NMBU

NMBU

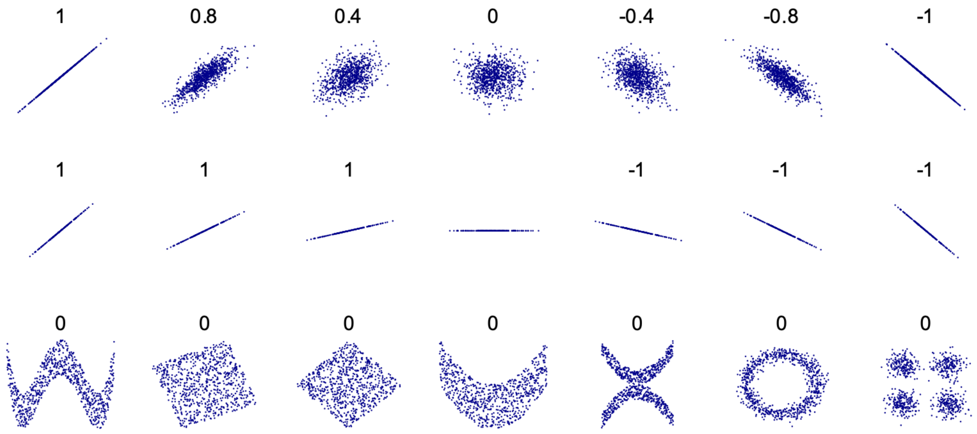
Norwegian University
of Life Sciences

Partial and multiblock matrix correlations

Kristian Hovde Liland

Correlation

- What is it?
  - Broadly: Any dependence or association between two variables
  - Typically: Degree of linear relation between two random variables



$$r(a\mathbf{X}_1, \mathbf{X}_2) = r(\mathbf{X}_1, b\mathbf{X}_2) = r(\mathbf{X}_1, \mathbf{X}_2)$$

$$r(\mathbf{X}_1, \mathbf{X}_2) = r(\mathbf{X}_2, \mathbf{X}_1)$$

$$r(\mathbf{X}_1, \mathbf{X}_2) = 1 \textit{ if } \mathbf{X}_1 = b\mathbf{X}_2$$

$$r(\mathbf{X}_1, \mathbf{X}_2) = 0 \textit{ iff } \mathbf{X}_1^t \mathbf{W} \mathbf{X}_2 = 0$$

Matrix correlation

- Generalisation of correlation to pairs of matrices
- Even more
  - possible definitions,
  - aspects to emphasise,
  - restrictions to apply
- Simplest variation?: Inner product correlation

A1	A2
0.4	1.2
0.3	2.1
0.6	2.9
0.4	2.5

B1	B2
0.5	1.5
0.2	1.9
0.5	2.3
0.3	1.3

$$R_{in}(\mathbf{X}_1, \mathbf{X}_2) = tr(\mathbf{X}_1^t \mathbf{X}_2) / \sqrt{tr(\mathbf{X}_1^t \mathbf{X}_1) tr(\mathbf{X}_2^t \mathbf{X}_2)}$$

Matrix correlation – some history

- 1965 – Rozeboom's squared vector correlation
- 1974 – Coxhead's coefficient
- 1974 – Yannai's Generalised Coefficient of Determination (GCD)
- 1976 – RV
- 1978 – Procrustes Similarity Index (PSI)
- 1984 – Ramsay's r1, r2, r3, r4
- 2009 – RV2
- 2011 – $RV_{adjusted}$ (Maye)
- 2015 – $RV_{adjusted}$ (Ghaziri)
- 2018 – Similarity of Matrices Index (SMI)

