Day 7: Demand - Evidence

Code will replicate basic results for these two papers

In this lecture, we talked about measuring the response to different demand-side policies.

We will depart a bit from our model to use high-frequency demand data.

We will compare different dynamic pricing programs by replicating some of the results in

- "Estimating the Elasticity to Real Time Pricing," by Fabra, Rapson, Reguant and Wang
  - Data: Smart-meter household data
  - Policy: RTP
  - Method: IV regression
- "Measuring the Impact of Time-of-Use Pricing on Electricity Consumption: Evidence from Spain" by Enrich, Li, Mizraghi and Reguant
  - Data: Utility-level consumption data
  - Policy: Time-of-Use
  - Method: Diff-in-diff policy comparison

We load packages and set the dirpath.

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

from linearmodels.iv import IV2SLS
import pyfixest as pf
```

] ✓ 0.0s

Real-time Pricing (RTP)

Data exploration

First paper: confidential data
(this is a synthetic dataset)

Loading **data**.

- data_rtp.csv: Smart meter data of a small sample of 40 consumers. We will use **kwh** (hourly electricity consumption in mwh) as our dependent variable.

The data is already merged with several other hourly data that can be either

Consumer specific:

- temp, temp2: temperature
- rtp / tou: whether consumers are under rtp pricing (for the energy cost) and tou pricing (for the charges component, more on that in the second part)

Market specific:

- price: price of a mwh of electricity
- wind_hat: wind forecast
- solar_actual: solar production
- mwh_dayaheadiberia: demand forecast

Time variables:

- y: year
- m: month
- hr: hour

Unit of observation: hourly
data at the individual level

```python
mydata = pd.read_csv(f"{dirpath}/data_rtp.csv").dropna().copy()
mydata.head()
```

✓ 0.1s

	id	rtp	tou	date	y	m	hr	weekend	kwh	price	wind_hat	solar_actual	temp	temp2	mwh_dayaheadiberia
0	8	1	0	20563	2016	4	2	0.0	0.074	0.06358	8950	108.666660	60.0	3600.0	22749.400
1	8	1	0	20563	2016	4	3	0.0	0.059	0.06356	9179	101.500000	58.0	3364.0	21948.699
2	8	1	0	20563	2016	4	4	0.0	0.009	0.06563	8486	87.333336	53.0	2809.0	21109.400
3	8	1	0	20563	2016	4	5	0.0	0.134	0.06974	8615	87.000000	53.0	2809.0	20930.500
4	8	1	0	20563	2016	4	6	0.0	0.069	0.08423	8708	62.333332	54.0	2916.0	21265.801

