Basic Fundamentals of Structural Equation Modeling (part 2)

Jim Grace

U.S. Geological Survey

Outline (continued)

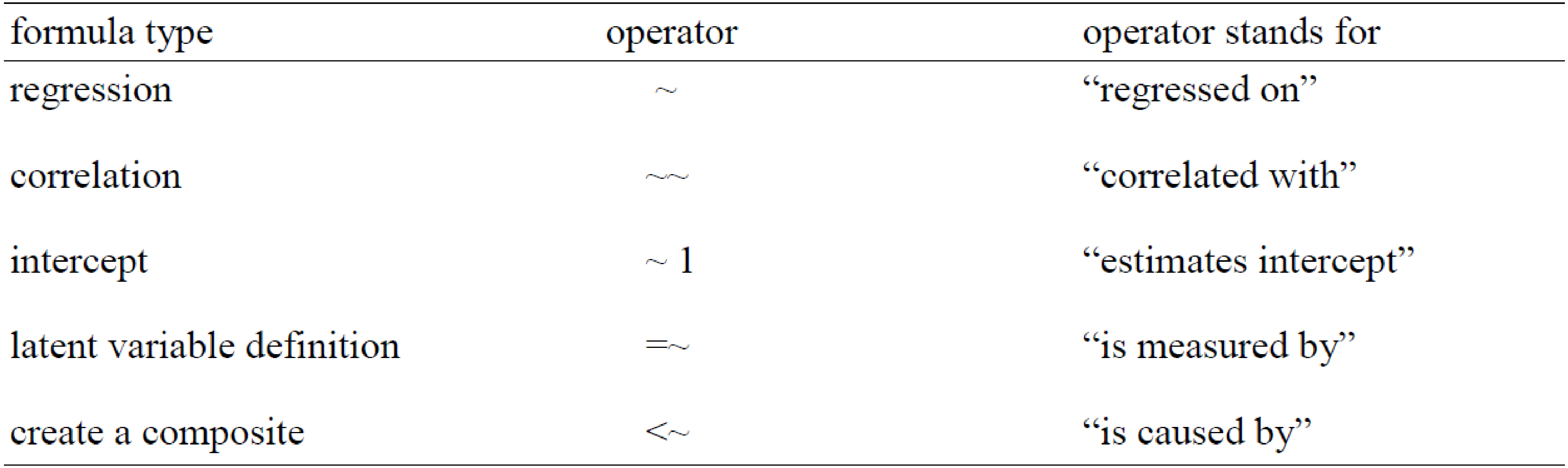
III. Additional Topics and lavaan Commands

- I. More lavaan syntax
- J. Composite variables
- K. Robust estimators
- L. Bootstrapping
- M. Multigroup comparisons

III. Additional Topics and lavaan Commands

I. More lavaan syntax

1. lavaan has a number of operators and syntax options.
   (see “Basic_lavaan_Syntax_Guide_Aug1_2013.pdf”)



formula type	operator	operator stands for
regression	~	“regressed on”
correlation	~~	“correlated with”
intercept	~ 1	“estimates intercept”
latent variable definition	=~	“is measured by”
create a composite	<~	“is caused by”

