Day 4: Demand I

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

We will replicate some of the results in "Estimating the Elasticity to Real Time Pricing," by Fabra, Rapson, Reguant, and Wang.

Loading **packages**. Here we load RCall, which will allow us to use some libraries in R.

```
begin
    using PlutoUI
    using DataFrames   , Statistics   ,Missings
    using CSV
    using Plots
    using StatsPlots
    using RCall
    using Binscatters
    using StatsBase
    using FixedEffectModels
end

```

Data exploration

Loading **data**.

- data_rtp.csv: Smart meter data of a small sample of 40 consumers.

The data is already merged with several other hourly data that can be either consumer specific (weather) or market specific (solar, wind, prices).

	id	rtp	tou	date	y	m	dom	hr	more
1	8	1	0	20563	2016	4	19	2	
2	8	1	0	20563	2016	4	19	3	
3	8	1	0	20563	2016	4	19	4	
4	8	1	0	20563	2016	4	19	5	
5	8	1	0	20563	2016	4	19	6	

```
begin
    mydata = CSV.read("data_rtp.csv", DataFrame)
    mydata = dropmissing(mydata)
    first(mydata, 5)
end
```

```
Summary Stats:
Length:         380063
Missing Count:  0
Mean:           0.107055
Minimum:        0.060730
1st Quartile:   0.096970
Median:         0.105800
3rd Quartile:   0.116400
Maximum:        0.182050
```

```
summarystats(mydata.price)
```

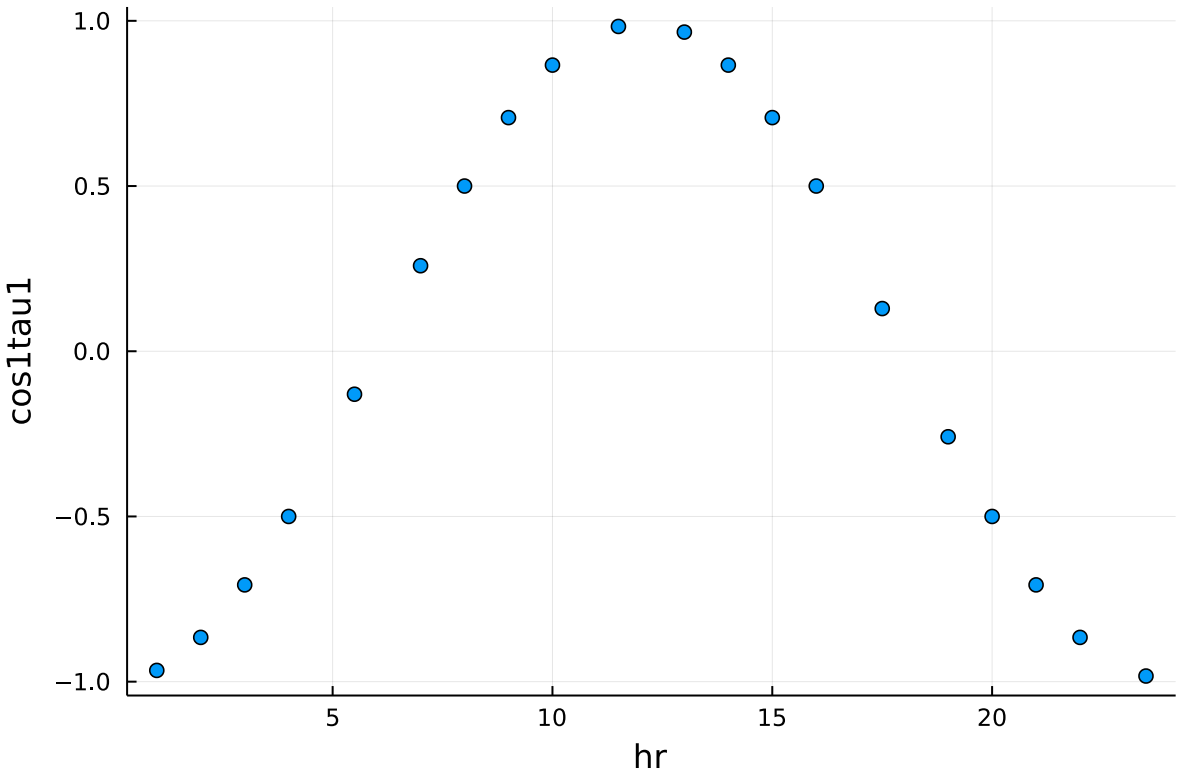
	variable	mean	min	median	max	nmissing	eltype	more
1	:id	3395.14	8	3733.0	7057	0	Int64	
2	:rtp	0.520608	0	1.0	1	0	Int64	
3	:tou	0.0	0	0.0	0	0	Int64	
4	:date	20708.6	20454	20703.0	20982	0	Int64	
5	:y	2016.31	2016	2016.0	2017	0	Int64	
6	:m	5.21937	1	4.0	11	0	Int64	
7	:dom	15.4816	1	15.0	31	0	Int64	
8	:hr	12.5732	1	13.0	24	0	Int64	
9	:wk	36.8488	1	36.0	76	0	Int64	
10	:dow	3.01718	0	3.0	6	0	Int64	
18	:mwh_dayaheadiberia	26605.9	0.0	26642.3	38955.6	0	Float64	

```
describe(mydata)
```

