df["month"].astype(str) + df["day"].astype(str)

df_day = df.groupby(["day_id", "year", "month"], as_index=False).agg(wind_forecast=("wind_forecast", "sum"), systemcosts=("system_costs", "sum"))
```

65] ✓ 0.0s Python

```
reg_d = smf.ols("systemcosts ~ wind_forecast + C(year) + C(month)", data=df_day).fit()
print(reg_d.summary())
```

72] ✓ 0.0s Python

```
C(year)[T.2012]    42.6135      3.052     13.961      0.000      36.629      48.598
C(year)[T.2013]    58.0648      3.096     18.754      0.000      51.994      64.135
C(year)[T.2014]    65.7179      3.071     21.397      0.000      59.696      71.740
C(year)[T.2015]    36.7338      3.061     12.002      0.000      30.733      42.735
C(year)[T.2016]     5.8983      3.060      1.927      0.054      -0.102      11.898
C(year)[T.2017]   -11.2469      3.057     -3.679      0.000     -17.241      -5.253
C(year)[T.2018]   -13.6806      3.976     -3.441      0.001     -21.477      -5.884
C(month)[T.2]      -9.6900      3.237     -2.994      0.003     -16.036      -3.344
C(month)[T.3]      14.7982      3.155      4.690      0.000       8.612      20.984
C(month)[T.4]      19.8577      3.188      6.229      0.000      13.607      26.108
C(month)[T.5]      -5.3168      3.118     -1.705      0.088     -11.430       0.796
C(month)[T.6]     -23.1291      3.261     -7.092      0.000     -29.523     -16.735
C(month)[T.7]     -23.1790      3.247     -7.140      0.000     -29.544     -16.814
C(month)[T.8]     -13.4584      3.253     -4.138      0.000     -19.836      -7.081
C(month)[T.9]     -10.3939      3.288     -3.161      0.002     -16.840      -3.947
C(month)[T.10]     20.8703      3.236      6.450      0.000      14.526      27.214
C(month)[T.11]     10.6521      3.214      3.314      0.001       4.350      16.954
C(month)[T.12]      4.7161      3.201      1.473      0.141      -1.560      10.992
wind_forecast       0.2556      0.011     23.194      0.000       0.234       0.277
==============================================================================
Omnibus:                      981.781   Durbin-Watson:                   1.165
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3765.696
Skew:                           1.432   Prob(JB):                         0.00
Kurtosis:                       7.386   Cond. No.                     1.98e+03
==============================================================================
```

Follow-up exercises

1. What is the correlation of wind and demand? How could that affect the valuation of wind power?
2. (*) What is the environmental benefit of wind power in this market per unit of wind? Try to quantify that by regressing emissions on wind and converting it to a monetary amount using a valuation for emissions reductions. Estimate the total welfare effects of wind production. For that, you need to add to the environmental benefit the consumer and producer surplus. With respect to the producer surplus assume that the LCOE ranges between 50 to 90 €/MWh. How does your answer depend on the monetary value of reducing emissions?