2. You can work with individual parameters by naming them.

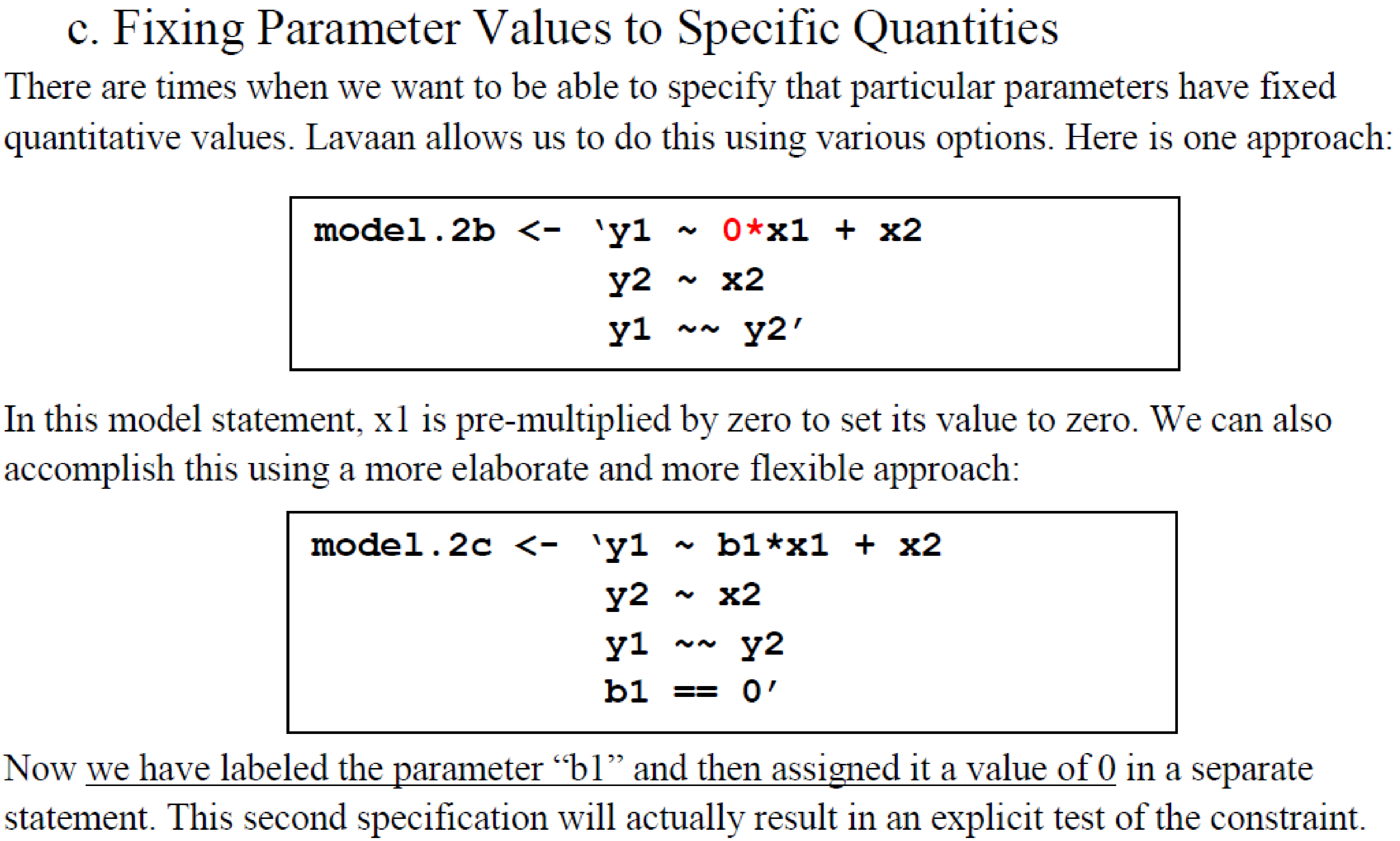
Lavaan names parameters as **“y1 ~ x1”**.

We assign names by pre-multiplying a predictor with the name being assigned.

```
model.2a <- ‘y1 ~ b1*x1
```

Note, parameter labels must start with a letter!

3. Assigning values to parameters by naming them.



c. Fixing Parameter Values to Specific Quantities

There are times when we want to be able to specify that particular parameters have fixed quantitative values. Lavaan allows us to do this using various options. Here is one approach:

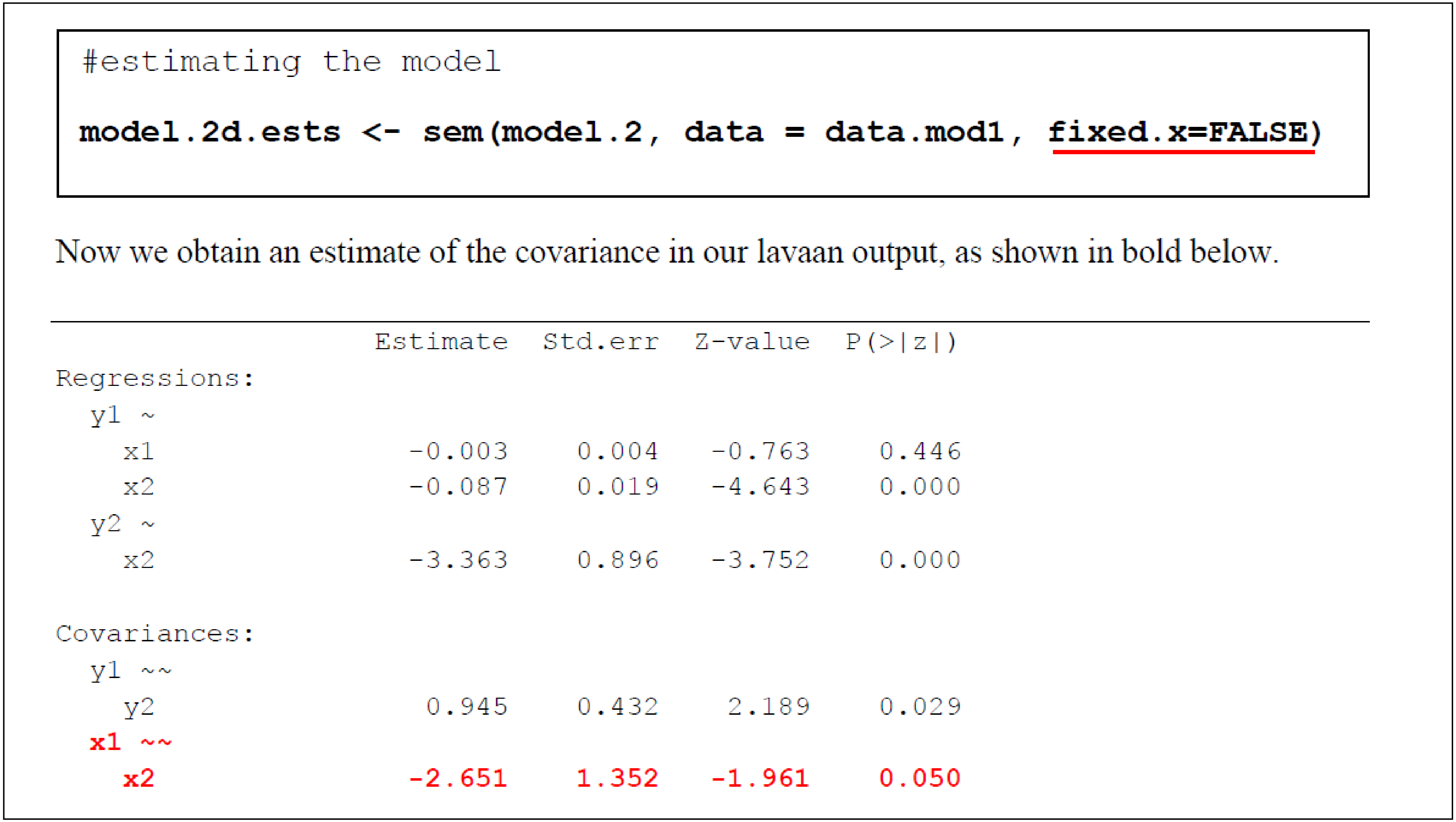
```
model.2b <-  ‘y1 ~ 0*x1 + x2
              y2 ~ x2
              y1 ~~ y2’
```