We create some Fourier transforms of time in order to capture temporal frequencies at daily, weekly and yearly levels due to seasonality.

	id	rtp	tou	date	y	m	dom	hr	more
1	8	1	0	20563	2016	4	19	2	
2	8	1	0	20563	2016	4	19	3	
3	8	1	0	20563	2016	4	19	4	
4	8	1	0	20563	2016	4	19	5	
5	8	1	0	20563	2016	4	19	6	

```
begin
    mydata.t = mydata.date + mydata.hr/24 .- 20560.0;
    for x in [1, 7, 365]
        mydata[!, string("tau",x)] = ((mydata.t .+ 0.5)/x) * π * 2.0
        mydata[!, string("tau2",x)] = mydata[!, string("tau",x)].^2.0
        for k = 1:4
            mydata[!, string("cos",k,"tau",x)] = cos.(mydata[!, string("tau",x)]/k)
        end
    end
    select!(mydata,Not(:t));
    first(mydata, 5)
end
```



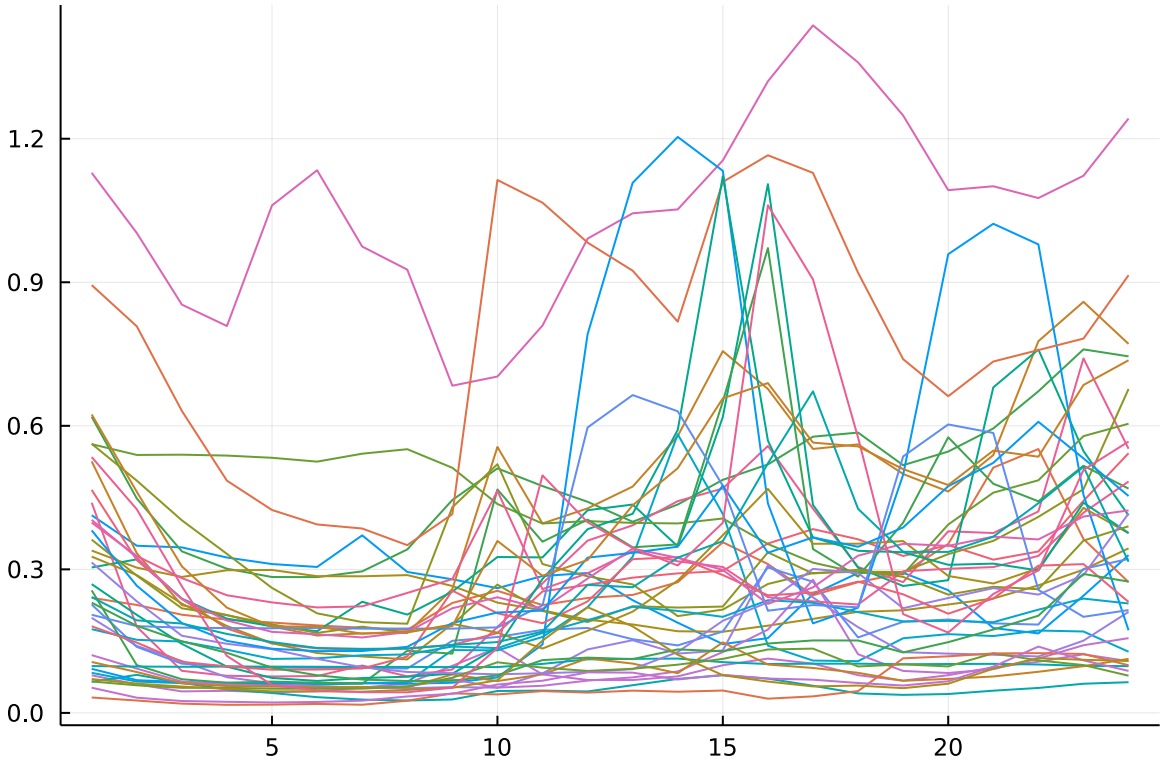

```
binscatter(mydata,@formula(cos1tau1~hr))
```

We rescale variables to avoid problems with Lasso, which is also sensitive to scaling!