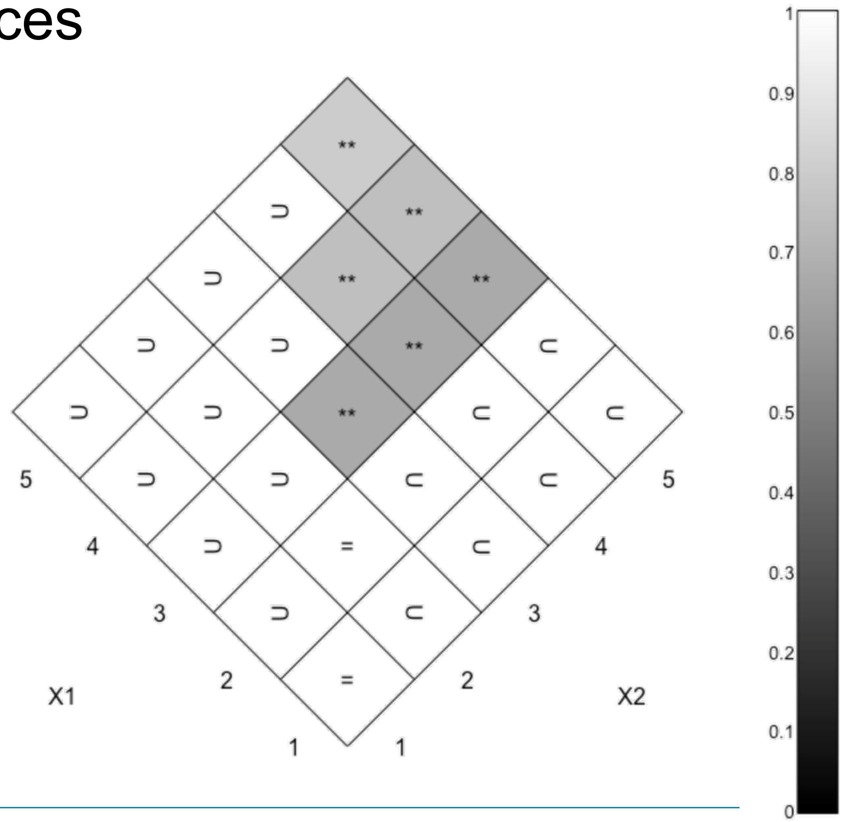
All these measures are included in the R package *MatrixCorrelation*, maintained by K.H. Liland.

Several are described in the Multiblock Data Fusion book by Smilde et al.

Similarity of matrices indexes – SMI

- Similarity between unscaled subspaces
- Compute left singular vectors (**T**, **U**) for two centred matrices
- Explained variance of smallest subspace
  when predicted from largest

$$SMI_{OP}(\mathbf{T},\mathbf{U})=max\left(\frac{\|\mathbf{B_T}\|_F^2}{p},\frac{\|\mathbf{B_U}\|_F^2}{q}\right)=\frac{\|\mathbf{T}^t\mathbf{U}\|_F^2}{r}$$



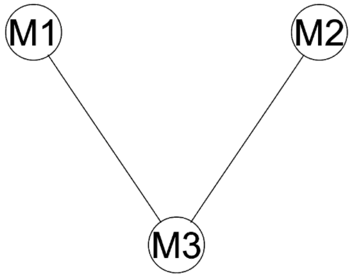
- Close relation to GCD, restricted number of comp.
- Can restrict regression to Procrustes rotations

Partial correlation

- Correlation between two random variables conditioned on a third variable
  corr(**a**, **b** | **c**)

- Can be calculated as correlation between residuals from regressions
  $\boldsymbol{a} \sim \boldsymbol{c} \Longrightarrow \boldsymbol{a}^{\perp}$ and $\boldsymbol{b} \sim \boldsymbol{c} \Longrightarrow \boldsymbol{b}^{\perp}$
  corr($\boldsymbol{a}^{\perp}$, $\boldsymbol{b}^{\perp}$)

- Useful on its own and in estimating connections between variables in a network



Partial matrix correlation

- RV, RV2*, GCD and SMI[♖] can all be formulated as correlations between vectorised configuration matrices:

$$RV = \frac{tr(\mathbf{S}_1\mathbf{S}_2)}{\sqrt{tr(\mathbf{S}_1)tr(\mathbf{S}_2)}} = cor(vec(\mathbf{S}_1), vec(\mathbf{S}_2))$$

$$RV2 = \frac{tr(\tilde{\mathbf{S}}_1\tilde{\mathbf{S}}_2)}{\sqrt{tr(\tilde{\mathbf{S}}_1)tr(\tilde{\mathbf{S}}_2)}} \approx cor(vec(\tilde{\mathbf{S}}_1), vec(\tilde{\mathbf{S}}_2))$$

$$SMI = cor(vec(\mathbf{C}_1), vec(\mathbf{C}_2))/\min(p,q) \cdot \sqrt{p \cdot q}$$