In this model statement, x1 is pre-multiplied by zero to set its value to zero. We can also accomplish this using a more elaborate and more flexible approach:

```
model.2c <-  ‘y1 ~ b1*x1 + x2
              y2 ~ x2
              y1 ~~ y2
              b1 == 0’
```

Now we have labeled the parameter “b1” and then assigned it a value of 0 in a separate statement. This second specification will actually result in an explicit test of the constraint.

4. Correlations/Covariances between exogenous variables are not usually estimated, but we can.

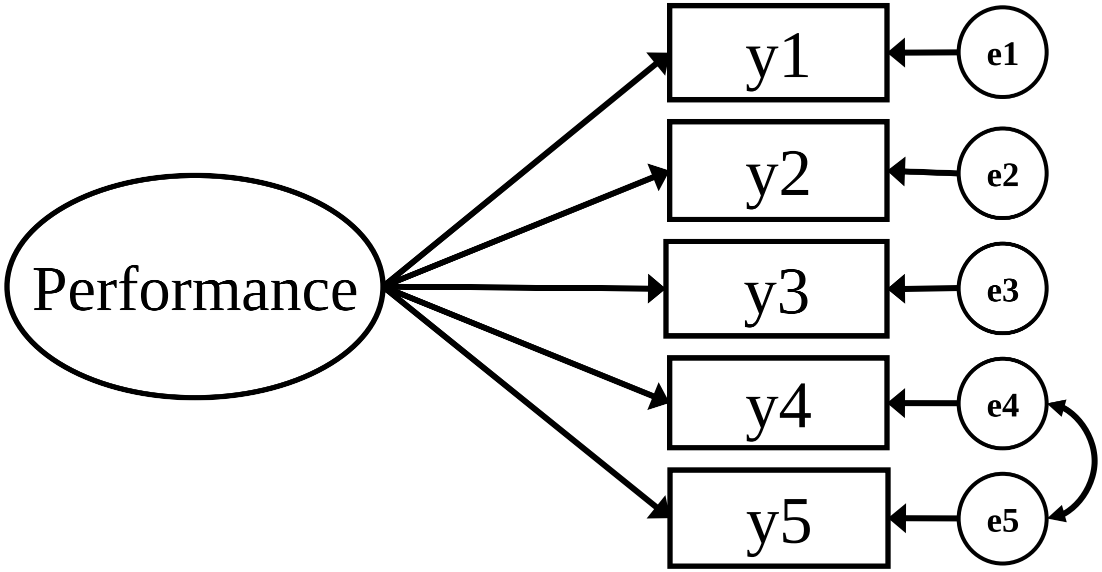


```
#estimating the model

model.2d.ests <- sem(model.2, data = data.mod1, fixed.x=FALSE)
```

Now we obtain an estimate of the covariance in our lavaan output, as shown in bold below.