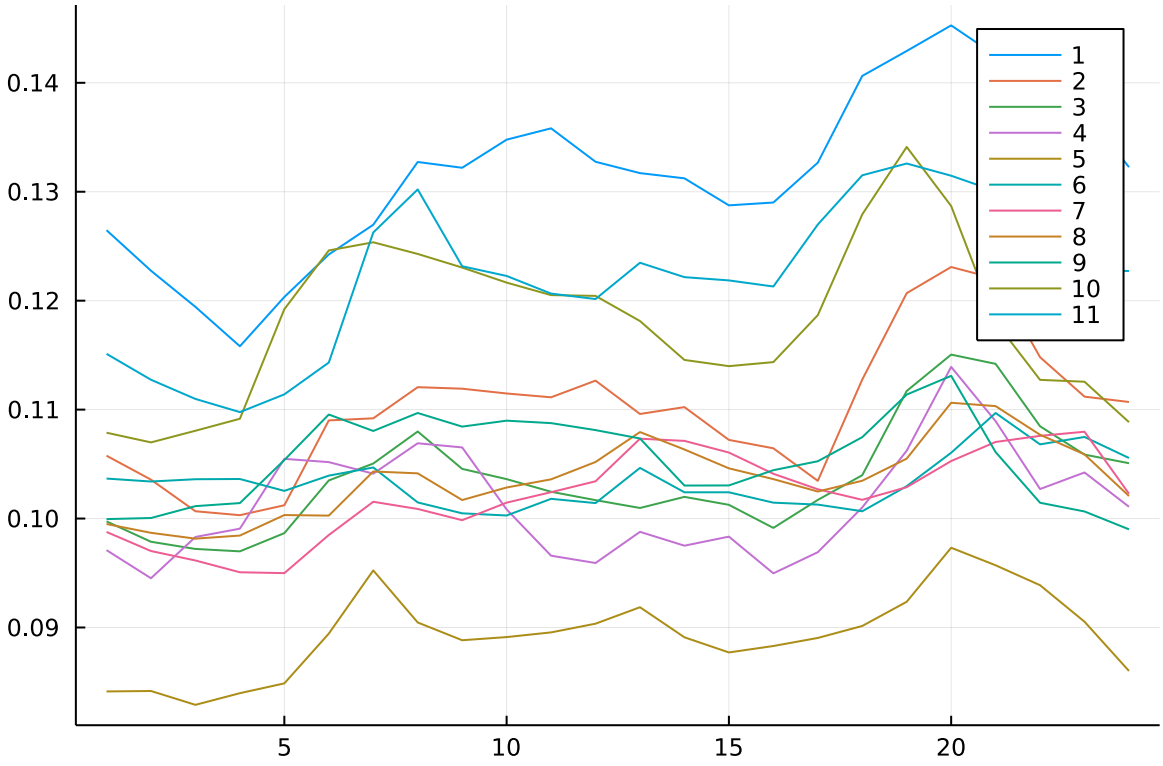
	id	rtp	tou	date	y	m	dom	hr	more
1	8	1	0	20563	2016	4	19	2	
2	8	1	0	20563	2016	4	19	3	
3	8	1	0	20563	2016	4	19	4	
4	8	1	0	20563	2016	4	19	5	
5	8	1	0	20563	2016	4	19	6	

```
begin
    # we rescale some variables as lasso can go "bananas". We do not rescale price
    and kwh because we will be using the log
    for v in names(mydata[!,Between(:wind_hat,:cos4tau365)])
        mu = mean(mydata[!,v]);
        sigma = std(mydata[!,v]);
        mydata[!,v] = (mydata[!,v] .- mu)/sigma
    end
    first(mydata, 5)
end
```

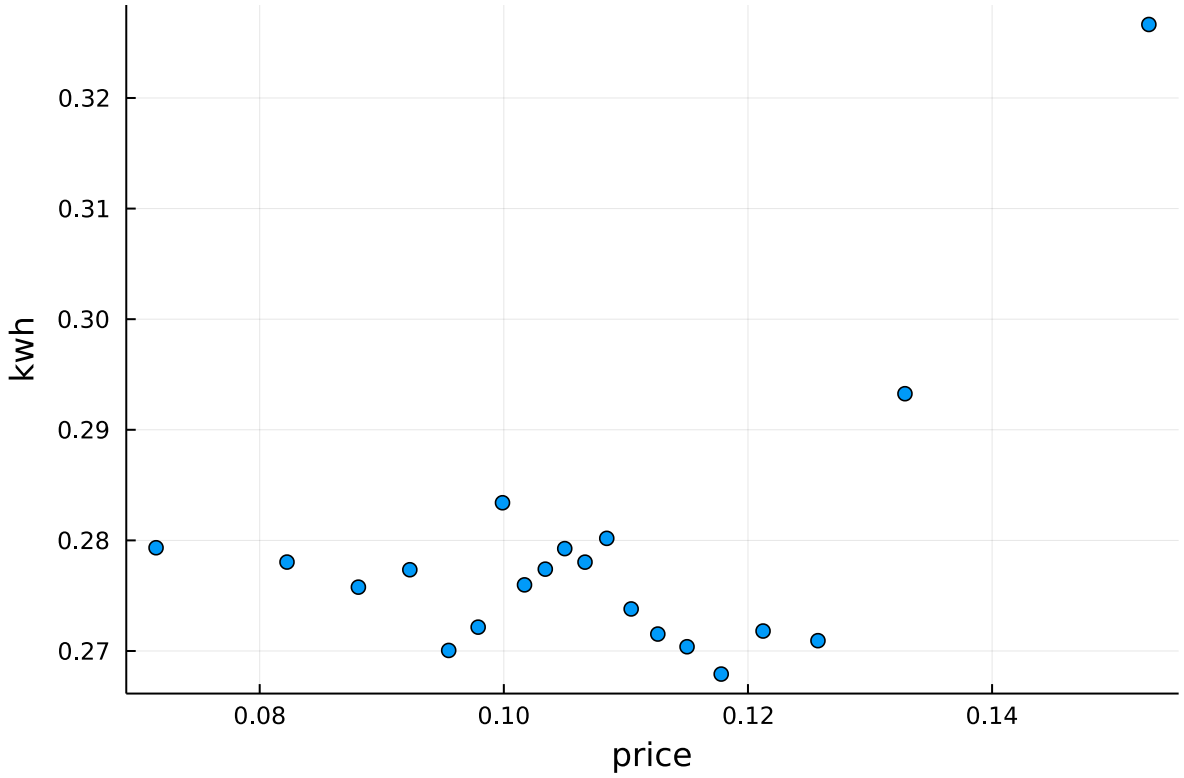
It can be useful to plot the data to examine patterns. We can plot the typical consumption pattern of consumers during the day.



