Carbon Policy Surprises and Stock Returns: Signals from Financial Markets

Discussion

Alessandro Tang-Andersen Martinello
Head of Data Science
alem@nationalbanken.dk
alemartinello.com

DANMARKS NATIONALBANK

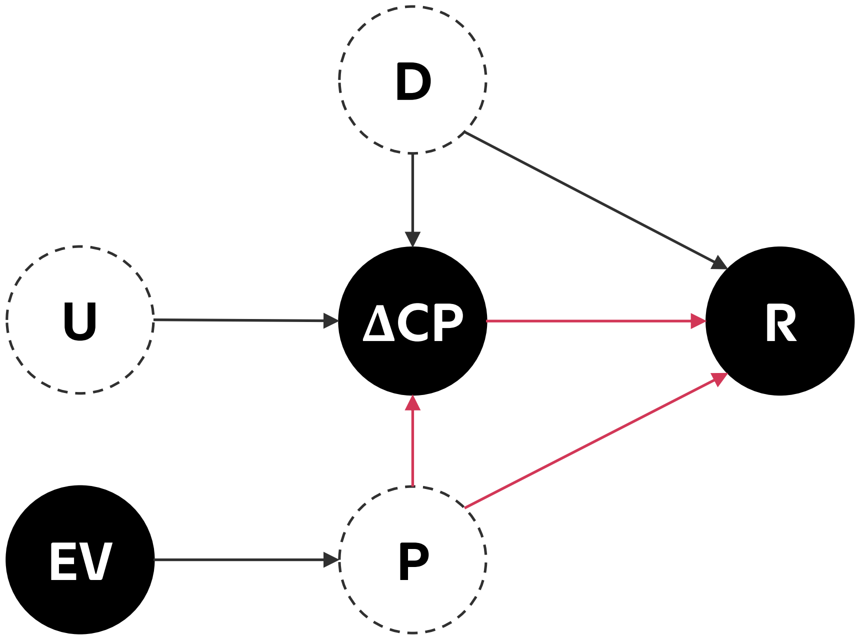
Hengge et al. is a smart piece of research providing a policy-relevant answer to a tough empirical problem

In research, your duty is providing a perfect answer to the question of your choice.
In policy, your duty is to answer a given question in the best way your data allows you to.

1	2	3
The focus is on a **policy relevant** quantity	Shows that the quantity is hard to estimate.	When life gives you lemons, add structure to the model.
Effect of carbon pricing ***policy*** on returns	The data and experiment are imperfect	Hengge et al. manage to provide an answer anyway

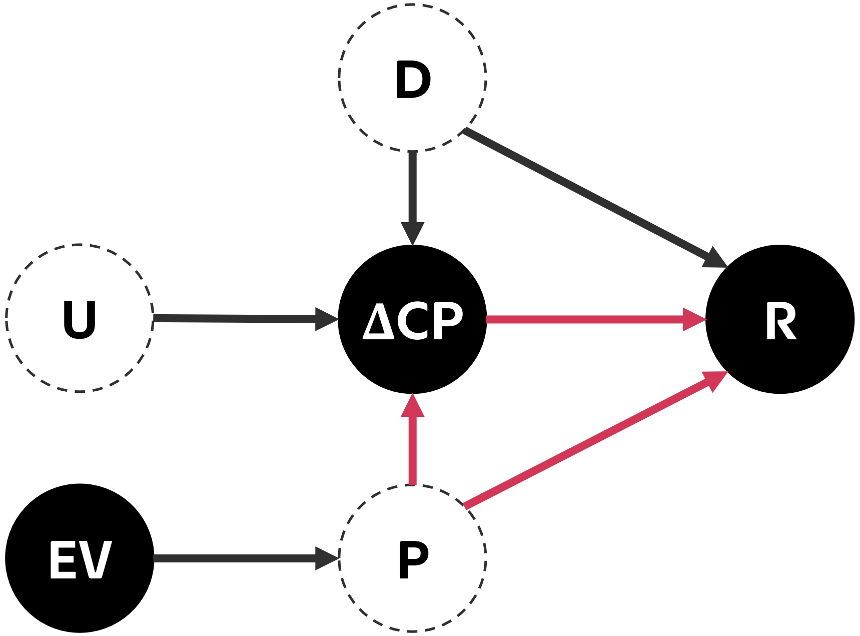
DANMARKS NATIONALBANK

The policy-relevant quantity can’t be identified…



The policy relevant question is not the *direct* effect of ΔCP to R, but the *overall* effect