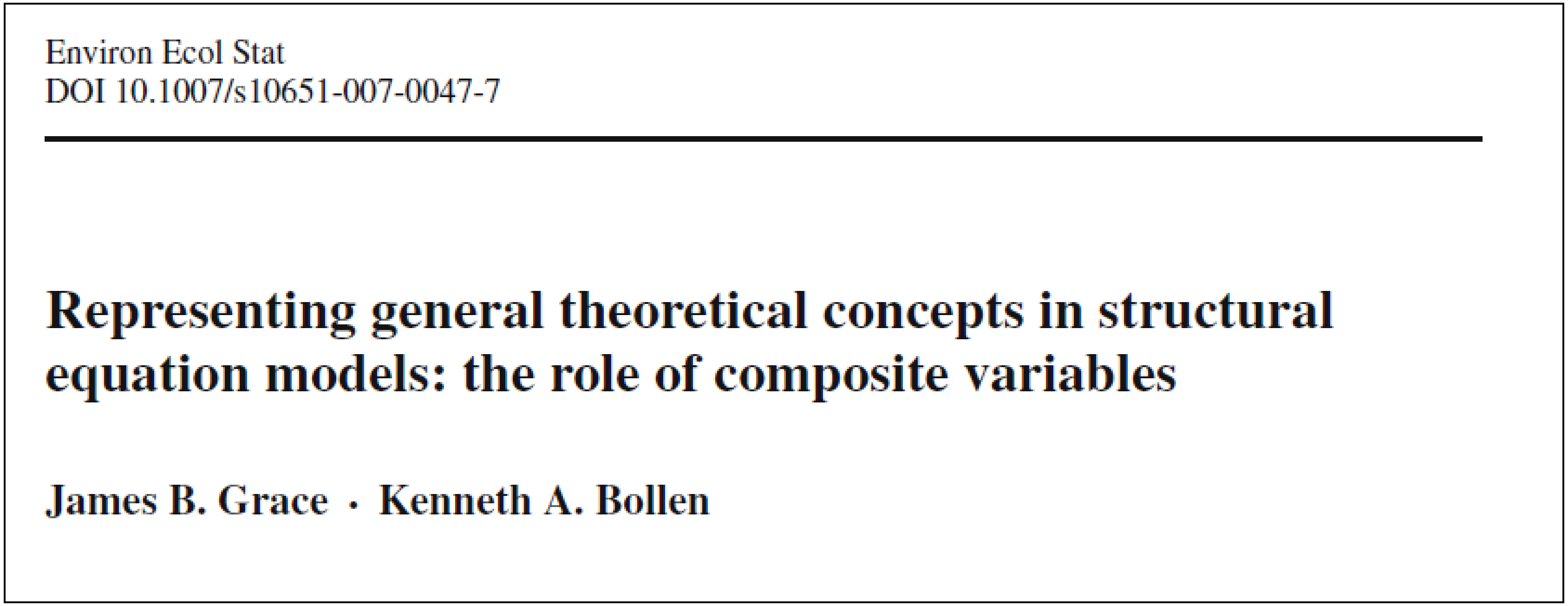
```
                   Estimate  Std.err  Z-value  P(>|z|)
Regressions:
  y1 ~
    x1               -0.003    0.004   -0.763    0.446
    x2               -0.087    0.019   -4.643    0.000
  y2 ~
    x2               -3.363    0.896   -3.752    0.000

Covariances:
  y1 ~~
    y2                0.945    0.432    2.189    0.029
  x1 ~~
    x2               -2.651    1.352   -1.961    0.050
```

5. Lavaan creates latent variables by declaring them in the absence of any known values.



```
lvmod.2 <- ' # Latent variable definition
               Perform =~ stems + infls + clonediam
                          + leafht + leafwdth

             # Error Covariances
               leafht ~~ leafwdth'
```

J. Composite Variables

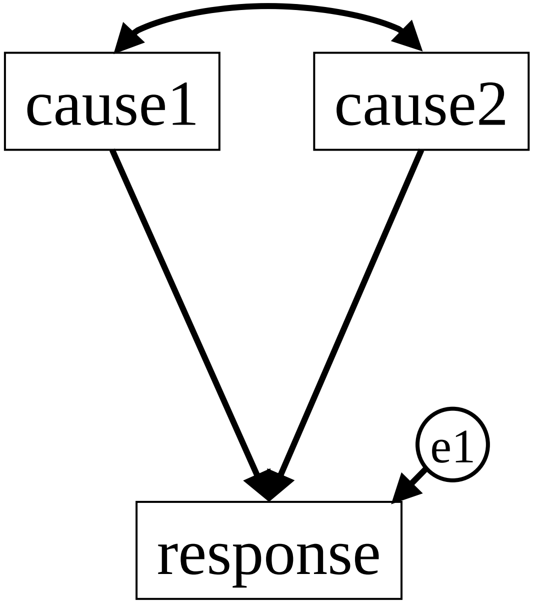


Environ Ecol Stat
DOI 10.1007/s10651-007-0047-7

Representing general theoretical concepts in structural equation models: the role of composite variables

James B. Grace · Kenneth A. Bollen

1. In the simple case, we can use composites to represent collective effects of a set of variables.

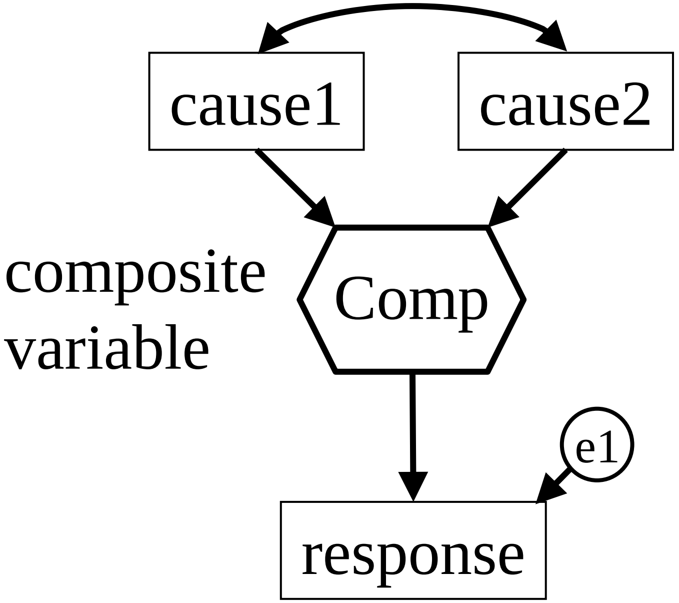

cause1

cause2

e1

response

First step in compositing process.