```python
# Adding some variables
mydata["log_price"] = np.log(mydata["price"])
mydata["log_wind_hat"] = np.log(mydata["wind_hat"])
mydata["log_kwh"] = np.log(mydata["kwh"] + 0.01)
```

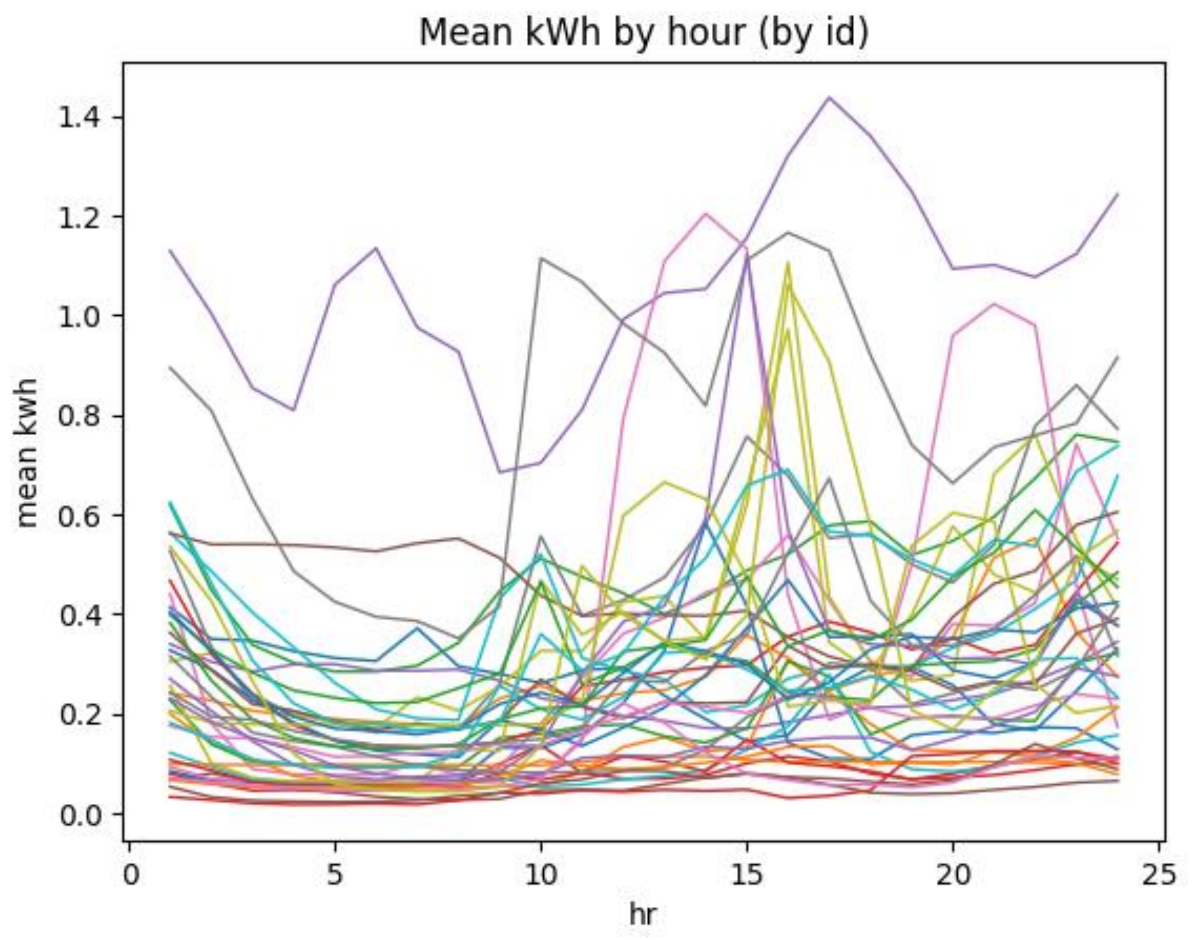
✓ 0.0s

It can be useful to plot the data to examine patterns. We can plot the typical consumption pattern of consumers during the day.

```
# plotting daily consumption patterns by id
df_plt = mydata.loc[:, ["id", "hr", "kwh"]].copy()
df_plt = (
    df_plt.groupby(["id", "hr"], as_index=False)
          .agg(kwh_mean=("kwh", "mean"))
          .sort_values(["id", "hr"])
)

