SIC mit SOA-Manipulation

Matthias

16 Oct 2020

# 100% Accuracy bzw. Accuracy ignorieren

Ich suche hier Gegenbeispiele mit brute force, indem ich vier Zeiten sample und und dann die 4-Tupel extrahiere, für die stochastische Dominanz erfüllt wird, also und . Die Funktion ist etwas umständlich programmiert, damit ich sie später leicht für accuaracy verallgemeinern kann.

sdom = function(a, A)  
{  
 a$D > A$D  
}  
  
sim = function()  
{  
 a = list(D=sample(1:10, size=1))  
 A = list(D=sample(1:10, size=1))  
   
 b = list(D=sample(1:10, size=1) + 0.1)  
 B = list(D=sample(1:10, size=1) + 0.1)  
  
 # Stochastic dominance satisfied?  
 if(sdom(a, A) & sdom(b, B))  
 return(list(a=a, A=A, b=b, B=B))  
   
 # Otherwise, retry  
 sim()  
}  
  
# Example  
unlist(sim())

## a.D A.D b.D B.D   
## 8.0 3.0 5.1 3.1

Später wird jeder dieser 4-Tupel als Punktverteilung aufgefasst und ausprobiert, ob der IC eingehalten wird. Zunächst werden die Reaktionszeiten auf ab, AB, aB, Ab generiert (Wettlaufmodell).

# Winner of the race  
race = function(A, B)  
{  
 if(A$D <= B$D)  
 return(A)  
   
 return(B)  
}  
  
# Responses to ab, aB, Ab, AB  
resp = function(d)  
{  
 ab = race(d$a, d$b)  
 aB = race(d$a, d$B)  
 Ab = race(d$A, d$b)  
 AB = race(d$A, d$B)  
   
 list(ab=ab, aB=aB, Ab=Ab, AB=AB)  
}  
  
# Example  
d = sim()  
unlist(d)

## a.D A.D b.D B.D   
## 5.0 3.0 4.1 1.1

unlist(resp(d))

## ab.D aB.D Ab.D AB.D   
## 4.1 1.1 3.0 1.1

Jetzt wird der IC für einen Satz von RTs berechnet und geschaut, ob er kleiner-gleich Null ist.

# Interaction contrast  
ic = function(ab, aB, Ab, AB, t=1:10)  
{  
 Fab = ecdf(ab$D)(t)  
 FaB = ecdf(aB$D)(t)  
 FAb = ecdf(Ab$D)(t)  
 FAB = ecdf(AB$D)(t)  
   
 all(Fab + FAB <= FaB + FAb)  
}  
  
# Example  
d = sim()  
unlist(d)

## a.D A.D b.D B.D   
## 8.0 5.0 6.1 2.1

r = resp(d)  
unlist(r)

## ab.D aB.D Ab.D AB.D   
## 6.1 2.1 5.0 2.1

ic(r$ab, r$aB, r$Ab, r$AB)

## [1] TRUE

## 1000 times, interrupt if IC test failed  
for(i in 1:1000)  
{  
 d = sim()  
 r = resp(d)  
 if(!ic(r$ab, r$aB, r$Ab, r$AB))  
 {  
 print(unlist(r))  
 break  
 }  
}

# Nonperfect accuracy

Jetzt verallgemeinere ich auf non-perfect accuracy. Wie bereits bekannt, funktioniert es nicht, also müssten wir auch ein Gegenbeispiel bekommen.

# Stochastic dominance for P(D < t & C = 1)  
sdom = function(a, A)  
{  
 if(a$C == 1 & A$C == 1)  
 return(a$D > A$D)  
   
 if(a$C == 1 & A$C != 1)  
 return(FALSE)  
   
 if(a$C != 1 & A$C == 1)  
 return(TRUE)  
   
 if(a$C != 1 & A$C != 1)  
 return(a$D < A$D)  
}  
  
sim = function()  
{  
 a = list(D=sample(1:10, size=1), C=sample(1:2, size=1))  
 A = list(D=sample(1:10, size=1), C=sample(1:2, size=1))  
   
 b = list(D=sample(1:10, size=1) + 0.1, C=sample(1:2, size=1))  
 B = list(D=sample(1:10, size=1) + 0.1, C=sample(1:2, size=1))  
  
 # Stochastic dominance satisfied?  
 if(sdom(a, A) & sdom(b, B))  
 return(list(a=a, A=A, b=b, B=B))  
   
 # Otherwise, retry  
 sim()  
}  
  
# Example  
unlist(sim())

## a.D a.C A.D A.C b.D b.C B.D B.C   
## 9.0 2.0 6.0 1.0 4.1 2.0 5.1 2.0

Die Funktionen race und resp bleiben unverändert. Für den IC werden falsche Reaktionen auf Inf gesetzt [dann ist ecdf(D < t & C = 1) = 0].