cause1

cause2

composite variable

Comp

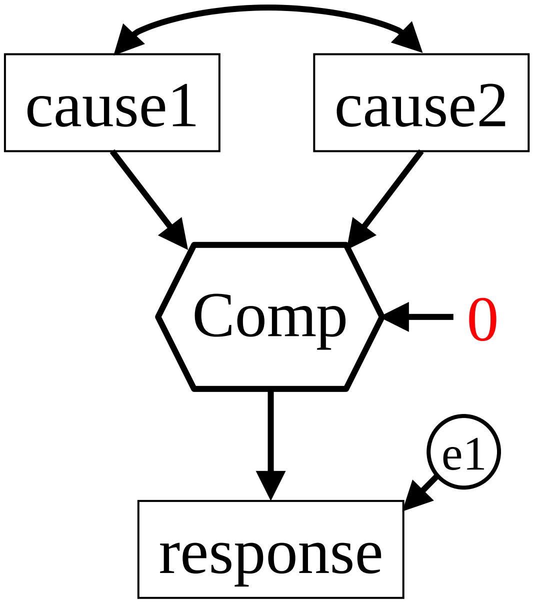
e1

response

Second step in compositing process.

2. Composites are abstractions, and we must declare them.

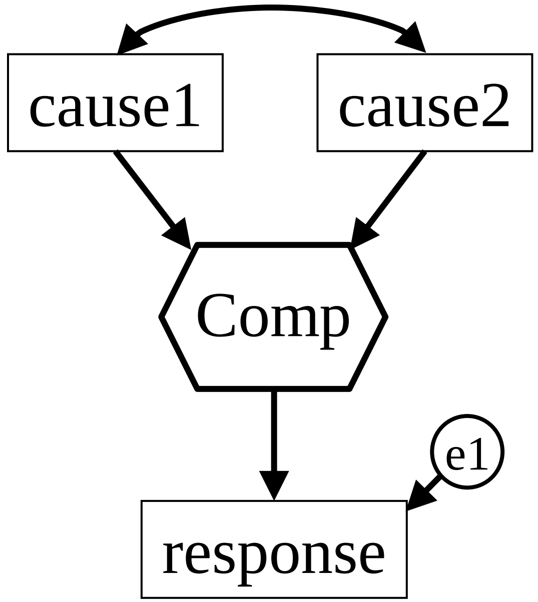
To specify composites, we need to conceptualize a composite as a latent variable with no error (error set = 0).



Composites were not part of the original LISREL conceptualization.

The composite captures the combined effects of its causes on its responses (akin to a multiple regression predictor).

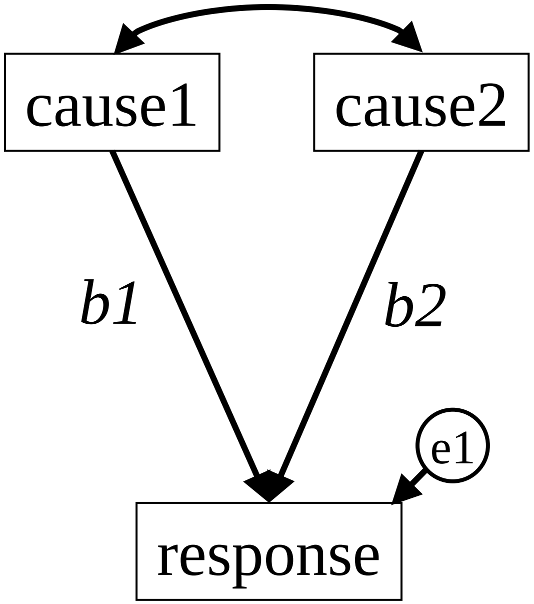
3. We can create composites with lavaan syntax.



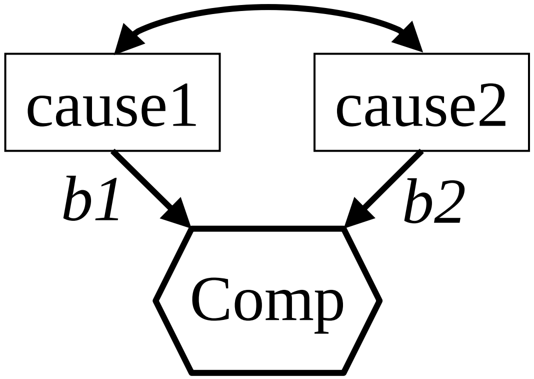
```
# we declare composite as effects of two variables

compmod.z <- 'Comp <~ 1*cause1 + cause2

              response ~ Comp'
```

We use ‘1*cause2’to set scale of composite.