plt.figure()
for i, g in df_plt.groupby("id"):
    plt.plot(g["hr"], g["kwh_mean"], linewidth=1)
plt.xlabel("hr")
plt.ylabel("mean kwh")
plt.title("Mean kWh by hour (by id)")
plt.legend([], [], frameon=False)
plt.show()
```

✓ 0.0s

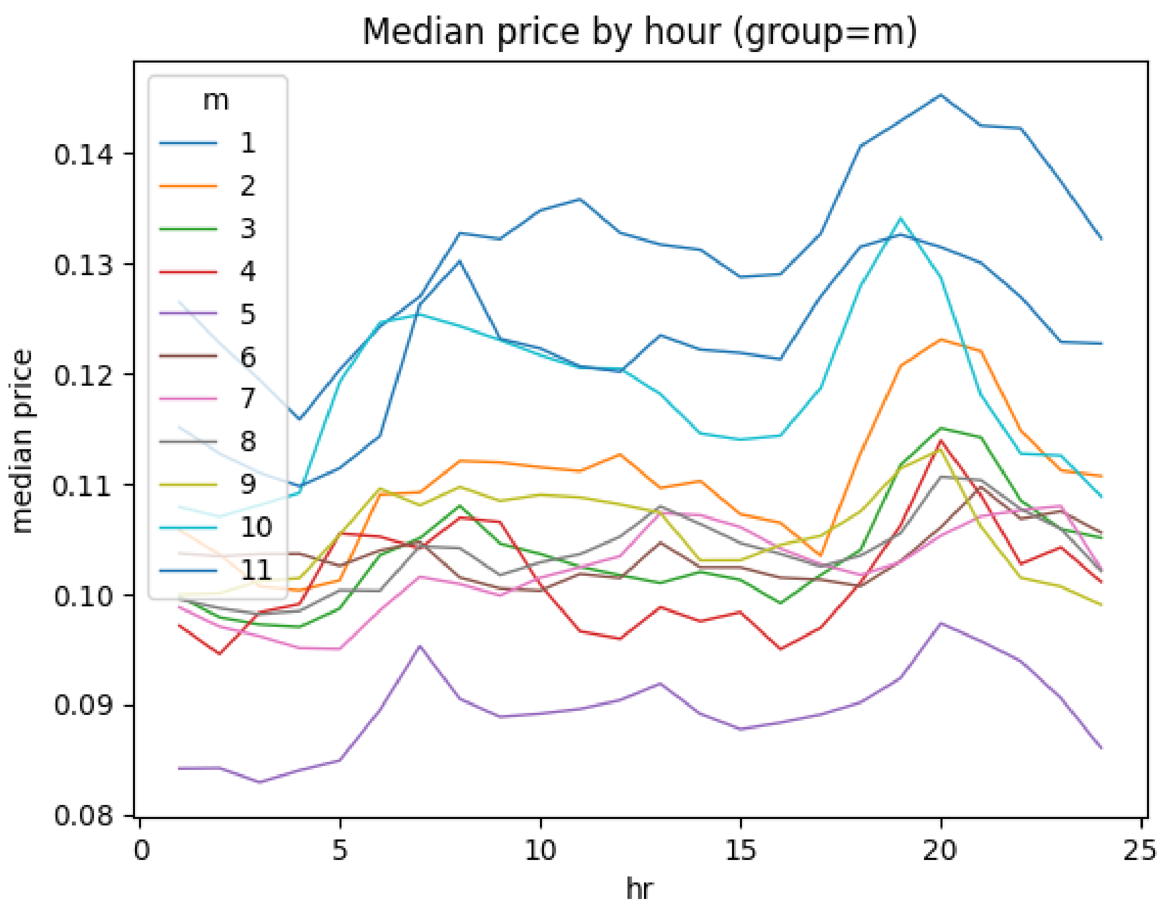


Substantial heterogeneity across households

```python
# we can also plot prices by month
df_plt = mydata.loc[:, ["hr", "price", "m"]].copy()
df_plt = (
    df_plt.groupby(["hr", "m"], as_index=False)
            .agg(price_median=("price", "median"))
            .sort_values(["m", "hr"])
)

plt.figure()
for m, g in df_plt.groupby("m"):
    plt.plot(g["hr"], g["price_median"], linewidth=1, label=str(m))
plt.xlabel("hr")
plt.ylabel("median price")
plt.title("Median price by hour (group=m)")
plt.legend(title="m")
plt.show()
```

✓ 0.0s



```python
reg1 = IV2SLS.from_formula("kwh ~ 1 + price", data=mydata).fit()
print(reg1.summary)
```
✓ 0.4s

```
                            OLS Estimation Summary
==============================================================================
Dep. Variable:                    kwh   R-squared:                      0.0003
Estimator:                        OLS   Adj. R-squared:                 0.0003
No. Observations:              380063   F-statistic:                    90.738
Date:                Sun, Jan 25 2026   P-value (F-stat)                0.0000
Time:                        21:00:21   Distribution:                  chi2(1)
Cov. Estimator:                robust