# Interaction contrast  
ic = function(ab, aB, Ab, AB, t=1:10)  
{  
 if(ab$C != 1)  
 ab$D = Inf  
   
 if(aB$C != 1)  
 aB$D = Inf  
   
 if(Ab$C != 1)  
 Ab$D = Inf  
   
 if(AB$C != 1)  
 AB$D = Inf  
  
 Fab = ecdf(ab$D)(t)  
 FaB = ecdf(aB$D)(t)  
 FAb = ecdf(Ab$D)(t)  
 FAB = ecdf(AB$D)(t)  
   
 all(Fab + FAB <= FaB + FAb)  
}  
  
## 1000 times, interrupt if IC test failed  
for(i in 1:1000)  
{  
 d = sim()  
 r = resp(d)  
 if(!ic(r$ab, r$aB, r$Ab, r$AB))  
 {  
 print('Counter example')  
 print(unlist(d))  
 print(unlist(r))  
 break  
 }  
}

## [1] "Counter example"  
## a.D a.C A.D A.C b.D b.C B.D B.C   
## 3.0 2.0 6.0 1.0 2.1 2.0 8.1 2.0   
## ab.D ab.C aB.D aB.C Ab.D Ab.C AB.D AB.C   
## 2.1 2.0 3.0 2.0 2.1 2.0 6.0 1.0

Wir bekommen ein Gegenbeispiel, in dem stochastische Dominanz erfüllt ist, weil

* .

Der IC ist aber positiv,

# Hybrid aus Miller und Townsend

Für Millers Spezialfall und funktioniert die Wettlaufungleichung, auch die Verallgemeinerung für incorrect responses. Für Townsends allgemeineres Design und funktioniert der IC für incorrect responses nicht.

Gibt es was dazwischen? Man könnte sich folgenden Hybrid anschauen, und , und prüfen, ob

eingehalten wird. Es ist eigentlich klar, dass das nicht funktionieren kann, denn das Problem ist symmetrisch und wir ändern ja nur , aber egal, vielleicht erfahren wir etwas Neues. Für diesen Spezialfall muss nur in der Simulation geändert werden.

sim = function()  
{  
 a = list(D=Inf, C=sample(1:2, size=1))  
 A = list(D=sample(1:10, size=1), C=sample(1:2, size=1))  
   
 b = list(D=sample(1:10, size=1) + 0.1, C=sample(1:2, size=1))  
 B = list(D=sample(1:10, size=1) + 0.1, C=sample(1:2, size=1))  
  
 # Stochastic dominance satisfied?  
 if(sdom(a, A) & sdom(b, B))  
 return(list(a=a, A=A, b=b, B=B))  
   
 # Otherwise, retry  
 sim()  
}  
  
## 1000 times, interrupt if IC test failed  
for(i in 1:1000)  
{  
 d = sim()  
 r = resp(d)  
 if(!ic(r$ab, r$aB, r$Ab, r$AB))  
 {  
 print('Counter example')  
 print(unlist(d))  
 print(unlist(r))  
 break  
 }  
}

## [1] "Counter example"  
## a.D a.C A.D A.C b.D b.C B.D B.C   
## Inf 1.0 10.0 1.0 6.1 2.0 10.1 2.0   
## ab.D ab.C aB.D aB.C Ab.D Ab.C AB.D AB.C   
## 6.1 2.0 10.1 2.0 6.1 2.0 10.0 1.0

# SOA statt Salienz manipulieren

Wir scheitern ja immer an der stochastischen Dominanz, die wird in unserem Szenario ja für definiert, wohingegen sich das Wettlaufmodell nur für die Processing time interessiert und eben nicht für . Eine Möglichkeit, das Problem zu entschärfen, ist, statt der Salienz einer Reizkomponente (schwach, stark) den Onset zu manipulieren (früh, spät, mit SOA ). Dadurch wird die Processing time einfach um nach hinten verschoben, und bleibt gleich.