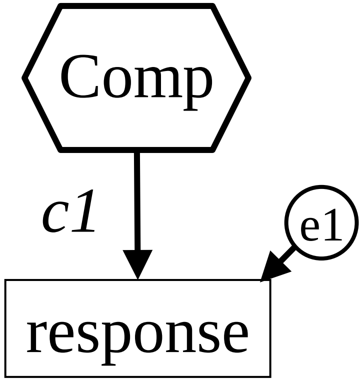
4. We can also compute composite scores by hand.



First, run model without composite and obtain coefficients.



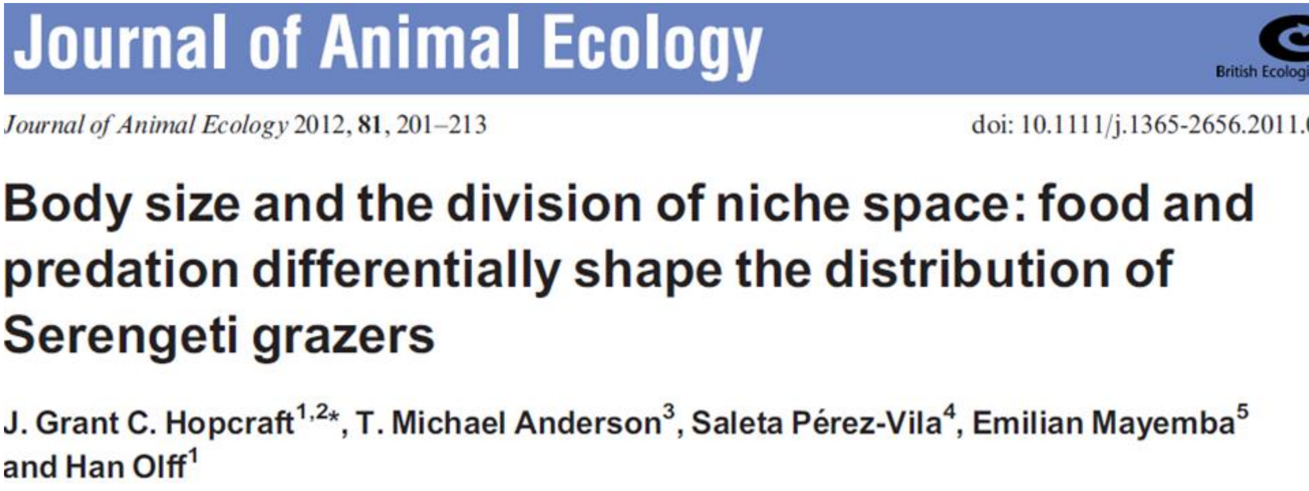
Use coefficients to calculate composite scores.



Use composite scores as a variable.

The coefficient *c1* represents collective effects and is only meaningful in standardized units (normally, its raw value is 1.0).

5. We can use composites to represent general concepts.



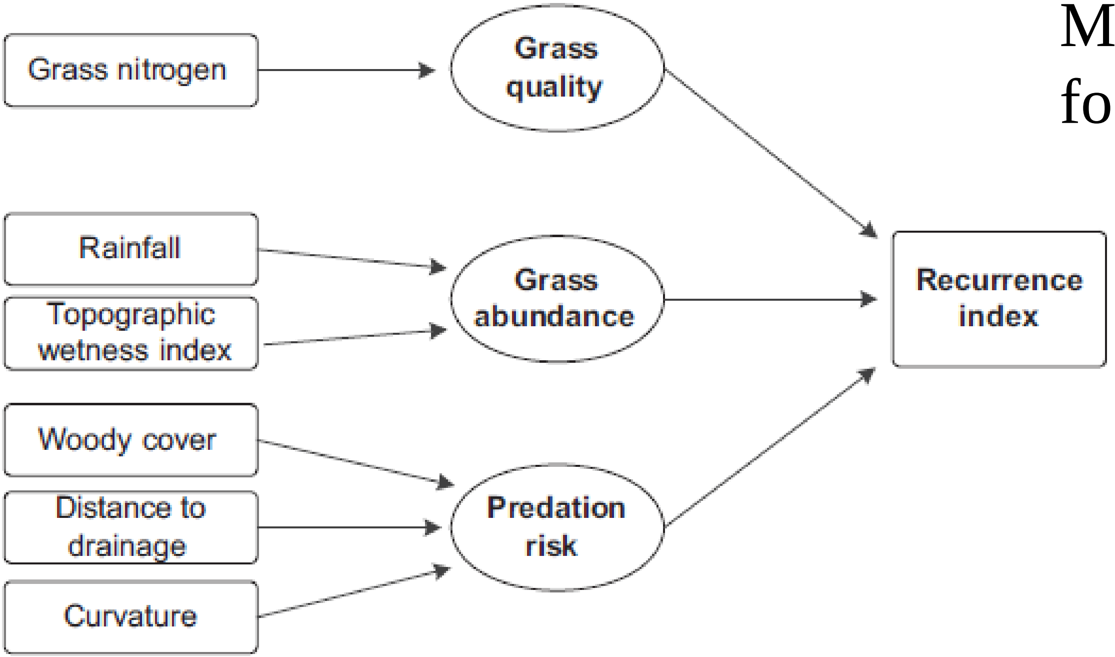
Journal of Animal Ecology

British Ecological Society

Journal of Animal Ecology 2012, **81**, 201–213 doi: 10.1111/j.1365-2656.2011.01885.x