                             Parameter Estimates
==============================================================================
            Parameter  Std. Err.     T-stat    P-value    Lower CI    Upper CI
------------------------------------------------------------------------------
Intercept      0.2422     0.0038     63.159     0.0000      0.2347      0.2497
price          0.3404     0.0357     9.5256     0.0000      0.2703      0.4104
==============================================================================
```

Demand increases with higher prices?

An instrument is needed!

Estimation of elasticities

We will be running a regression for each consumer in our sample, instrumenting price with wind forecast and obtaining a distribution of elasticities.

```
iv_formula = (
    "log_kwh ~ 1 + solar_actual + temp + temp2 + mwh_dayaheadiberia"
    " + C(y) + C(hr):C(m) + C(weekend):C(hr)"
    " [log_price ~ log_wind_hat]"
)

iv_full = IV2SLS.from_formula(iv_formula, data=mydata).fit(
    cov_type="clustered",  # optional
    clusters=mydata["id"], # (not in Julia; handy default if you want)
)
print("beta_price =", iv_full.params["log_price"])
print("se_price   =", iv_full.std_errors["log_price"])
```

✓ 34.0s

Py

```
beta_price = -0.04004233653378894
se_price   = 0.15478752401595355
```

We can also compute elasticities at the individual level, to get a sense of the noise/distribution of effects.

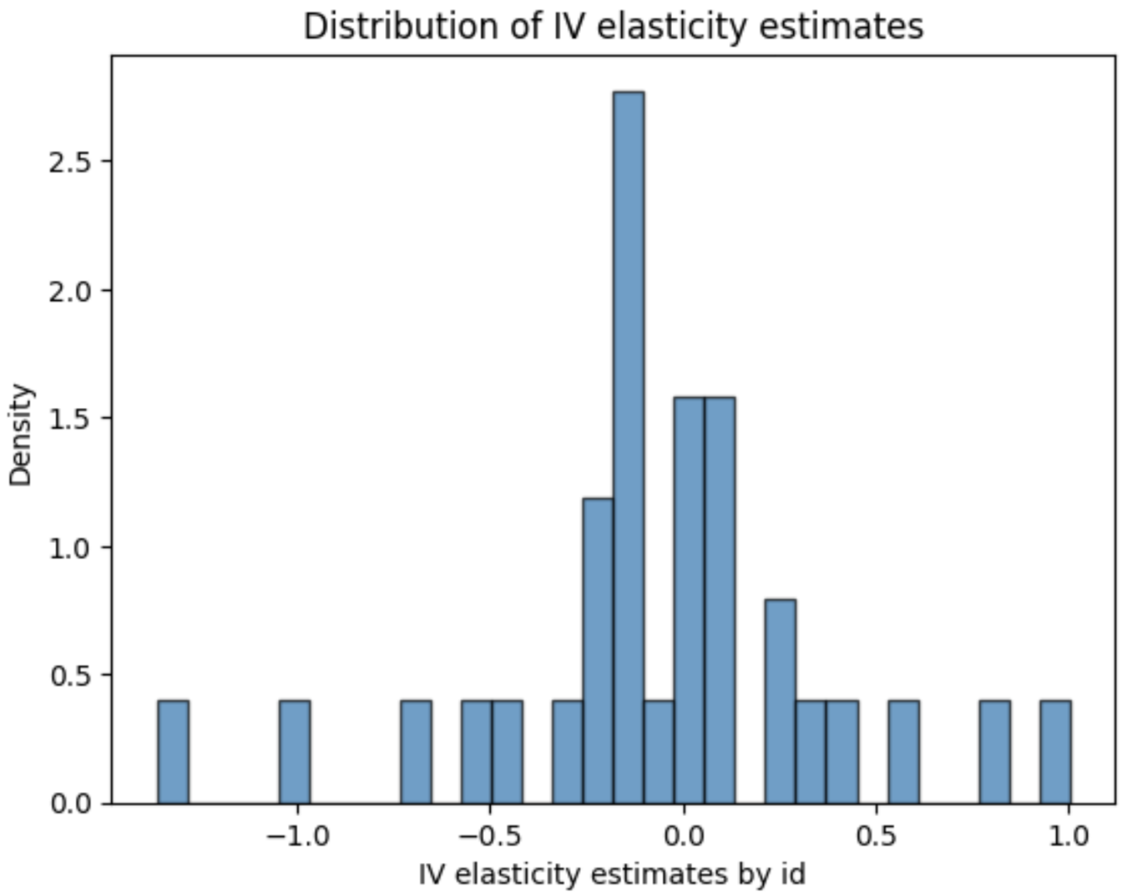
```
# -------------------------
# Per-id IV elasticities (Julia loop storing beta and rtp) :contentReference[oaicite:3]{index=3}
# -------------------------
betas = []
for i, g in mydata.groupby("id"):
    # optional guard (avoid tiny groups / collinearity from FE)
    if len(g) < 30:
        continue
    try:
        res_i = IV2SLS.from_formula(iv_formula, data=g).fit()
        betas.append({"id": i, "beta": res_i.params["log_price"], "rtp": g["rtp"].mean()})
    except Exception:
        pass