- … which give simple translations to partial correlation

$$pRV = cor(vec(\mathbf{S}_1), vec(\mathbf{S}_2)|vec(\mathbf{S}_3))$$

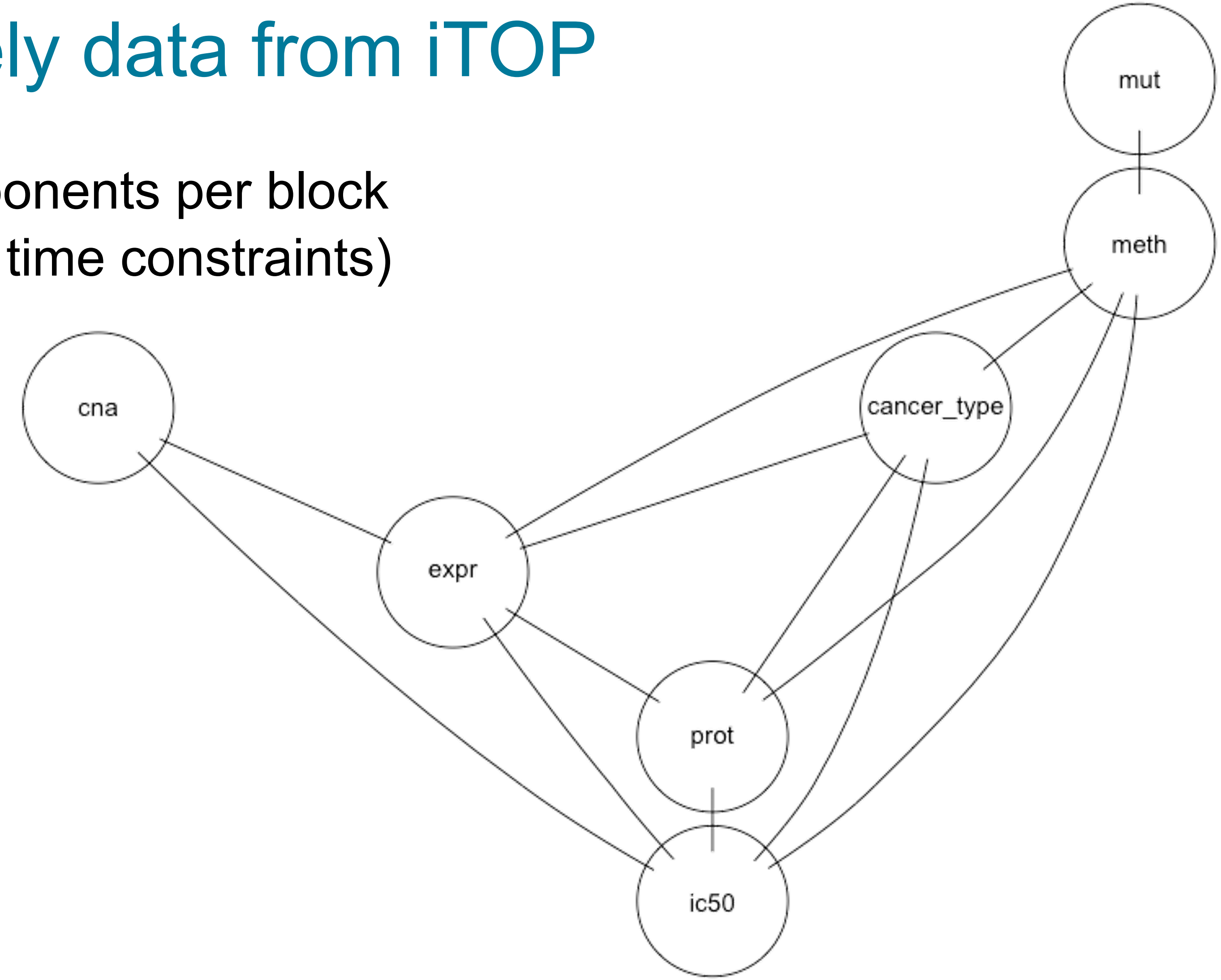
$$pRV2 = cor(vec(\tilde{\mathbf{S}}_1), vec(\tilde{\mathbf{S}}_2)|vec(\tilde{\mathbf{S}}_3))$$

$$pSMI = cor(vec(\mathbf{C}_1), vec(\mathbf{C}_2)|vec(\mathbf{C}_3))$$

* non-centred configuration matrices
♖ needs rescaling to be exact

Approximately data from iTOP

- SMI with 5 components per block (cheating due to time constraints)



