Simpson's Paradox analyzed through Causal Reasoning

pyAgrum User day 15/06/2021

Santiago Cortijo, PhD, Head R&D santiago.cortijo@scalnyx.com
This presentation is based on:

1. **Understanding the Simpson’s Paradox**, by Judea Pearl (2013)

2. Causality Course, by Brady Neal:
   
   [https://www.bradyneal.com/causal-inference-course](https://www.bradyneal.com/causal-inference-course)
Correlation ≠ Causation
Motivating examples
Correlation ≠ Causation (1)

Sleep with shoes  Headache next morning.
Correlation ≠ Causation (2)

Heavy drinking before

Sleep with shoes

Headache next morning.
Correlation ≠ Causation (3)

- Heavy drinking before
- Confounder Association
- Sleep with shoes
- Causal Association
- Headache next morning.
Randomized Control Trials (RCT), (1)
Randomized Control Trials (RCT), (2)

RCT eliminates confounding association
Randomized Control Trials (RCT), (3)

Sadly, RCT can are often expensive, unethical or even impossible.
Observational studies

How do we measure causal effects
Motivating example: Treatment to choose (1)

T : Treatment (A or B)

Death rate table:

<table>
<thead>
<tr>
<th>T</th>
<th>Total</th>
</tr>
</thead>
</table>
| A | 16 %  
   (240 / 1500) |
| B | 19 %  
   (105 / 550) |

Decision Makers:
Need to invest in a Treatment. Which one? which is better, A or B?
Motivating example: Treatment to choose (2)

T : Treatment (A or B)

Death rate table, with mode details about symptoms intensity:

<table>
<thead>
<tr>
<th></th>
<th>Mild Symptoms</th>
<th>Severe Symptoms</th>
<th>Total ☠</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>15 % (210 / 1400)</td>
<td>30 % (30 / 100)</td>
<td>16 % (240 / 1500)</td>
</tr>
<tr>
<td>B</td>
<td>10 % (5 / 50)</td>
<td>20 % (100 / 500)</td>
<td>19 % (105 / 550)</td>
</tr>
</tbody>
</table>

Decision Makers:
*Wait… what!?
Motivating example: Treatment to choose (3)

T : Treatment (A or B)

<table>
<thead>
<tr>
<th>T</th>
<th>Mild</th>
<th>Severe</th>
<th>Calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>15 %</td>
<td>30 %</td>
<td>0.15 + 0.30 = 0.45</td>
</tr>
<tr>
<td></td>
<td>(210 / 1400)</td>
<td>(30 / 100)</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>10 %</td>
<td>20 %</td>
<td>0.10 + 0.20 = 0.30</td>
</tr>
<tr>
<td></td>
<td>(5 / 50)</td>
<td>(100 / 500)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>19 %</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(240 / 1500)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(105 / 550)</td>
<td></td>
</tr>
</tbody>
</table>
Motivating example: When is B preferred?

Causal Graph:
Symptom intensity (C) causes Treatment (T)

Mild Symptoms
Treatment A

Severe Symptoms
Treatment B
Motivating example: When is A preferable?

Causal Graph:
Treatment (T) causes Symptom intensity (C), due to waiting list.

Treatment A
Mild Symptoms

Treatment B
Severe Symptoms
Measure causal effects in observational data (1)

For our example, we adjust/control for confounders

<table>
<thead>
<tr>
<th>T</th>
<th>Mild</th>
<th>Severe</th>
<th>☠️</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>15 % (210 / 1400)</td>
<td>30 % (30 / 100)</td>
<td>16 % (240 / 1500)</td>
</tr>
<tr>
<td>B</td>
<td>10 % (5 / 50)</td>
<td>20 % (100 / 500)</td>
<td>19 % (105 / 550)</td>
</tr>
</tbody>
</table>

$E[Y \mid do(T = t)] = E_C E[Y \mid T = t]$
Measure causal effects in observational data (2)

\[ E[Y \mid do(T = t)] = E_C E[Y \mid T = t] \]

<table>
<thead>
<tr>
<th></th>
<th>Mild</th>
<th>Severe</th>
<th>☠</th>
<th>causal</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>15 % (210 / 1400)</td>
<td>30 % (30 / 100)</td>
<td>16 % (240 / 1500)</td>
<td>19.4 %</td>
</tr>
<tr>
<td>B</td>
<td>10 % (5 / 50)</td>
<td>20 % (100 / 500)</td>
<td>19 % (105 / 550)</td>
<td>12.9 %</td>
</tr>
</tbody>
</table>

\[
E[Y|t, C = 0] \quad E[Y|t, C = 1] \quad E[Y|t] \quad E[Y|do(t)]
\]

\[
\frac{1450}{2050} \times 0.15 + \frac{600}{2050} \times 0.30 = 0.194
\]

\[
\frac{1450}{2050} \times 0.10 + \frac{600}{2050} \times 0.20 = 0.129
\]
Done with pyAgrum (1)