betas = pd.DataFrame(betas)
print(betas.head())
```

✓ 38.7s

```python
# Density plot of betas
plt.figure()
plt.hist(betas["beta"], bins=30, density=True, alpha=0.7, edgecolor='black')
plt.xlabel("IV elasticity estimates by id")
plt.ylabel("Density")
plt.title("Distribution of IV elasticity estimates")
plt.show()
```

✓ 0.0s

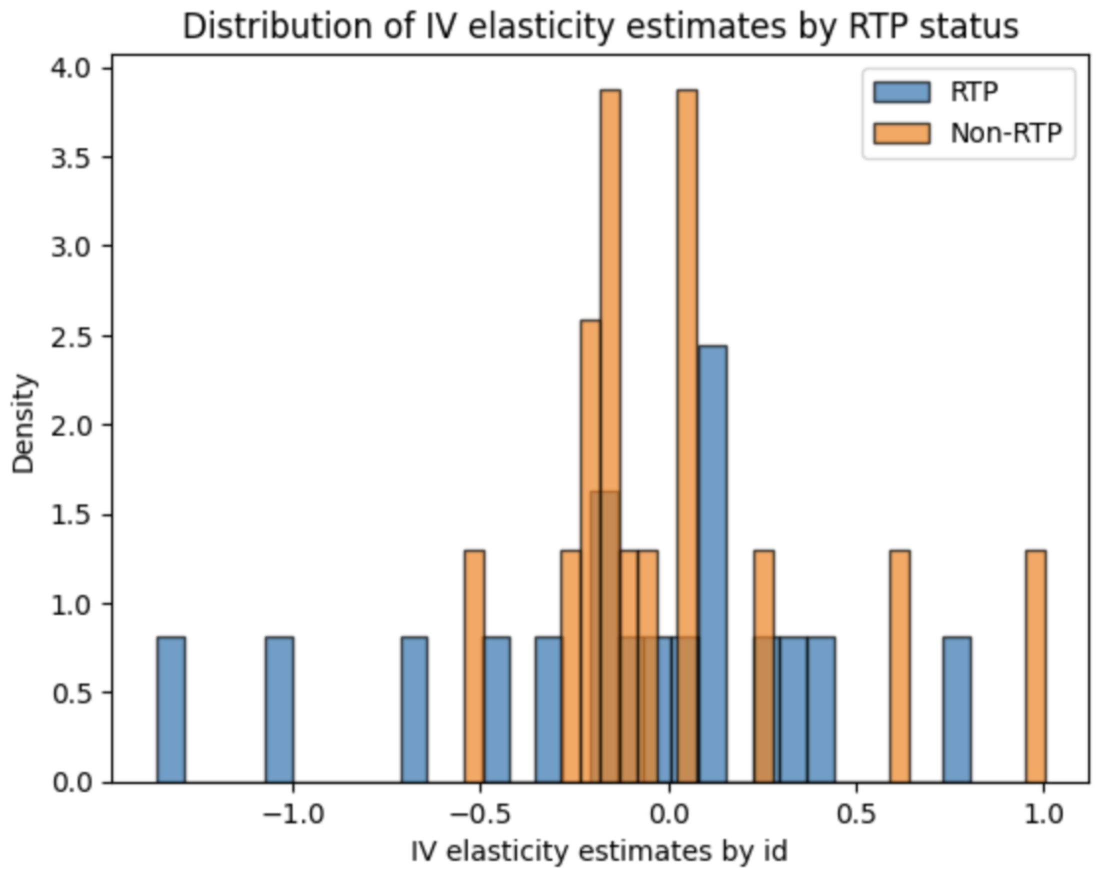


Julia env: v1.10

Effects are centered around zero but noisy, in paper implied elasticity is zero