Multiblock matrix correlation

- Basic idea:
  - Reuse configuration matrix formulations
  - Exchange correlation with Generalised Correlation Analysis (GCA)

$$mRV = GCA(vec(\mathbf{S}_1), vec(\mathbf{S}_2), vec(\mathbf{S}_3), ...)$$

$$mRV2 \approx GCA(vec(\tilde{\mathbf{S}}_1), vec(\tilde{\mathbf{S}}_2), vec(\tilde{\mathbf{S}}_3), ...)$$

$$mSMI = GCA(vec(\mathbf{C}_1), vec(\mathbf{C}_2), vec(\mathbf{C}_3), ...)$$

Statistical test for common component

- Permutation test:
  - H_0: blocks originate from same source

- P-value for common components in PCA-GCA

Dimensionality of single blocks

```python
#%% Projection with rank reduction
def Proj(X, rrank='none', prop=0.95):
    """Projection matrix

    Compute X inv(X'X) X' with support for singular X solved
    by Moore-Penrose pseudo inverse of (X'X) or PCA of X using
    components up to a chosen cummulative explained variance.

    Parameters
    ----------
    X : numpy array
        Data to create projection matrix from.
    rrank : str
        Type of rank reduction. Either 'none' (default), 'pinv' for
        Moore-Penrose pseudo-inverse or 'PCA' (see next argument).
    prop : float
        Proportion of explained variance in PCA rank reduction
        (see previous argument).
    """
    if rrank == 'none':
        return X @ npl.inv(X.T @ X) @ X.T
    elif rrank == 'pinv':
        return X @ npl.pinv(X.T @ X) @ X.T
    elif rrank == 'PCA':
        X = X - np.mean(X, axis=0)                     # Centre
        U,D,_ = npl.svd(X, full_matrices=False)        # PCA (by SVD)
        expl = np.cumsum(D**2/np.sum(D**2))            # Explained variance
        comp = np.where(np.array(expl) >= prop)[0][0]  # First comp > prop
        return U[:,:comp] @ U[:,:comp].T
```

Wines – A -> E

Single path

Direct and indirect effects between two blocks

```
tab, effs, orth_inds = pe.path_effects(blocks, paths, 'A', 'E', rrank='PCA', continue_from_prediction=True)
tab
```

✓ 0.1s

	effect	explained variance
0	direct: A -> E	1.532492
1	indirect: A -> B -> C -> D -> E	0.000000
2	indirect: A -> B -> C -> E	0.000000
3	indirect: A -> B -> D -> E	0.000000
4	indirect: A -> B -> E	0.000000
5	indirect: A -> C -> D -> E	1.339977
6	indirect: A -> C -> E	0.140002
7	indirect: A -> D -> E	0.046028
8	sum of effects	3.058498
9	uncorrected total: A -> E	49.894067

Wines – all paths ending at E

All effects

All direct and indirect effects ending at the specified block.

```
tab = pe.all_effects(blocks, paths, 'E', rrank='PCA', continue_from_prediction=True)
tab
```

	effect	explained variance
0	direct: A -> E	1.532492
1	indirect: A -> B -> C -> D -> E	0.000000
2	indirect: A -> B -> C -> E	0.000000
3	indirect: A -> B -> D -> E	0.000000
4	indirect: A -> B -> E	0.000000
5	indirect: A -> C -> D -> E	1.339977
6	indirect: A -> C -> E	0.140002
7	indirect: A -> D -> E	0.046028
8	direct: B -> E	0.566825
9	indirect: B -> C -> D -> E	0.000000
10	indirect: B -> C -> E	0.000000
11	indirect: B -> D -> E	0.000000
12	direct: C -> E	6.132573
13	indirect: C -> D -> E	3.620878
14	direct: D -> E	7.952575
15	sum of effects	21.331349