Body size and the division of niche space: food and predation differentially shape the distribution of Serengeti grazers

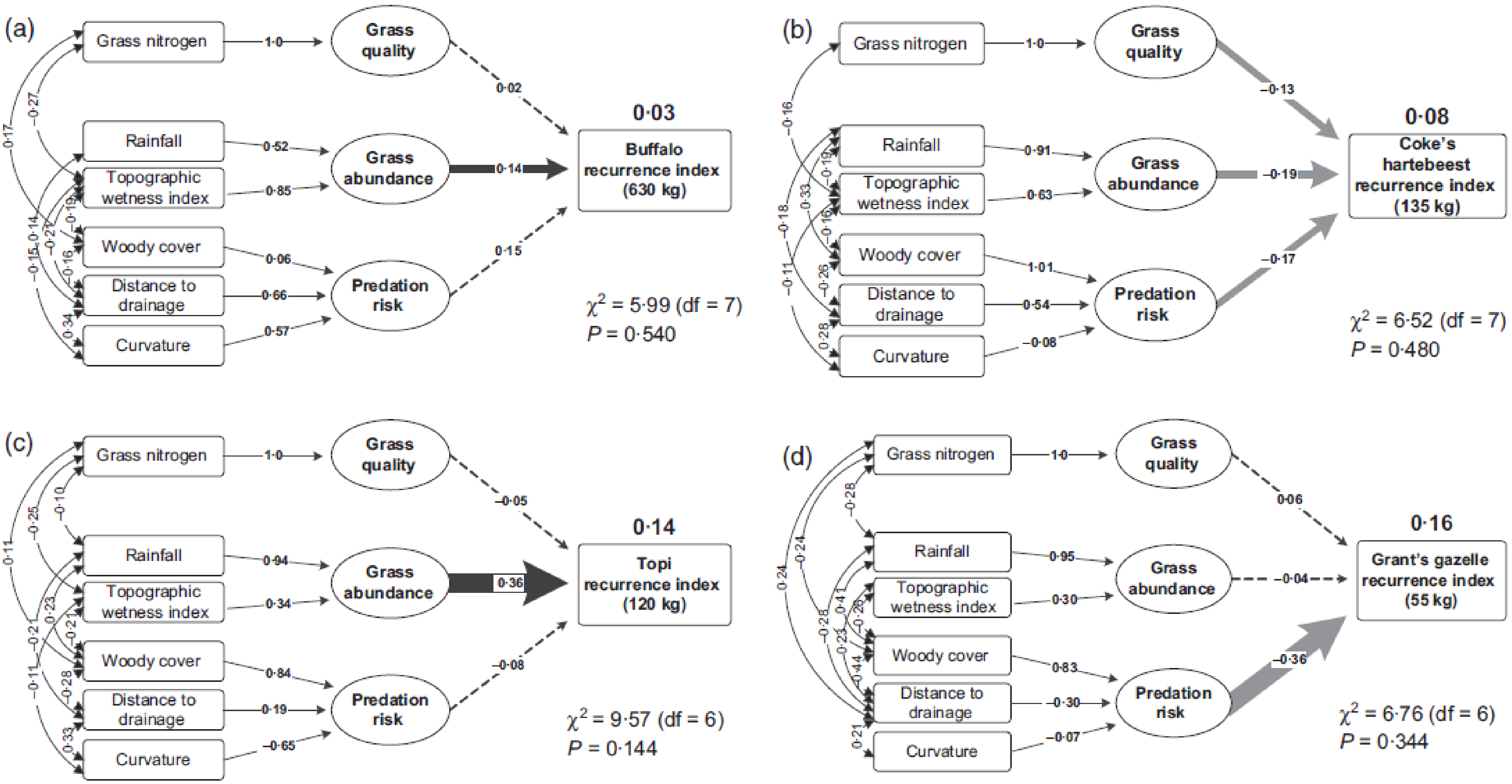
J. Grant C. Hopcraft[1,2]*, T. Michael Anderson[3], Saleta Pérez-Vila[4], Emilian Mayemba[5] and Han Olff[1]



Model of predation risk versus food rewards for grazers.