Generating data

```python
[2]:

filename1 = 'data1.csv'
columns = ['Treatment', 'Symptoms', 'Outcome']
data = \n  [['A', 'mild', 'dead']]*210 + \n  [['A', 'mild', 'alive']]*(1400-210) + \n  [['A', 'severe', 'dead']]*30 + \n  [['A', 'severe', 'alive']]*(100-30) + \n  [['B', 'mild', 'dead']]*5 + \n  [['B', 'mild', 'alive']]*(50-5) + \n  [['B', 'severe', 'dead']]*100 + \n  [['B', 'severe', 'alive']]*(500-100)
shuffle(data)

df = pd.DataFrame(data=data, columns=columns)
df.to_csv(filename1, index=False)
df
```

2050 rows × 3 columns
Done with pyAgrum (2) : Learning

Learning model from data (using constraints)

```python
learner=gum.BN Learner(filename1)
learner.addMandatoryArc('Symptoms', 'Treatment')
learner.addMandatoryArc('Symptoms', 'Outcome')
learner.addMandatoryArc('Treatment', 'Outcome')

bn = learner.learnBN()
bn
```
Done with pyAgrum (3): Inference

```python
: gnb.flow.row(bn cpt('Treatment'), bn cpt('Symptoms'), bn cpt('Outcome'))
```

<table>
<thead>
<tr>
<th></th>
<th>Treatment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symptoms</td>
<td>A</td>
</tr>
<tr>
<td>mild</td>
<td>0.9654</td>
</tr>
<tr>
<td>severe</td>
<td>0.1669</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Symptoms</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mild</td>
</tr>
<tr>
<td>A</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.7072</td>
</tr>
<tr>
<td>B</td>
<td>0.7999</td>
</tr>
</tbody>
</table>

```
: ie=gum.LazyPropagation(bn)
ie.evidenceImpact("Outcome",[])
```

<table>
<thead>
<tr>
<th>Outcome</th>
<th>alive</th>
<th>dead</th>
</tr>
</thead>
<tbody>
<tr>
<td>alive</td>
<td>0.8315</td>
<td>0.1685</td>
</tr>
</tbody>
</table>

```
: ie=gum.LazyPropagation(bn)
ie.evidenceImpact("Outcome",["Treatment"])```

<table>
<thead>
<tr>
<th>Treatment</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
</tr>
<tr>
<td></td>
<td>alive</td>
</tr>
<tr>
<td></td>
<td>0.8399</td>
</tr>
<tr>
<td>B</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.8088</td>
</tr>
</tbody>
</table>

Look, that Simpson’s paradox still there!
Done with pyAgrum (4)

Generate (Draw) data from the model itself

```python
filename2 = 'data2.csv'
dbgen = gum.BNDatabaseGenerator(bn)
dbgen.drawSamples(2050)
dbgen.toCSV(filename2)

df2 = pd.read_csv(filename2)
df2
```

<table>
<thead>
<tr>
<th>Treatment</th>
<th>Symptoms</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A</td>
<td>mild</td>
</tr>
<tr>
<td>1</td>
<td>B</td>
<td>severe</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>severe</td>
</tr>
<tr>
<td>3</td>
<td>B</td>
<td>severe</td>
</tr>
<tr>
<td>4</td>
<td>B</td>
<td>severe</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>2045</td>
<td>B</td>
<td>severe</td>
</tr>
<tr>
<td>2046</td>
<td>A</td>
<td>mild</td>
</tr>
<tr>
<td>2047</td>
<td>B</td>
<td>severe</td>
</tr>
<tr>
<td>2048</td>
<td>A</td>
<td>mild</td>
</tr>
<tr>
<td>2049</td>
<td>A</td>
<td>mild</td>
</tr>
</tbody>
</table>

2050 rows × 3 columns
Done with pyAgrum (5)

Simpson’s paradox still there!

```
import pandas as pd

df2 = pd.read_csv(filename2)
ct = pd.crosstab(df2.Treatment, [df2.Outcome])
p_dead = ct['dead'] / (ct['alive'] + ct['dead'])
prob_dead = pd.DataFrame(p_dead.rename('prob(dead)'))

# Treatment probabilities

<table>
<thead>
<tr>
<th>Treatment</th>
<th>prob(dead)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.153428</td>
</tr>
<tr>
<td>B</td>
<td>0.164645</td>
</tr>
</tbody>
</table>
```

```
[15]:
df2 = pd.read_csv(filename2)
ct = pd.crosstab(df2.Treatment, [df2.Symptoms, df2.Outcome])
p_dead_mild = ct[['mild', 'dead']] / (ct[['mild', 'alive']] + ct[['mild', 'dead']])
p_dead_severe = ct[['severe', 'dead']] / (ct[['severe', 'alive']] + ct[['severe', 'dead']])
pd.concat([p_dead_mild.rename('mild'), p_dead_severe.rename('severe')], axis=1).rename_axis('Prob(dead)'), axis=1)