Mobile – A -> E and C -> E

Single path

Direct and indirect effects between two blocks

```
1  tab, effs, orth_inds = pe.path_effects(blocks, paths, 'A', '[
2  tab
```

✓ 0.9s

	effect	explained variance
0	direct: A -> E	1.866591
1	indirect: A -> B -> C -> D -> E	0.000000
2	indirect: A -> B -> C -> E	0.000000
3	indirect: A -> B -> D -> E	0.000000
4	indirect: A -> B -> E	0.192438
5	sum of effects	2.059028
6	uncorrected total: A -> E	34.382584

Single path

Direct and indirect effects between two blocks

```
1  tab, effs, orth_inds = pe.path_effects(blocks, paths, 'C', 'E
2  tab
```

✓ 0.8s

	effect	explained variance
0	direct: C -> E	9.833953
1	indirect: C -> D -> E	1.913509
2	sum of effects	11.747462
3	uncorrected total: C -> E	46.564378

Mobile – all paths ending at E

All effects

All direct and indirect effects ending at the specified block.

```
tab = pe.all_effects(blocks, paths, 'E', rrank='PCA', continue_from_prediction=True)
tab
```

✓ 0.3s

	effect	explained variance
0	direct: A -> E	1.866591
1	indirect: A -> B -> C -> D -> E	0.000000
2	indirect: A -> B -> C -> E	0.000000
3	indirect: A -> B -> D -> E	0.000000
4	indirect: A -> B -> E	0.192438
5	direct: B -> E	0.767219
6	indirect: B -> C -> D -> E	0.000000
7	indirect: B -> C -> E	0.000000
8	indirect: B -> D -> E	0.000000
9	direct: C -> E	9.833953
10	indirect: C -> D -> E	1.913509
11	direct: D -> E	1.853860
12	sum of effects	16.427570

Simulation

Orthogonal info in middle block

[A] -> [Ba Bc] -> [C], i.e., local effect from A to Ba, but neither direct or indirect effect from A to C

```python
# Random data
X = np.random.normal(0,1,[10,5])

# PCA
X = X - np.mean(X, axis=0)
u,s,v = npl.svd(X,False)

# Orthogonal block variables relating to A and C
Ba = u[:,:-1:2]
Bc = u[:,1::2]
Bs = np.hstack([Ba, Bc])

# Create A and C as linear combinations of orthogonal block variables plus a little noise
As = Ba @ np.array([[1,2],[2.3,-1.6]]) + np.random.normal(0,0.001,[10,2])    # A made from Ba
Cs = Bc @ np.array([[1.4,3],[-1.1,1.3]]) + np.random.normal(0,0.001,[10,2])  # C made from Bc

# Blocks and paths
blocks_sim = {'A': As, 'B': Bs, 'C': Cs}
paths_sim = np.array([[0,1],[0,2],[1,2]])
```

Single path

Direct and indirect paths from A to C

```
# A->B and Bhat->C
tab, effs, orth_inds = pe.path_effects(blocks_sim, paths_sim, 'A', 'C', rrank='none', prop=0.99, continue_from_prediction=True)
print(tab)
print(effs)
```