```python
# K-density by rtp/non-rtp
plt.figure()
betas_rtp = betas[betas["rtp"] == 1]["beta"]
betas_nonrtp = betas[betas["rtp"] == 0]["beta"]
plt.hist(betas_rtp, bins=30, density=True, alpha=0.7, label="RTP", edgecolor='black')
plt.hist(betas_nonrtp, bins=30, density=True, alpha=0.7, label="Non-RTP", edgecolor='black')
plt.xlabel("IV elasticity estimates by id")
plt.ylabel("Density")
plt.title("Distribution of IV elasticity estimates by RTP status")
plt.legend()
plt.show()
```

✓ 0.1s



Noisy across RTP/non-RTP, but no statistical difference.

2. Time-of-Use (TOU)

This is a public dataset based on a combination of sources (REE, OMIE)

Data

Loading data.

- df_tou.csv: time series with hourly data at distribution level

```
df_tou = pd.read_csv(f"{dirpath}/df_tou.csv").copy()
df_tou["tou"] = df_tou["tou"].astype(str)
df_tou["tou_allweek"] = df_tou["tou_allweek"].astype(str)
```

✓ 0.1s

```
df_tou.head()
```

✓ 0.0s

	date	hour	dist	year	month	country	tou	tou_allweek	month_count	policy	...	week	week_c	temp	total_price	temph	charges
0	2018-01-01	1	EDP	2018	1	ES	1	1	1	0	...	True	week	9.0734	76.13	0.0	44.03
1	2018-01-01	2	EDP	2018	1	ES	1	1	1	0	...	True	week	8.9438	74.24	0.0	44.03
2	2018-01-01	3	EDP	2018	1	ES	1	1	1	0	...	True	week	9.0122	73.05	0.0	44.03
