DEMI ALGORITHM

Total Effect Calculation

Step 2 of the DEMI Pipeline — Odds Ratio Estimation & Edge Cases

Step 1	**Step 2**	Step 3	Step 4
Temporal Ordering	**Total Effect**	Direct Effects	Multiply & Predict

This video covers Step 2 only

2 The 2×2 Table: Reading from the Knowledgebase

Contingency table

	$Y = 1$	$Y = 0$
$Z_k = 1$	**a**	**b**
$Z_k = 0$	**c**	**d**

$a = C(Z_k=1, Y=1)$ $b = C(Z_k=1, Y=0)$

$c = C(Z_k=0, Y=1)$ $d = C(_Zk=0, Y=0)$

2 The 2×2 Table: Reading from the Knowledgebase

Contingency table

	$Y = 1$	$Y = 0$
$Z_k = 1$	**a**	**b**
$Z_k = 0$	**c**	**d**

$a = C(Z_k=1, Y=1)$ $b = C(Z_k=1, Y=0)$

$c = C(Z_k=0, Y=1)$ $d = C(Zk=0, Y=0)$

Raw odds ratio formula

$$OR_{raw} = a \cdot d / b \cdot c$$

$$T_{kY} = \log(OR)$$

3 Normal Case: Large Samples, No Zero Cells

Worked example: Hypertension → Sertraline response

	$Y = 1$	$Y = 0$
$Z_k = 1$	**a = 420**	**b = 210**
$Z_k = 0$	**c = 580**	**d = 1,140**

a+b = 630 ≥ 30 ✓ **a+c = 1,000 ≥ 30 ✓** **No zero cells ✓**

Step-by-step calculation

$OR = a \cdot d / b \cdot c$

$= 420 \times 1{,}140 / 210 \times 580$

$= 478{,}800 / 121{,}800$

= 3.93

$T_{kY} = \log(3.93) =$ **1.37**

4 Edge Case 1: Cell a = 0

Feature never co-occurs with the outcome. Rule applies regardless of sample size.

$a = 0,\ b > 0,\ c > 0$

0	b
c	d

$$OR = 1 / \min(b, c)$$

4 Edge Case 1: Cell a = 0

Feature never co-occurs with the outcome. Rule applies regardless of sample size.

a = 0, b > 0, c > 0

0	b
c	d

`OR = 1 / min(b, c)`

a = 0, b = 0

0	0
c	d

`OR = 1 / (c + 1)`

4 Edge Case 1: Cell a = 0

Feature never co-occurs with the outcome. Rule applies regardless of sample size.

a = 0, b > 0, c > 0

0	b
c	d

`OR = 1 / min(b, c)`

a = 0, b = 0

0	0
c	d

`OR = 1 / (c + 1)`

a = 0, c = 0 ⚠

0	b
0	d

→ **EXCLUDE ROW**

5 Edge Case 2: b = 0 or c = 0 (Large Samples)

a > 0, but b or c is zero and both samples are large (≥ 30).

b = 0, c > 0 (large)

a	0
c	d

```
OR = (a+c) / (b+1)
   = (a+c) / 1
```

5 Edge Case 2: b = 0 or c = 0 (Large Samples)

a > 0, but b or c is zero and both samples are large (≥ 30).

b = 0, c > 0 (large)

a	0
c	d

```
OR = (a+c) / (b+1)
   = (a+c) / 1
```

b > 0, c = 0 (large)

a	b
0	d

```
OR = (a+b) / (c+1)
   = (a+b) / 1
```

5 Edge Case 2: b = 0 or c = 0 (Large Samples)

a > 0, but b or c is zero and both samples are large (≥ 30).

b = 0, c > 0 (large)

a	0
c	d

```
OR = (a+c) / (b+1)
   = (a+c) / 1
```

b > 0, c = 0 (large)

a	b
0	d

```
OR = (a+b) / (c+1)
   = (a+b) / 1
```

b = 0 AND c = 0 (large)

a	0
0	d

```
OR = a + 1
```

6 Edge Case 3: Small Sample Instability

The problem: d inflates the raw OR

	Y = 1	Y = 0
$Z_k = 1$	**a = 3**	**b = 7**
$Z_k = 0$	**c = 5**	**d = 50,000**

a+b = 10 < 30 (small!)

Raw OR = 3×50,000 / 35 = 4,286 ← Inflated!

Two d* estimates from binomial samples

Exposed sample: a/(a+b) = b/(b+d*)

d1* = b(a+b)/a = 7×10/3 = 23.3

OR1 = (a+b)/c = 10/5 = 2.0

6 Edge Case 3: Small Sample Instability

The problem: d inflates the raw OR

	Y = 1	Y = 0
$Z_k = 1$	**a = 3**	**b = 7**
$Z_k = 0$	**c = 5**	**d = 50,000**

a+b = 10 < 30 (small!)

Raw OR = 3×50,000 / 35 = 4,286 ← Inflated!

Two d* estimates from binomial samples

Exposed sample: a/(a+b) = b/(b+d*)

d1* = b(a+b)/a = 7×10/3 = 23.3

OR1 = (a+b)/c = 10/5 = **2.0**

Case sample: a/(a+c) = c/(c+d*)

d2* = c(a+c)/a = 5×8/3 = 13.3

OR2 = (a+c)/b = 8/7 = 1.14

6 Edge Case 3: Small Sample Instability

The problem: d inflates the raw OR

	Y = 1	Y = 0
$Z_k = 1$	**a = 3**	**b = 7**
$Z_k = 0$	**c = 5**	**d = 50,000**

a+b = 10 < 30 (small!)