✓ 0.8s

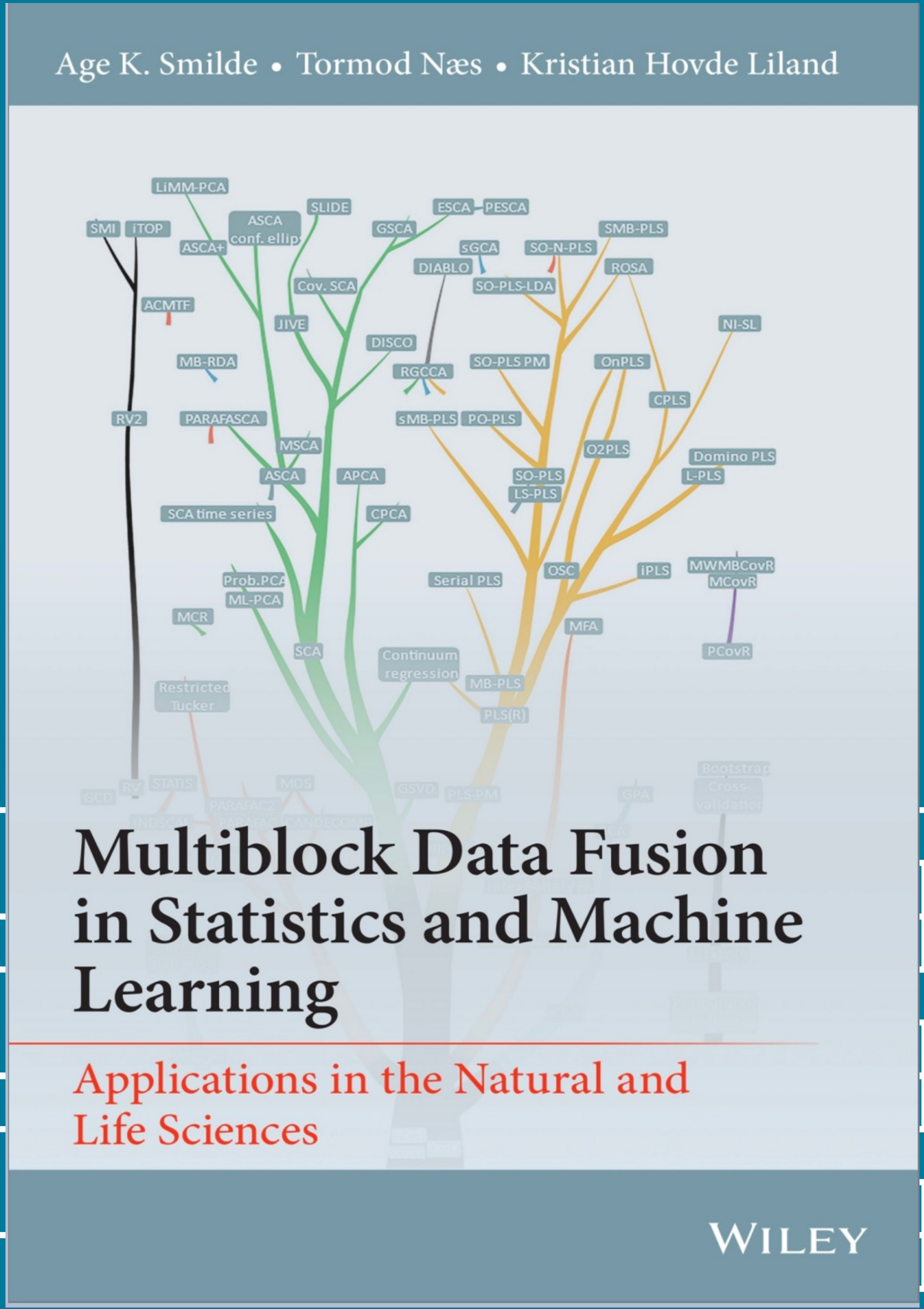
```
                     effect  explained variance
0           direct: A -> C            0.000050
1      indirect: A -> B -> C          0.000031
2            sum of effects           0.000080
3  uncorrected total: A -> C          0.000052
[[4.97051740668818e-05], [49.999909693296914, 6.138974705392872e-05], [8.039999215478956e-05], [5.1991218477012093e-05]]
```

```
# A->B and B->C
tab, effs, orth_inds = pe.path_effects(blocks_sim, paths_sim, 'A', 'C', rrank='none', prop=0.99, continue_from_prediction=False)
print(tab)
print(effs)
```

✓ 0.2s

```
                     effect  explained variance
0           direct: A -> C            0.000050
1      indirect: A -> B -> C         49.999864
2            sum of effects          49.999914
3  uncorrected total: A -> C          0.000052
[[4.97051740668818e-05], [49.999909693296914, 99.99990934548399], [49.99991407129484], [5.1991218477012093e-05]]
```

Buy the book! 😉


Age K. Smilde • Tormod Næs • Kristian Hovde Liland
Multiblock Data Fusion in Statistics and Machine Learning
Applications in the Natural and Life Sciences
WILEY