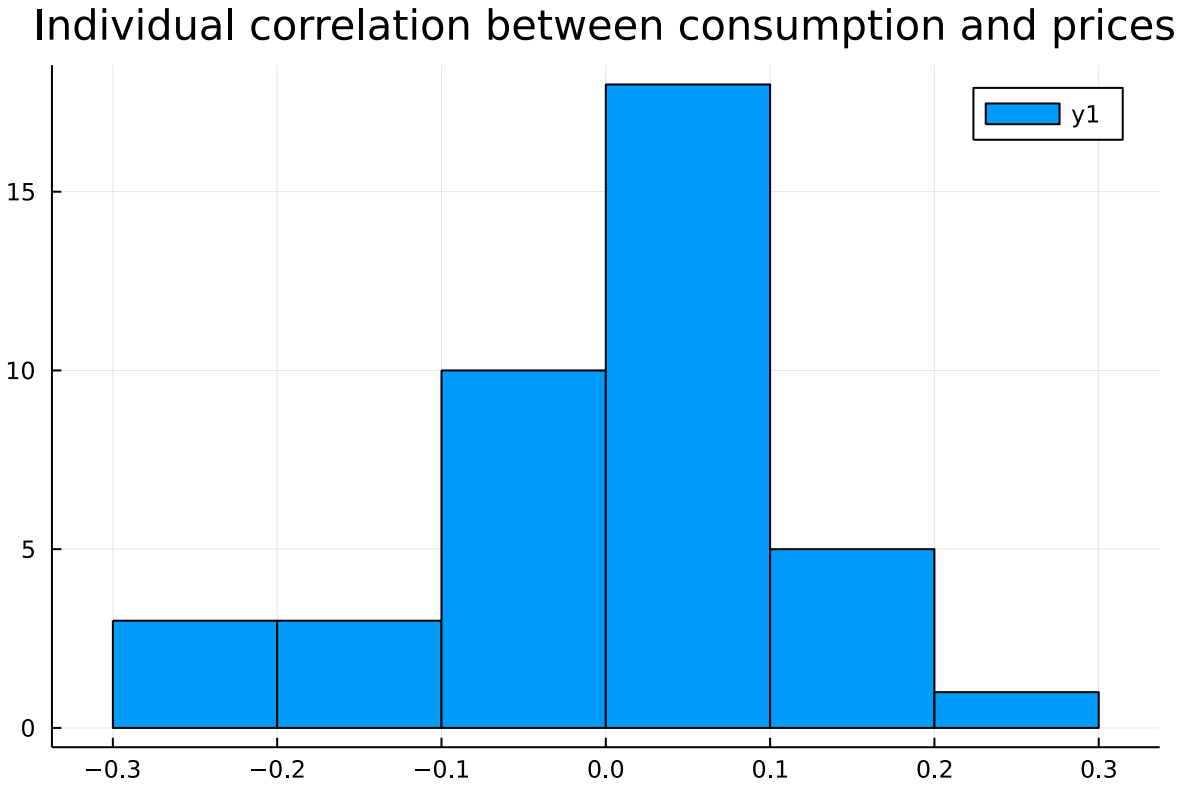

```
let
    # here we can learn about ways of collapsing data in Julia
    # lots going on here! see suggested exercise below

    df_plt = select(mydata,[:id,:hr,:kwh])
    df_plt = combine(groupby(df_plt, [:id,:hr]), :kwh => mean)
    @df df_plt plot(:hr, :kwh_mean, group=:id, legend=false)

end
```