3	2018-	4	EDP	2018	1	ES	1	1	1	0	...	True	week	9.2309	69.48	0.0	44.03

Description of variables

Time variables:

- date, hour, year, month
- month_count: month of sample
- week, week_c: dummy variables indicating whether the observation falls into a weekday or a weekend

Identifier:

- dist: distribution area

Unit of observation: hourly data at the distribution area level (large areas)

Policy variables:

- policy: takes 1 for all distribution areas in Spain after the introduction of the policy
- placebo: takes 1 for all distribution areas in Spain one month before the introduction of the policy
- tou: TOU tariffs, split between **Off-peak, Mid-Peak, and Peak hours**
- tou_allweek: TOU tariffs but artificially differentiating hours during the weekend (even though all hours had the same electricity price).

Controls:

- temperature: temp, temph (whether the temperature is above 20ºC)

Prices:

- charges: charges component of the electricity price, affected by TOU tariffs
- total_price: charges + energy cost

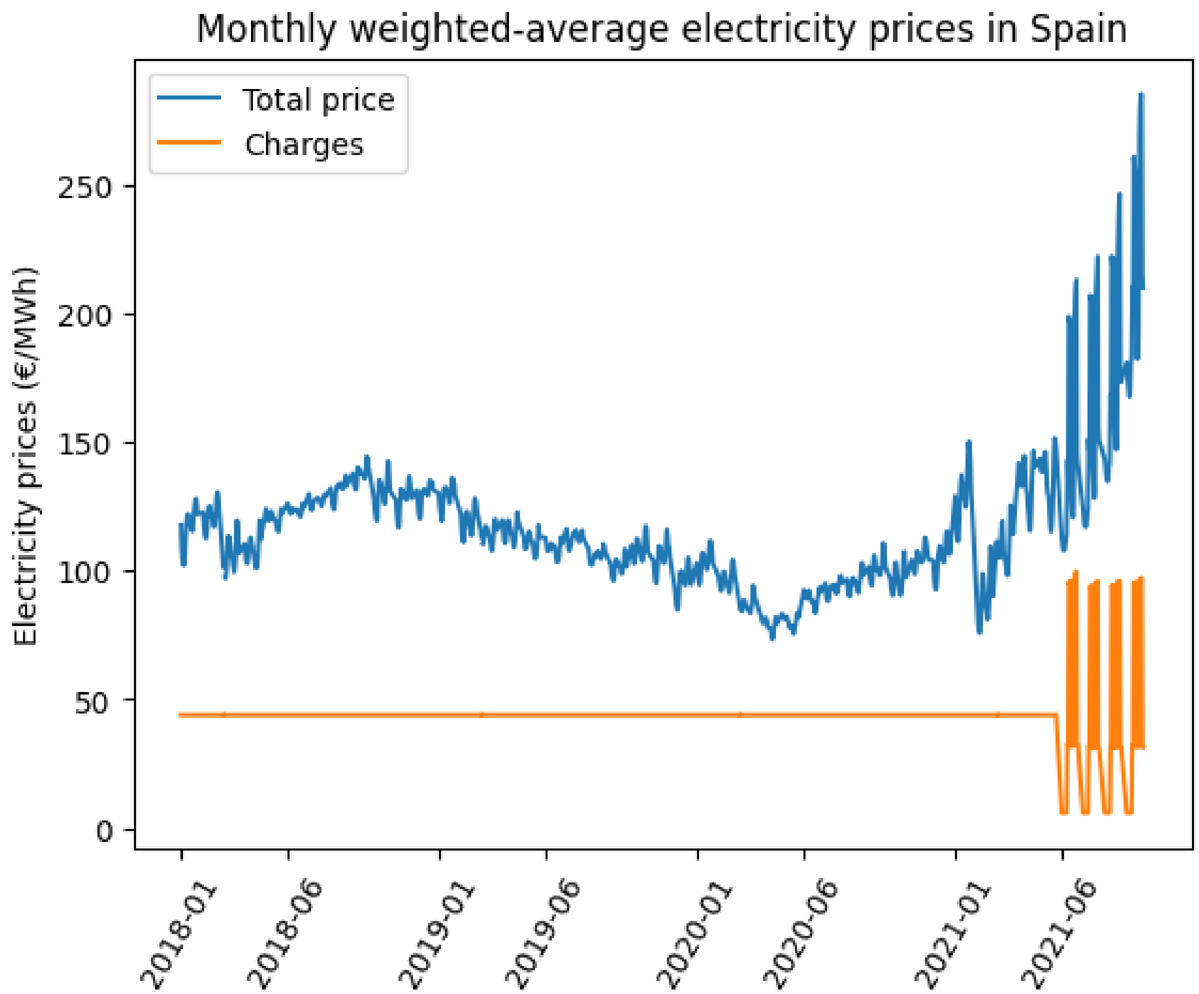
Outcomes:

- demand: demand (in mwh) at the distribution level
- consumer: number of consumers at the distribution level
- demand_pc (in kwh): demand / consumers