The policy-relevant quantity can't be identified…
…unless we impose some structure to the problem

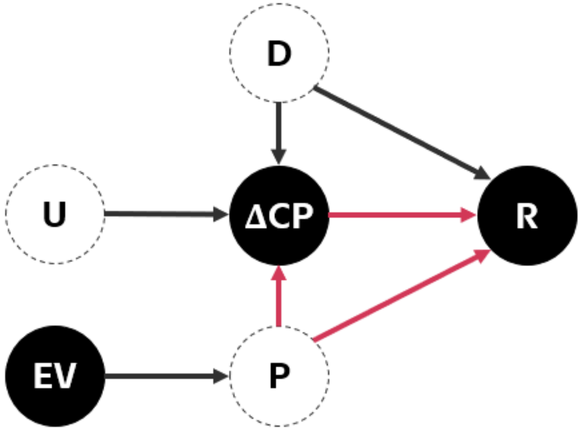


The policy relevant question is not the *direct* effect of ΔCP to R, but the *overall* effect

Identification needs to come from somewhere.

Can't be turtles all the way down

If the structure is linear the estimation becomes akin to a classical measurement error problem




```python
np.random.seed(2023-5-3)
# DAG
ev = 1*(np.random.normal(size=(n, 1))>0.8)
u = np.random.normal(size=(n, 1))
p = np.random.normal(size=(n, 1))*ev
d = np.random.normal(size=(n, 1))
cp = u + d + p
r = 1 + d - p - cp

# Throw stuff in observable & unobservable dataframe
obs = pd.DataFrame(np.hstack([cp, ev]), columns=['cp_1', 'ev_2'])
obs['int_3'] = cp*ev
unobs = pd.DataFrame(np.hstack([u, d, p]), columns=['u', 'd', 'p'])
```

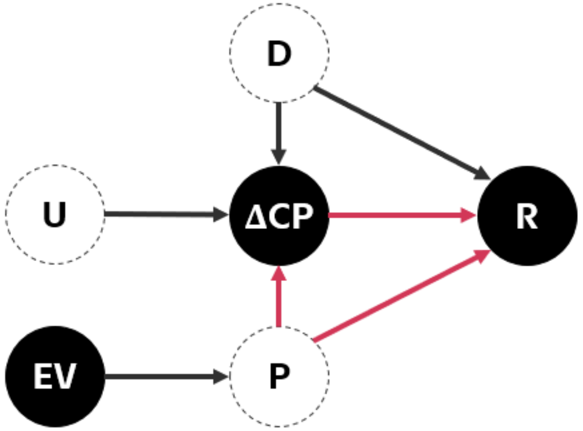
The bias can be expressed in closed-form solution, and corrected

```python
# Get true effect
mod = sm.OLS(r, sm.add_constant(unobs))
p_effect = mod.fit().params['p']
# Estimate k
k = np.var(cp[ev==1])/np.var(cp[ev==0])
# Estimate beta_1, beta_3, g
mod2 = sm.OLS(r, sm.add_constant(obs))
results2 = mod2.fit()
g = results2.params['cp_1'] + results2.params['int_3']*(k/(k-1))
print(f"True effect: {p_effect:.4f}\nEstimate: {g:.4f}")
```

```
True effect: -2.0000
Estimate: -2.0028
```