Raw OR = 3×50,000 / 35 = 4,286 ← Inflated!

Two d* estimates from binomial samples

Exposed sample: a/(a+b) = b/(b+d*)
d1* = b(a+b)/a = 7×10/3 = 23.3
OR1 = (a+b)/c = 10/5 = **2.0**

Case sample: a/(a+c) = c/(c+d*)
d2* = c(a+c)/a = 5×8/3 = 13.3
OR2 = (a+c)/b = 8/7 = 1.14

Raw OR ≥ 1 → use max(OR1, OR2) = 2.0 (vs. 4,286)

7 Edge Case 4: Small Samples + Zero Cells

Zero-cell rules always take priority over the sample-size path. Check for zeros first, then check sample size.

Small + a = 0

```
OR = 1 / min(b, c)
```

Same as large-sample rule. a=0 overrides the sample-size path in all cases.

Small + b = 0, c > 0

```
OR = (a+c) / (b+1)
```

Laplace +1 still applied. Sample size does not change zero-marginal handling.

Small + b > 0, c = 0

```
OR = (a+b) / (c+1)
```

Same logic as large sample. Direction and sign are preserved.

Small + b = 0 AND c = 0

```
OR = a + 1
```

Perfect association in sparse data. Consistent with the large-sample rule.

8 Putting It Together: Full Decision Tree

a=0, c=0
→ Exclude row entirely

a=0, b=0
→ OR = 1 / (c+1)

a=0, b>0
→ OR = 1 / min(b, c)

b=0, c=0
→ OR = a + 1

b=0, c>0
→ OR = (a+c) / (b+1)

b>0, c=0
→ OR = (a+b) / (c+1)

All > 0, large
→ OR = a times d / (b times c)

All > 0, small
→ OR = max or min of (a+b)/c & (a+c)/b

8 Putting It Together: Full Decision Tree

`a=0, c=0`

→ Exclude row entirely

a=0, b=0

→ OR = 1 / (c+1)

`a=0, b>0`

→ OR = 1 / min(b, c)

`b=0, c=0`

→ OR = a + 1

`b=0, c>0`

→ OR = (a+c) / (b+1)

`b>0, c=0`

→ OR = (a+b) / (c+1)

`All > 0, large`

→ OR = a times d / (b times c)

`All > 0, small`

→ OR = max or min of (a+b)/c & (a+c)/b

8 Putting It Together: Full Decision Tree

`a=0, c=0`

→ Exclude row entirely

`a=0, b=0`

→ OR = 1 / (c+1)

`a=0, b>0`

→ OR = 1 / min(b, c)

`b=0, c=0`

→ OR = a + 1

`b=0, c>0`

→ OR = (a+c) / (b+1)

`b>0, c=0`

→ OR = (a+b) / (c+1)

`All > 0, large`

→ OR = a times d / (b times c)

`All > 0, small`

→ OR = max or min of (a+b)/c & (a+c)/b

8 Putting It Together: Full Decision Tree

a=0, c=0

→ Exclude row entirely

a=0, b=0

→ OR = 1 / (c+1)

a=0, b>0

→ OR = 1 / min(b, c)

b=0, c=0

→ **OR = a + 1**

b=0, c>0

→ OR = (a+c) / (b+1)

b>0, c=0

→ OR = (a+b) / (c+1)

All > 0, large

→ OR = a times d / (b times c)

All > 0, small

→ OR = max or min of (a+b)/c & (a+c)/b

8 Putting It Together: Full Decision Tree

`a=0, c=0`
→ Exclude row entirely

`a=0, b=0`
→ OR = 1 / (c+1)

`a=0, b>0`
→ OR = 1 / min(b, c)

`b=0, c=0`
→ OR = a + 1

`b=0, c>0`
→ OR = (a+c) / (b+1)

`b>0, c=0`
→ OR = (a+b) / (c+1)

`All > 0, large`
→ OR = a times d / (b times c)

`All > 0, small`
→ OR = max or min of (a+b)/c & (a+c)/b

8 Putting It Together: Full Decision Tree

a=0, c=0
→ Exclude row entirely

a=0, b=0
→ OR = 1 / (c+1)

a=0, b>0
→ OR = 1 / min(b, c)

b=0, c=0
→ OR = a + 1

b=0, c>0
→ OR = (a+c) / (b+1)

b>0, c=0
→ OR = (a+b) / (c+1)

All > 0, large
→ OR = a times d / (b times c)

All > 0, small
→ OR = max or min of (a+b)/c & (a+c)/b

8 Putting It Together: Full Decision Tree

`a=0, c=0`

→ Exclude row entirely

`a=0, b=0`

→ OR = 1 / (c+1)

`a=0, b>0`

→ OR = 1 / min(b, c)

`b=0, c=0`

→ OR = a + 1

`b=0, c>0`

→ OR = (a+c) / (b+1)

`b>0, c=0`

→ OR = (a+b) / (c+1)

All > 0, large

→ OR = a d / (b c)

`All > 0, small`

→ OR = max or min of (a+b)/c & (a+c)/b

8 Putting It Together: Full Decision Tree

a=0, c=0

→ Exclude row entirely

a=0, b=0

→ OR = 1 / (c+1)

a=0, b>0

→ OR = 1 / min(b, c)

b=0, c=0

→ OR = a + 1

b=0, c>0

→ OR = (a+c) / (b+1)

b>0, c=0

→ OR = (a+b) / (c+1)

All > 0, large

→ OR = a times d / (b times c)

All > 0, small

→ **OR = max/min [(a+b)/c, (a+c)/b]**