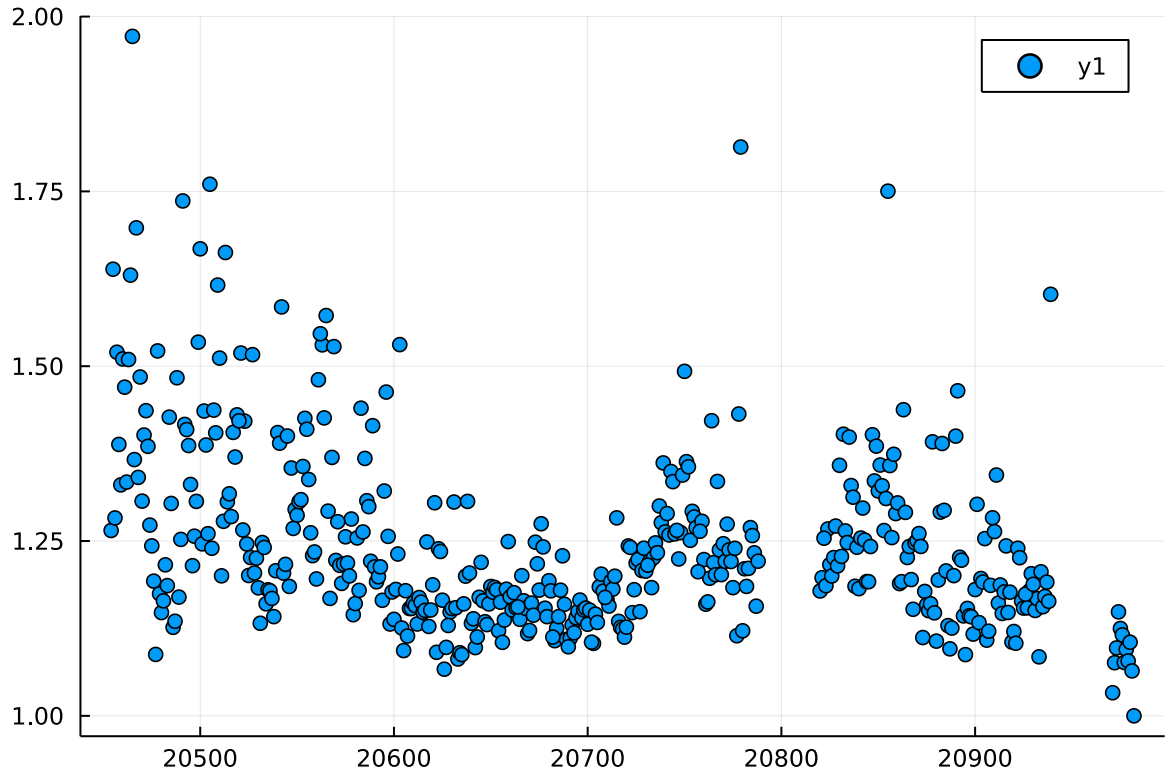

```
let
    # we can also plot prices
    df_plt = select(mydata,[:hr,:price, :m])
    df_plt = combine(groupby(df_plt, [:hr,:m]), :price => median)
    @df df_plt plot(:hr, :price_median, group=:m)

end
```




```
let
    # we can plot the raw correlation of consumption and prices
    # interesting that not particularly correlated with price
    binscatter(mydata,  @formula(kwh ~ price + fe(id)))
end
```




```
let
    # let's look at individual correlations
    ids = unique(mydata.id);
    corrs = [cor(mydata[mydata.id.==i,:].price,
            mydata[mydata.id.==i,:].kwh) for i in ids];
    histogram(corrs, title="Individual correlation between consumption and prices")
end
```




```
let
    # let's look at temporal correlations
    dates = unique(mydata.date);
    corrs = [maximum(mydata[mydata.date.==i,:].price)/
        minimum(mydata[mydata.date.==i,:].price) for i in dates];
    #histogram(corrs, title="Temporal correlation between consumption and prices")
    scatter(dates, corrs)
#   Plots.savefig("scatter_diffs.pdf");
end
```

467

```
length(unique(mydata.date))
```

Estimation of elasticities

We will be running the regression in R. Julia allows you to call R packages, as well as have access to R REPL mode.

- Make sure you have the needed libraries installed in R (`hdm`, `dplyr`).
- Use **@rput** to send Julia objects to R.

	id	rtp	tou	date	y	m	dom	hr	more
1	8	1	0	20563	2016	4	19	2	
2	8	1	0	20563	2016	4	19	3	
3	8	1	0	20563	2016	4	19	4	
4	8	1	0	20563	2016	4	19	5	
5	8	1	0	20563	2016	4	19	6	
6	8	1	0	20563	2016	4	19	7	
7	8	1	0	20563	2016	4	19	8	
8	8	1	0	20563	2016	4	19	9	
9	8	1	0	20563	2016	4	19	10	
10	8	1	0	20563	2016	4	19	11	
more									
380063	7057	0	0	20981	2017	6	11	12	

```
begin
    @rput mydata
end
```

```
"solar_actual + temp + temp2 + mwh_dayaheadiberia + tau1 + tau21 + cos1tau1 + cos2tau1 +
```

```
begin
    controls = join(names(mydata[!,Between(:solar_actual,:cos4tau365)]), " + ");
    @rput controls
end
```

To insert a chunk of R code, use R and triple quotes.