Code (and slides) here! →

If the structure is non-linear/non-separable the trick does not work anymore




```
np.random.seed(2023-5-3)
# DAG
ev = 1*(np.random.normal(size=(n, 1))>0.8)
u = np.random.normal(size=(n, 1))
p = np.random.normal(size=(n, 1))*ev
d = np.random.normal(size=(n, 1))
cp = u + d + p + 2*p*d
r = 1 + d - p - cp

# Throw stuff in observable & unobservable dataframe
obs = pd.DataFrame(np.hstack([cp, ev]), columns=['cp_1', 'ev_2'])
obs['int_3'] = cp*ev
unobs = pd.DataFrame(np.hstack([u, d, p]), columns=['u', 'd', 'p'])
```

The "measurement error" is not classical anymore

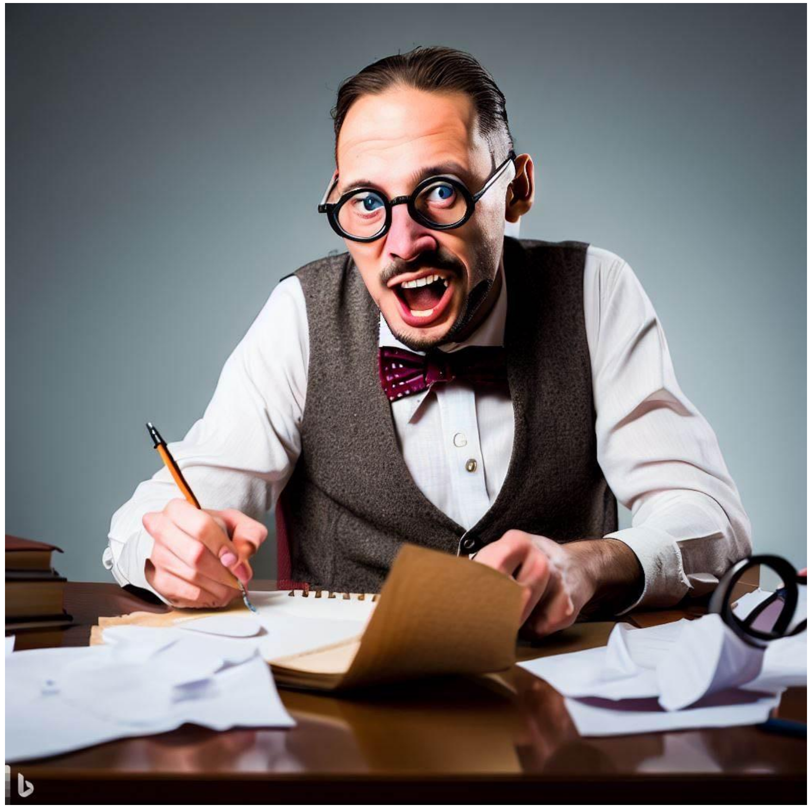
```
# Get true effect
mod = sm.OLS(r, sm.add_constant(unobs))
p_effect = mod.fit().params['p']
# Estimate k
k = np.var(cp[ev==1])/np.var(cp[ev==0])
# Estimate beta_1, beta_3, g
mod2 = sm.OLS(r, sm.add_constant(obs))
results2 = mod2.fit()
g = results2.params['cp_1'] + results2.params['int_3']*(k/(k-1))
print(f"True effect: {p_effect:.4f}\nEstimate: {g:.4f}")
```

```
True effect: -2.0051
Estimate: -1.1988
```

DANMARKS NATIONALBANK

Code (and slides) here! →

Some more Referee#2 comments



Note: No actual referees were harmed in the making of this picture. The image was generated by an AI engine. It may or may not provide an accurate representation of the refereeing process in the economic sciences

Leave-1-out robustness:

- Also for **events**
- Eventually by **industry**

Are you taking into account **uncertainty in the computation of k?**

Is the **effect for non-ECTS firms** (statistically) significantly different than ECTS firms?

Is the effect different for **scope1 VS scope2** emissions?

…