```
df_tou.describe()
```

✓ 0.0s

policy	placebo	temp	total_price	temph	charges	log_demand_pc	demand_pc	demand
01.000000	194601.000000	194597.000000	162196.000000	194597.000000	162196.00000	194581.000000	194581.000000	194581.000000
0.065241	0.019116	15.423593	115.142563	0.238786	44.09815	-1.340757	0.274592	566.948186
0.246952	0.136933	6.845133	29.552857	0.426343	14.40830	0.311960	0.086480	520.716513
0.000000	0.000000	-5.112264	18.570000	0.000000	6.00000	-2.383460	0.092231	20.800000
0.000000	0.000000	10.341200	99.000000	0.000000	44.03000	-1.550165	0.212213	62.200000
0.000000	0.000000	14.686452	112.410000	0.000000	44.03000	-1.327339	0.265182	439.800000
0.000000	0.000000	19.733700	126.710000	0.000000	44.03000	-1.127303	0.323906	902.800000
1.000000	1.000000	44.625421	338.620000	1.000000	133.12000	-0.198091	0.820295	3163.900000



Diff-in-diff will exploit sudden large change in prices

Differences-in-differences

To identify the potential demand response to the policy, we will estimate a **DiD model**, where our policy variable equals one for all Spanish distribution areas after the policy was implemented.

Moreover, we will identify an effect for each of the TOU tariffs: **Off-peak, Mid-Peak, and Peak hours**.

The regressions will have many controls, so I create here a simple function to run regressions with many fixed effects (there might be other solutions in Python, I use `reghdfe` in Stata/R)

```
# DID model: i(tou, policy) = one coefficient per tou level for policy (same idea as policy & tou)
# ref=None => "no reference category" (fully saturated interaction block)
fml = """
log_demand_pc ~
    i(tou, policy) +
    i(tou, placebo) +
    temp * temph
  | dist^month^hour^tou + dist^year^tou^hour + month_count^tou^hour
"""

# Two-way clustered SEs: pass a dict {"CRV1": "dist + month"} (supports up to two-way) :contentReference[oaicite:1]{index=1}
# weights_type: choose 'aweights' vs 'fweights' explicitly :contentReference[oaicite:2]{index=2}
m = pf.feols(
    fml=fml,
    data=df_reg,
    weights="consumer",
    weights_type="aweights",          # try "fweights" if consumer is a frequency/count weight
    vcov={"CRV1": "dist + month"},
)

m.summary()
```

This basic DiD compares Portugal and Spain before and after, the Placebo is one month before the policy

See code to also get weekday/weekend effects.

Estimation: OLS
Dep. var.: log_demand_pc, Fixed effects: dist^month^hour^tou+dist^year^tou^hour+month_count^tou^hour
Inference: CRV1
Observations: 142000

Coefficient	Estimate	Std. Error	t value	Pr(>|t|)	2.5%	97.5%
temp	-0.013	0.003	-4.122	0.009	-0.022	-0.005
temph	-0.548	0.162	-3.380	0.020	-0.964	-0.131
C(tou)[1]:policy	0.000	0.036	0.003	0.998	-0.093	0.094
C(tou)[2]:policy	-0.056	0.028	-2.002	0.102	-0.128	0.016
C(tou)[3]:policy	-0.103	0.022	-4.729	0.005	-0.159	-0.047
C(tou)[1]:placebo	0.047	0.015	3.109	0.027	0.008	0.086
C(tou)[2]:placebo	0.008	0.013	0.644	0.548	-0.025	0.042
C(tou)[3]:placebo	-0.006	0.013	-0.467	0.660	-0.039	0.027
temp:temph	0.028	0.008	3.512	0.017	0.007	0.048

RMSE: 93.702 R2: 0.954 R2 Within: 0.157

Classifying HHs with millions of consumers can be really helpful (research or business analytics!)

Follow-up exercises

1. Consider classifying households into types, using the k-means method (note: here you have a limited sample, so the approach is just for illustrative purposes).
2. Think about the assumptions behind the diff-in-diff comparison between utilities and the role of fixed effects, which are making the comparison between similar hours. How does the interpretation change as we change the fixed effects? [This might be easier after QSM II - Part 2]