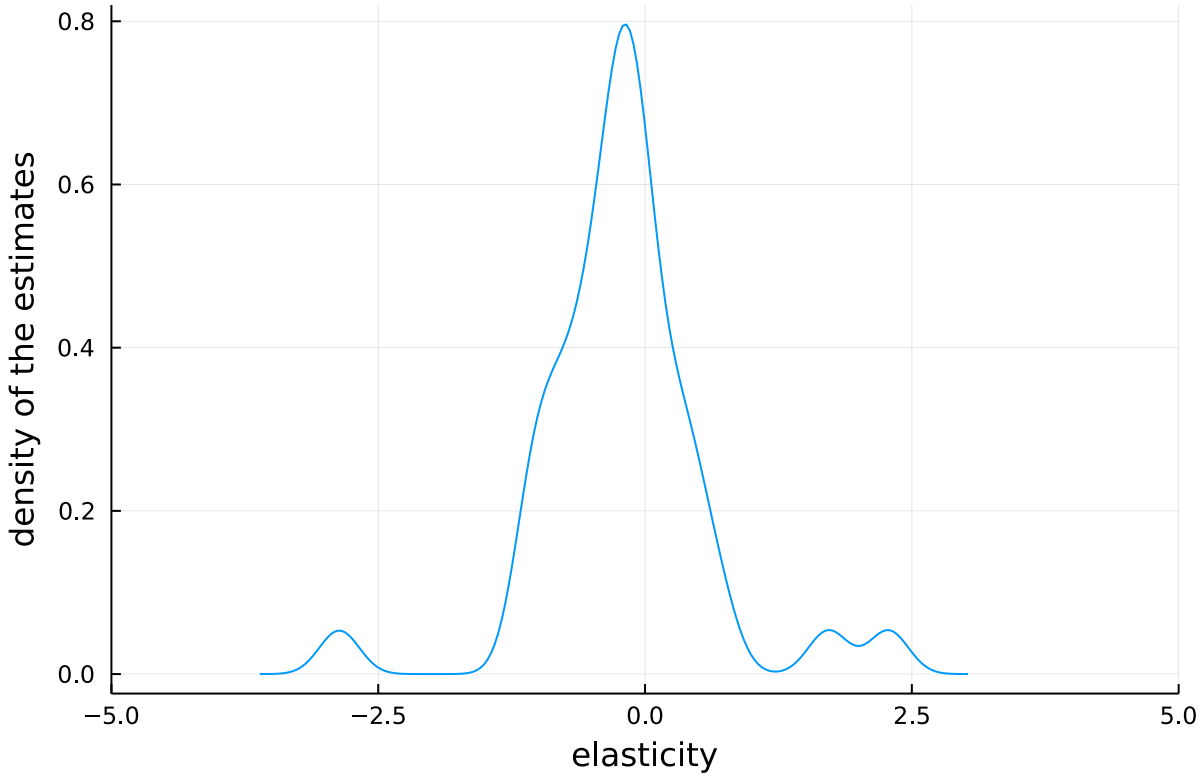
Note that in the following chunk, the loop is defined in Julia, while the regression calls an R package.

We run a separate regression per household.

```
[-0.226943, -0.20244, -0.422858, 2.2808, -2.86462, -0.527884, -0.185933, -0.775516, -0.2
```

```
begin
R"""
library(hdm)
library(dplyr)

sample = unique(mydata$id)
beta_Hh=rep(0,length(sample))
se_Hh=rep(0,length(sample))



for (i in 1:length(sample)){
        iv.reg = rlassoIV(as.formula(paste0("log(kwh+.01) ~
            log(price+.01) + ", controls, " |
            wind_hat + ", controls)),
            data = filter(mydata,id==sample[i]),
            select.X = TRUE, select.Z = FALSE)

        beta_Hh[i]<-iv.reg$coef
        }

 """
 @rget beta_Hh

end
```

We can plot the distribution of our household estimates




```
begin
    density(beta_Hh,legend = false)
    plot!(xlab="elasticity",ylab="density of the estimates",xlim=(-5,5))
end
```

`rlassoIV` is partialling out the effect of the controls from the endogenous variable, the outcome variable and the instrument. Second, it then uses the residuals to compute the IV estimator (2SLS). In theory, we could have partially out using OLS. The difference is that the LASSO method first selects the relevant variables in each case (and those could be different). For example, we can check which of the controls is relevant for the outcome variable of the first household in our sample:

```
RObject{VecSxp}

Call:
rlasso.formula(formula = fmla.y, data = filter(mydata, id ==
    unique(mydata$id)[1]))

Coefficients:
       (Intercept)        solar_actual                temp               temp2
         -2.143096           -0.017095          -0.011903            0.000000
mwh_dayaheadiberia                tau1               tau21            cos1tau1
          0.048072            0.000000            0.000000            0.000000
           cos2tau1            cos3tau1            cos4tau1                tau7
           0.000000            0.000000           -0.006185            0.000000
              tau27            cos1tau7            cos2tau7            cos3tau7
           0.000000            0.000000           -0.005330           -0.005858
           cos4tau7              tau365              tau2365          cos1tau365
           0.006777            0.000000            0.000000           -0.040636
         cos2tau365          cos3tau365          cos4tau365
          -0.076222            0.000000            0.000000
```

```
begin
R"""
library(hdm)
library(dplyr)

fmla.y = as.formula(paste0("log(price+.01) ~",controls))
Ylasso = rlasso(fmla.y, data = filter(mydata,id == unique(mydata$id)[1]))
summary(Ylasso)

"""

end
```