Außerdem ignorieren wir bei der Prüfung des IC. Wenn man genauer darüber nachdenkt, wird ein solches Experiment dadurch komplett uninteressant, denn wenn man ignoriert, kann man Millers oder Townsends Ungleichung immer herleiten (das ist der Punkt, auf den wir uns noch nicht geeinigt haben).

sim = function(tau = 2.5)  
{  
 A = list(D=sample(1:10, size=1), C=sample(1:2, size=1))  
 a = list(D=tau + A$D, C=A$C)  
   
 B = list(D=sample(1:10, size=1) + 0.1, C=sample(1:2, size=1))  
 b = list(D=tau + B$D, C=B$C)  
  
 # Stochastic dominance satisfied?  
 if(sdom(a, A) & sdom(b, B))  
 return(list(a=a, A=A, b=b, B=B))  
   
 # Otherwise, retry  
 sim()  
}  
  
# Interaction contrast  
ic = function(ab, aB, Ab, AB, t=1:10)  
{  
 if(ab$C != 1)  
 ab$D = Inf  
   
 if(aB$C != 1)  
 aB$D = Inf  
   
 if(Ab$C != 1)  
 Ab$D = Inf  
   
 if(AB$C != 1)  
 AB$D = Inf  
  
 Fab = ecdf(ab$D)(t)  
 FaB = ecdf(aB$D)(t)  
 FAb = ecdf(Ab$D)(t)  
 FAB = ecdf(AB$D)(t)  
   
 all(Fab + FAB <= FaB + FAb)  
}  
  
## 1000 times, interrupt if IC test failed  
for(i in 1:1000)  
{  
 d = sim()  
 r = resp(d)  
 if(!ic(r$ab, r$aB, r$Ab, r$AB))  
 {  
 print('Counter example')  
 print(unlist(d))  
 print(unlist(r))  
 break  
 }  
}

(kein Gegenbeispiel)

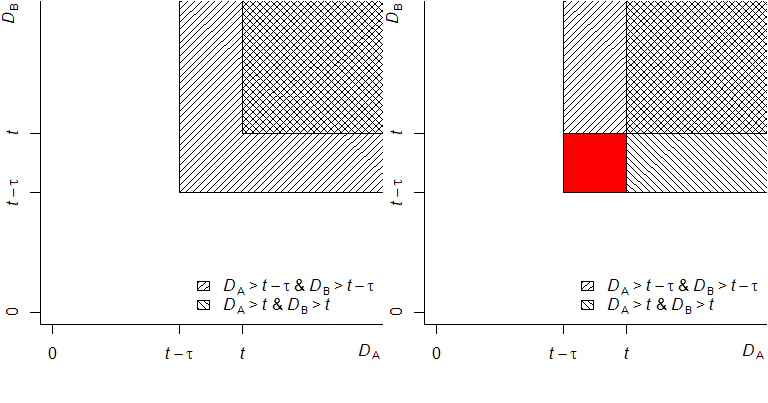
Vermutung: Funktioniert, weil stochastische Dominanz für ja eingehalten wird. Sehr schön. Das ist sozusagen “neu”.

# Hybrid aus SOA-Manipulation und Miller

(hat sich erledigt, ist ja nur ein Spezialfall des SOA interaction contrasts)

# Abbildung

Zunächst mal den klassischen SIC. Man erkennt, dass das Rechteck im Subtrahend “fehlt”; deswegen ist der SIC positiv, der DIC dann negativ.



Diese Darstellung geht von einer leicht veränderten selective influence-Annahme aus, wodurch sich der Beweis auch etwas unterscheidet: Wir gehen von einer bivariaten Verteilung aus, die in allen Bedingungen (AB, ab, Ab, aB) gleich ist und durch das SOA nur um “verschoben” wird. Für Bedingung (also A zu , B zu ) ist die bivariate Verteilung dann einfach

Ich hoffe, die Notation ergibt irgendwie Sinn. Wir bekommen für

Und für die Wahrscheinlichkeiten dann wie gehabt

Das sind die Rechtecke von oben. Jetzt wird nicht ausmultipliziert, denn Unabhängigkeit von und wird ga nicht benötigt, sondern direkt die Doppeldifferenz angewendet. Was übrigbleibt, ist das rote Rechteck.

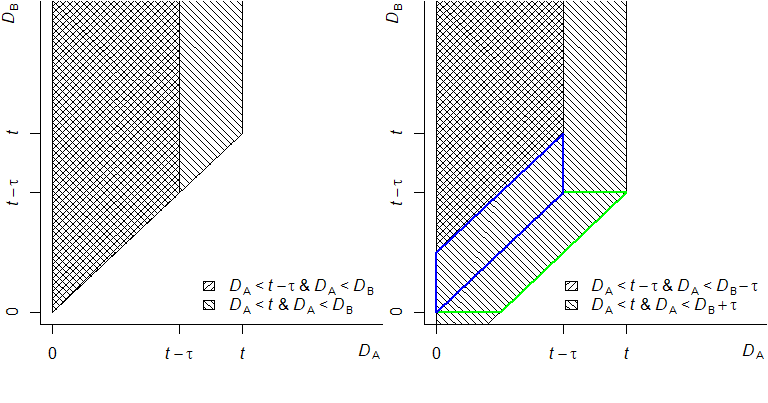
## Herleitung per Fallunterscheidung

Es ist ja bisher nicht gelungen, den interaction contrast per Fallunterscheidung herzuleiten, nicht einmal für accuracy 100%. Vielleicht hilft eine Abbildung. Wir haben:

Die Wahrscheinlichkeit für eine schnelle Reaktion ist dann:

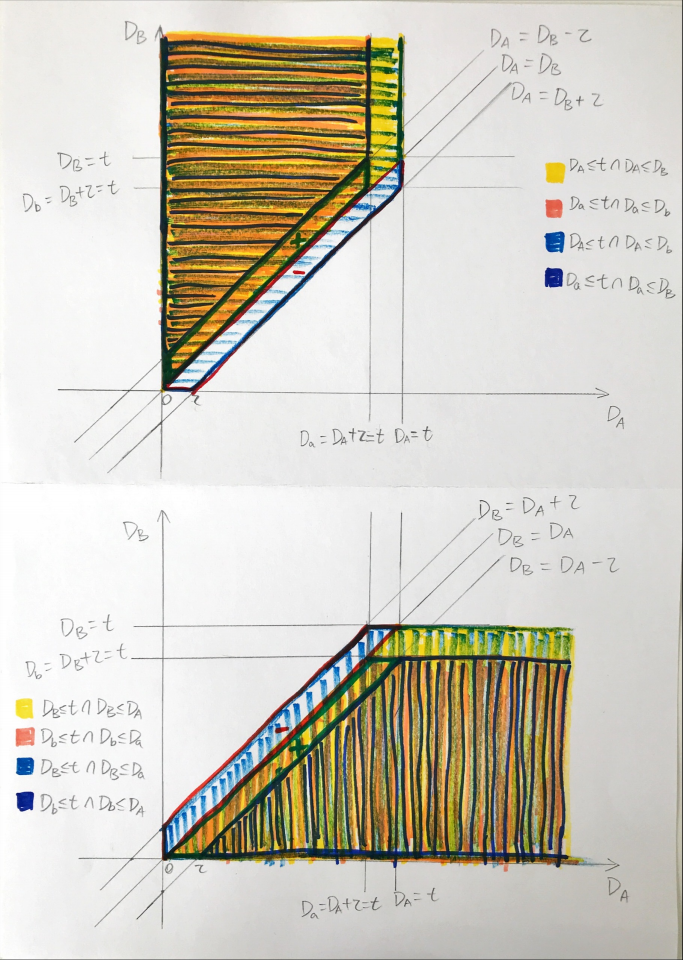
Für den Interaktionskontrast haben wir

Kann man die grafisch darstellen? Die haben die Form eines “Teppichmessers”.



Weil hier der distribution interaction contrast betrachtet wird und nicht der SIC, müssen die Flächen im rechten Bild diejenigen im linken Bild einschließen. Man sieht aber, dass das blaue Parallelogramm nicht doppelt belegt ist. So einfach ist es also nicht, man muss und vermutlich gemeinsam betrachten. Das sieht aber vielversprechend aus, denn das Parallelogramm kann an der Winkelhalbierenden gespiegelt werden (grün eingezeichnet) und verdeckt dann die entsprechende Fläche in und vice-versa.

* Es müsste also möglich sein, den SOA interaction contrast für 100% accuracy auch herzuleiten, indem man nicht annimmt, sondern die Fallunterscheidung.
* Und wenn das funktioniert, funktioniert vielleicht auch die Herleitung des SOA interaction contrasts für non-perfect accuracy.



Das obere Bild zeigt das Ergebnis im Fall 1 (entspricht P\_L). Das Integral über dem Gebiet mit dem roten Rand (Trapez) hat ein Minuszeichen; das über dem Gebiet mit dem grünen Rand (Parallelogramm) hat ein Pluszeichen.

Das obere Bild zeigt das Ergebnis im Fall 2 (entspricht P\_R). Analog.

Das Trapeze im oberen Bild ist um ein Dreieck größer als das Parallelogramm im unteren Bild. Das Trapeze im unteren Bild ist um ein Dreieck größer als das Parallelogramm im oberen Bild. Die zwei Dreiecke zusammen machen das Quadrat aus, dessen Ränder D\_A=t, D\_A=t-tau, D\_B=t, D\_B=t-tau sind.