basic_sim

April 6, 2020

0.1 Basic mediation analysis with simulated data

[1]:
```
import numpy as np
import statsmodels.api as sm
import pandas as pd
```

```
/nfs/kshedden/python3/lib/python3.7/site-
packages/statsmodels/compat/pandas.py:23: FutureWarning: The Panel class is
removed from pandas. Accessing it from the top-level namespace will also be
removed in the next version
  data_klasses = (pandas.Series, pandas.DataFrame, pandas.Panel)
```

Make the simulation reproducible

[2]:
```
np.random.seed(2343)
```

The sample size

[3]:
```
n = 400
```

The function below simulates data that exhibits different types of mediation behavior. The type of mediation behavior is controlled by the 'mode' argument.

[4]:
```
def gendat(mode):
    """
    Generate data for demonstrating a mediation analysis.  Setting
    mode = 0, 1, 2, correspond, respectively, to no, full, and partial
    mediation, respectively.
    """

    # The exposure
    x = np.random.normal(size=n)

    # The mediator
    m = x + np.random.normal(size=n)
    m /= np.sqrt(2)

    if mode == 0:
        # No mediation
        y = x + np.random.normal(size=n)
    elif mode == 1:
```

```
        # Full mediation
        y = m + np.random.normal(size=n)
    else:
        # Partial mediation
        y = m + x + np.random.normal(size=n)

    return pd.DataFrame({"x": x, "m": m, "y": y})
```

The function below carries out a simplified mediation analysis. The purpose of this analysis is to illustrate the main idea behind how estimates of mediation are constructed. It omits a few important but technical steps for the sake of clarity.

[5]:
```
def fake_mediation(mode):
    """
    Conduct a simplified mediation analysis.  This shows the most
    important steps, but is incomplete since is treats the fitted
    models as being the exactly equal to the population.
    """

    df = gendat(mode)
    m_model = sm.OLS.from_formula("m ~ x", data=df).fit()
    o_model = sm.OLS.from_formula("y ~ x + m", data=df).fit()

    # Create counterfactual mediator values, forcing the exposure
    # to be low.
    df_xlow = df.copy()
    df_xlow.x = 0
    m_xlow = m_model.predict(exog=df_xlow)
    m_xlow += np.sqrt(m_model.scale) * np.random.normal(size=n)

    # Create counterfactual mediator values, forcing the exposure
    # to be high.
    df_xhigh = df.copy()
    df_xhigh.x = 1
    m_xhigh = m_model.predict(exog=df_xhigh)
    m_xhigh += np.sqrt(m_model.scale) * np.random.normal(size=n)

    # Create counterfactual outcomes for the indirect effect.
    df0 = df.copy()
    df0["x"] = 0
    df0["m"] = m_xlow
    y_low = o_model.predict(exog=df0)
    y_low += np.sqrt(o_model.scale) * np.random.normal(size=n)
    df0["x"] = 0
    df0["m"] = m_xhigh
    y_high = o_model.predict(exog=df0)
    y_high += np.sqrt(o_model.scale) * np.random.normal(size=n)
```

```
    # The average indirect effect
    aie = np.mean(y_high - y_low)
    aie_se = np.std(y_high - y_low) / np.sqrt(n)

    # Create counterfactual outcomes for the direct effect.
    df0 = df.copy()
    df0["x"] = 0
    df0["m"] = m_xlow
    y_low = o_model.predict(exog=df0)
    y_low += np.sqrt(o_model.scale) * np.random.normal(size=n)
    df0["x"] = 1
    y_high = o_model.predict(exog=df0)
    y_high += np.sqrt(o_model.scale) * np.random.normal(size=n)

    # The average direct effect
    ade = np.mean(y_high - y_low)
    ade_se = np.std(y_high - y_low) / np.sqrt(n)

    return aie, aie_se, ade, ade_se
```

Run the simplified mediation analysis for each type of mediation (no mediation, full mediation, partial mediation).

[6]:
```
for mode in 0, 1, 2:
    aie, aie_se, ade, ade_se = fake_mediation(mode)
    print("AIE=%8.4f (%.4f)  ADE=%8.4f (%.4f)" % (aie, aie_se, ade, ade_se))
```

```
AIE= -0.0504 (0.0700)  ADE=  1.0401 (0.0685)
AIE=  0.8445 (0.0926)  ADE= -0.0184 (0.0732)
AIE=  0.7475 (0.0844)  ADE=  1.0485 (0.0729)
```

Run a mediation analysis using the Mediation package for each type of mediation (no/full/partial).

[7]:
```
for mode in 0, 1, 2:

    df = gendat(mode)
    outcome_model = sm.OLS.from_formula("y ~ x + m", data=df)
    mediator_model = sm.OLS.from_formula("m ~ x", data=df)
    med = sm.stats.Mediation(outcome_model, mediator_model, "x", "m").
 ↪fit(n_rep=100)
    print(med.summary(), "\n")
```

	Estimate	Lower CI bound	Upper CI bound	P-value
ACME (control)	-0.057994	-0.160131	0.048136	0.3
ACME (treated)	-0.057994	-0.160131	0.048136	0.3
ADE (control)	1.026008	0.888199	1.184337	0.0
ADE (treated)	1.026008	0.888199	1.184337	0.0
Total effect	0.968015	0.870300	1.084963	0.0

Prop. mediated (control)	-0.056700	-0.163011	0.048579	0.3
Prop. mediated (treated)	-0.056700	-0.163011	0.048579	0.3
ACME (average)	-0.057994	-0.160131	0.048136	0.3
ADE (average)	1.026008	0.888199	1.184337	0.0
Prop. mediated (average)	-0.056700	-0.163011	0.048579	0.3

	Estimate	Lower CI bound	Upper CI bound	P-value
ACME (control)	0.824715	0.694464	0.989623	0.00
ACME (treated)	0.824715	0.694464	0.989623	0.00
ADE (control)	-0.137848	-0.282513	-0.033049	0.02
ADE (treated)	-0.137848	-0.282513	-0.033049	0.02
Total effect	0.686866	0.515419	0.864148	0.00
Prop. mediated (control)	1.185473	1.046777	1.457412	0.00
Prop. mediated (treated)	1.185473	1.046777	1.457412	0.00
ACME (average)	0.824715	0.694464	0.989623	0.00
ADE (average)	-0.137848	-0.282513	-0.033049	0.02
Prop. mediated (average)	1.185473	1.046777	1.457412	0.00

	Estimate	Lower CI bound	Upper CI bound	P-value
ACME (control)	0.678753	0.551686	0.820375	0.0
ACME (treated)	0.678753	0.551686	0.820375	0.0
ADE (control)	0.901873	0.738679	1.029238	0.0
ADE (treated)	0.901873	0.738679	1.029238	0.0
Total effect	1.580626	1.422505	1.705761	0.0
Prop. mediated (control)	0.432554	0.355283	0.505385	0.0
Prop. mediated (treated)	0.432554	0.355283	0.505385	0.0
ACME (average)	0.678753	0.551686	0.820375	0.0
ADE (average)	0.901873	0.738679	1.029238	0.0
Prop. mediated (average)	0.432554	0.355283	0.505385	0.0