Only 5 out of the 22 possible controls are relevant in that case...

RTP vs Non-RTP elasticities

Next, we will check whether consumers under RTP have different price elasticities.

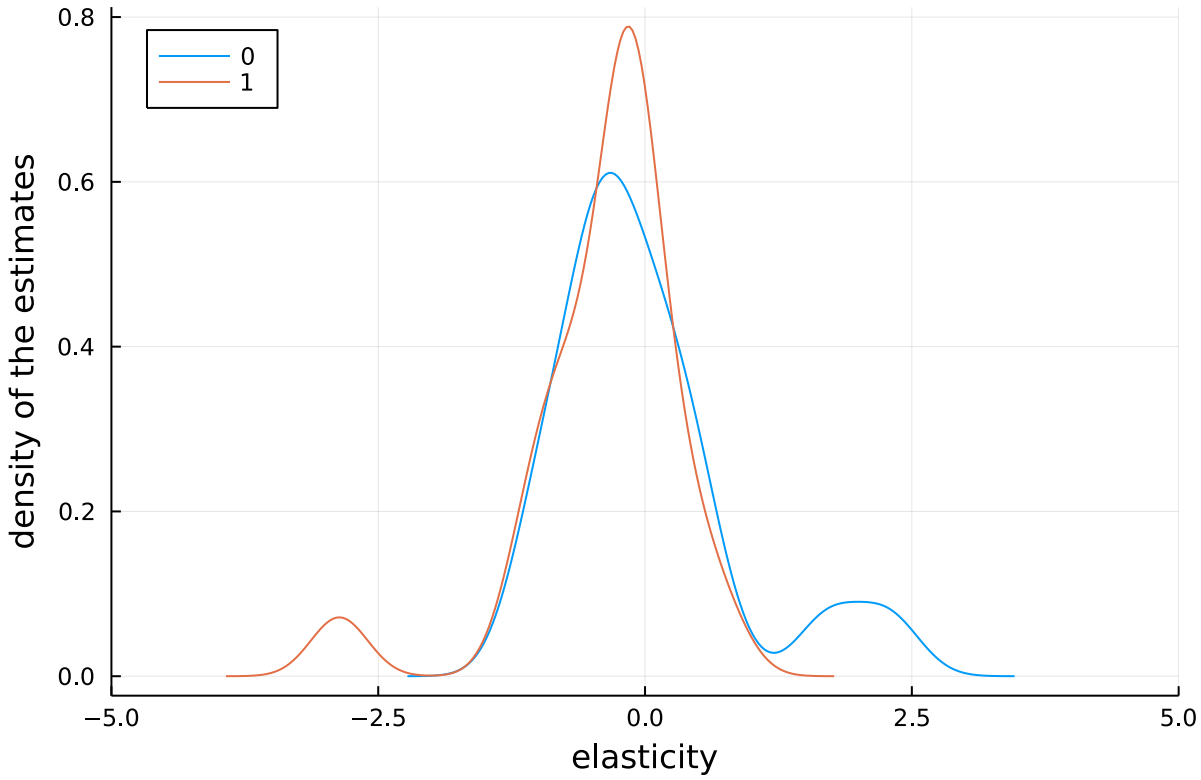
For that, we split our data between household with and without RTP.

The rest is similar than the previous exercise.

	estimate	rtp
1	-0.226943	1
2	-2.86462	1
3	-0.185933	1
4	-0.775516	1
5	-0.208226	1
6	0.312143	1
7	-1.06439	1
8	0.0153953	1
9	0.709645	1
10	-0.265946	1
	more	
40	-0.968756	0