[15]:
<table>
<thead>
<tr>
<th>Prob(dead)</th>
<th>mild</th>
<th>severe</th>
</tr>
</thead>
<tbody>
<tr>
<td>Treatment</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>0.140671</td>
<td>0.326733</td>
</tr>
<tr>
<td>B</td>
<td>0.074074</td>
<td>0.173996</td>
</tr>
</tbody>
</table>
Done with pyAgrum (6)

```
learner=gum.BNLearner(filename1)
learner.addMandatoryArc('Symptoms', 'Treatment')
learner.addMandatoryArc('Symptoms', 'Outcome')
learner.addMandatoryArc('Treatment', 'Outcome')

bn = learner.learnBN()
bn

d1 = cs1.CausalModel(bn)
cslnb.showCausalImpact(d1, "Outcome", doing="Treatment",values={})
```

Causal Model

\[
P(\text{Outcome} \mid \uparrow \text{Treatment}) = \sum_{\text{Symptoms}} P(\text{Outcome} \mid \text{Symptoms, Treatment}) \cdot P(\text{Symptoms})
\]

Explanation: backdoor ['Symptoms'] found.

Impact: \( P(\text{Outcome} \mid \uparrow \text{Treatment}) \)

<table>
<thead>
<tr>
<th>Treatment</th>
<th>Outcome</th>
<th>alive</th>
<th>dead</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td>0.6059</td>
<td>0.1941</td>
</tr>
<tr>
<td>B</td>
<td></td>
<td>0.8693</td>
<td>0.1307</td>
</tr>
</tbody>
</table>

Mild Symptoms

Severe Symptoms

Treatment A

Treatment B
Done with pyAgrum (7)

```
learner=gum.BNLearner(filename)
learner.addMandatoryArc('Treatment', 'Symptoms')
learner.addMandatoryArc('Symptoms', 'Outcome')
learner.addMandatoryArc('Treatment', 'Outcome')

bn2 = learner.learnBN()
d2 = csl.CausalModel(bn2)
cslnb.showCausalImpact(d2, "Outcome", doing="Treatment", values={})
```

**Causal Model**

\[
P(\text{Outcome} \mid \sim \text{Treatment}) = \sum_{\text{Symptoms}} \frac{P(\text{Outcome} \mid \text{Symptoms}, \text{Treatment})}{P(\text{Symptoms} \mid \text{Treatment})}
\]

**Explanation:** Do-calculus computations

**Impact:** \(P(\text{Outcome} \mid \sim \text{Treatment})\)

<table>
<thead>
<tr>
<th>Treatment</th>
<th>Outcome</th>
<th>alive</th>
<th>dead</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.8399</td>
<td>0.1601</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>0.8088</td>
<td>0.1912</td>
<td></td>
</tr>
</tbody>
</table>
Some causal diagrams (like these) won’t ever generate Simpson’s paradoxes.
IF THERE IS A BACKDOOR, YOU MUST CONDITION ON IT.

IF THERE IS A BACKDOOR CHOOSE B
Done with pyAgrum (10): Backdoors

```python
learner=gum.BNLearner(filename1)
learner.addMandatoryArc('Symptoms', 'Treatment')
learner.addForbiddenArc('Symptoms', 'Outcome')
learner.addMandatoryArc('Treatment', 'Outcome')
bn6 = learner.learnBN()
d6 = csl.CausalModel(bn6, ["L1", ["Treatment", "Outcome"]], keepArcs=True)
cslnb.showCausalImpact(d6, "Outcome", doing="Treatment", values={})
```

IF THERE IS A LATENT VARIABLE AS BACKDOOR, THERE IS NOTHING TO DO.

Causal Model

Explanation: Hedge Error: G={"Treatment", "Outcome"}, G[S]={"Outcome"}

Impact: ?
Done with pyAgrum (11): Backdoors

\[
P(\text{Outcome} \mid \lnot \text{Treatment}) = \sum_{\text{Symptoms}} P(\text{Outcome} \mid \text{Symptoms}, \text{Treatment}) \cdot P(\text{Symptoms})
\]

Causal Model | Explanation: Do-calculus computations

IF THERE IS NO BACKDOOR, CHOOSE THE AGGREGATE.

\[
P(\text{Outcome} \mid \lnot \text{Treatment}) = P(\text{Outcome} \mid \text{Treatment})
\]
When facing the Simpson's Paradox:

1. Draw a causal Diagram
2. If there is a backdoor
   a. If it is observed, the right decision is to use it as conditioning variable
   b. If it is unobserved, no conclusion can be made.
3. If there is no backdoor, the right decision is to use the aggregate result.
Thank you.