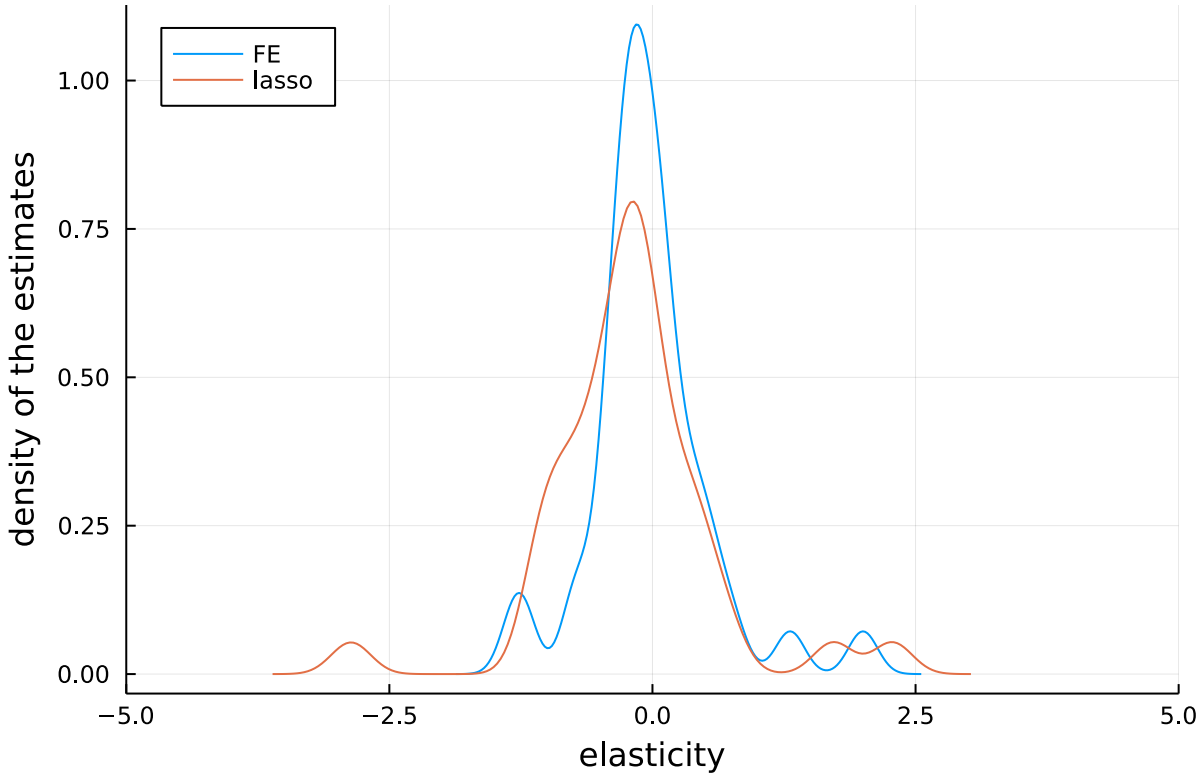
```
begin

    R"""
    library(hdm)
    library(dplyr)

    betas<-NULL

for (j in unique(mydata$rtp)){
  data_rtp = filter(mydata, rtp==j)
  sample_rtp = unique(data_rtp$id)
  beta_rtp=rep(0,length(sample_rtp))


  for (i in 1:length(sample_rtp)){
    iv.reg = rlassoIV(as.formula(paste0("log(kwh+.01) ~
                log(price+.01) + ", controls, " |
                wind_hat + ", controls)),
                      data = filter(data_rtp,id==sample_rtp[i] ),
                      select.X = TRUE, select.Z = FALSE)

    beta_rtp[i]<-iv.reg$coef
  }
  betas_j = data.frame(estimate=beta_rtp,rtp=j)
  betas = rbind(betas,betas_j)
}


     """
     @rget betas

end
```

Again, we can plot the density of each group of estimates and check whether there are significant differences.




```
begin
    @df betas density(:estimate, group = :rtp,legend = :topleft)
    plot!(xlab="elasticity",ylab="density of the estimates",xlim=(-5,5))
end
```

Finally, we can compare the LASSO results with an standard fixed effects regression. For that, instead of including the date Fourier transformations, we will directly include time fixed effects:




```
begin
    betas_fe = zeros(length(unique(mydata.id)))

for i in 1:length(unique(mydata.id))

reg_fe = reg(mydata[mydata.id .== unique(mydata.id)[i], :],
                        @formula(log(kwh +  1.0/100.0) ~ (log(price + 1/100) ~
wind_hat)
                                    + fe(hr)*fe(m) + fe(y) + fe(weekend)*fe(hr)
                                    + solar_actual + temp + temp2 +
mwh_dayaheadiberia
                                     ))

betas_fe[[i]] = coef(reg_fe)[coefnames(reg_fe) .=="log(price + 1 / 100)"]
end

df_beta = DataFrame(betas=[beta_Hh;betas_fe],
                    model=[repeat(["lasso"],
length(unique(mydata.id)));repeat(["FE"], length(unique(mydata.id)))])


@df df_beta density(:betas, group = :model,legend = :topleft)
plot!(xlab="elasticity",ylab="density of the estimates",xlim=(-5,5))



end
```

Follow-up exercises

1. Explore your ML method of choice as an alternative method to estimate the elasticities.

2(*). Include the consumption of non-RTP households as a potential control to the Lasso, as in Burlig et al. (2020). What can be some challenges without a pre-treatment period?

3(*). Use the Clustering.jl library we used on day 2 to classify consumers into "typical" profiles. This can be a useful way of reducing the dimensionality of the data. We will do something like this on day 5 as well.