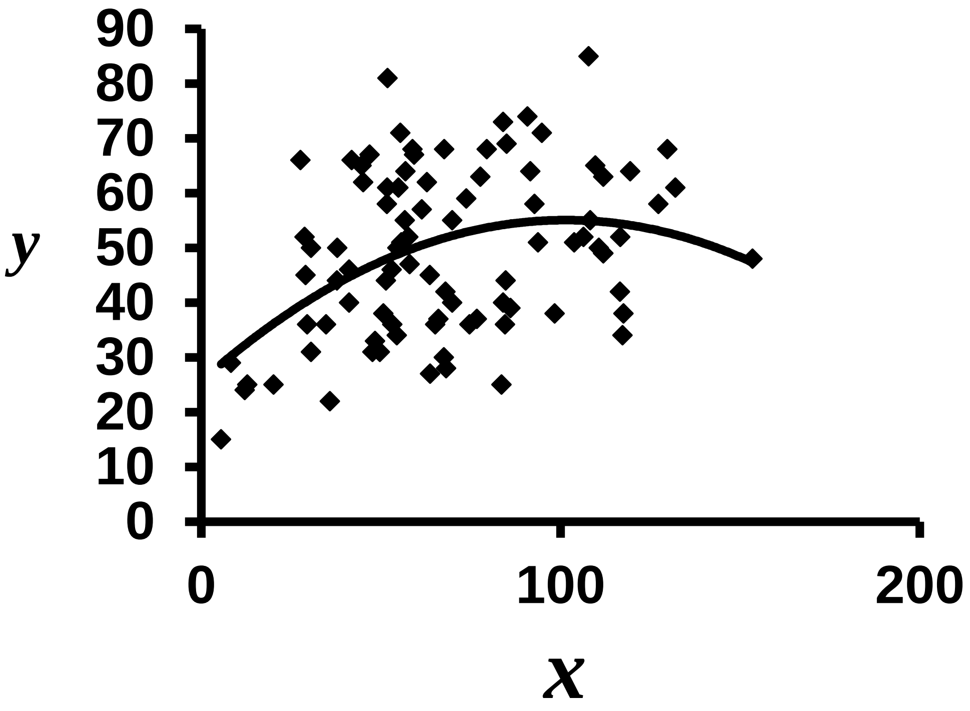
Fig. 2. The *a priori* structural equation model used to assess the

6. And we can use them to compare apples and oranges (with care in interpretation).



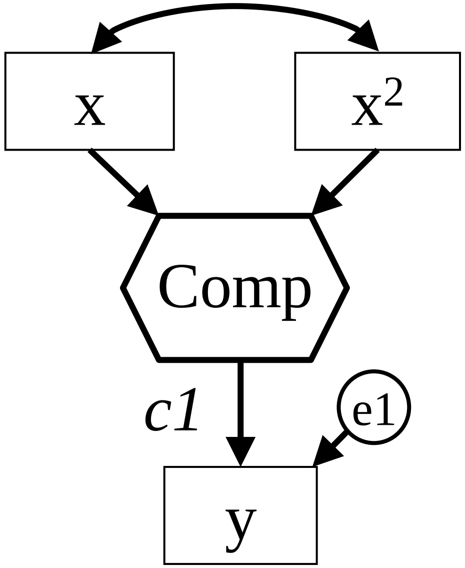
7. We can also use composites as devices.

Consider this nonlinear relationship.



We can model this using polynomial regression:

$$y = \gamma x + \gamma x^2 + \varepsilon$$



Centering a variable before squaring it reduces the autocorrelation between polynomial terms.

```
# center x before squaring
x <- x- mean(x)
x2 <- x^2
```

Discussion

K. Robust Estimators

1. Lavaan permits use of “robust” estimation.

Lavaan has two main options for robust estimation:

MLM – produces chi-squares and standard errors robust to non-normality. AKA the Satorra-Bentler correction.

MLR – similar to MLM, but uses the Yuan-Bentler method so that missing data can be accommodated.

see discussion in:
Yuan & Bentler. 2000. In Sobel & Becker (eds.) Sociological Methodology (pp 165-200)

2. Robust estimation invoked with ‘estimator =’ command.

```
# create model
mod <- 'y2 ~ y1
        y1 ~ x1'

# estimate model
mod.fit <- sem(mod, data=dat, fixed.x=F,
               estimator="mlm")

# get results
summary(mod.fit)
```

“fixed.x=F” is required when using “mlm” option.

3. Results

```
 Estimator                              ML       Robust
  Chi-square                         4.213        4.082
  Degrees of freedom                     1            1
  P-value                            0.040        0.043
  Scaling correction factor
  for the Yuan-Bentler correction                 1.032
```

```
 Standard Errors                      Robust.mlm

              Estimate  Std.err  Z-value  P(>|z|)
Regressions:
  y2 ~
    y1          -0.154    0.024   -6.386    0.000
  y1 ~
    x1           1.185    0.290    4.091    0.000
```

L. Bootstrapping

1. Lavaan has resampling methods for non-normal data.

```
# fit model and request bootstrapped results

mod.fit <- sem(mod, dat=dat,
           test="boot", se="boot", bootstrap=200)
```

2. Results

```
 Estimator                               ML
  Chi-square                          4.213
  Degrees of freedom                      1
  P-value                             0.040
  P-value (Bollen-Stine)              0.053
```

```
Estimate  Std.err  Z-value  P(>|z|)
Regressions with Robust.mlm standard errors:
  y2.ln ~
    y1.ln      -0.154    0.024   -6.386    0.000
  y1.ln ~
    x1.ln       1.185    0.290    4.091    0.000

Regressions with bootstrapped standard errors:
  y2.ln ~
    y1.ln      -0.154    0.027   -5.750    0.000
  y1.ln ~
    x1.ln       1.185    0.423    2.800    0.005
```

Discussion

M. Multigroup Modeling

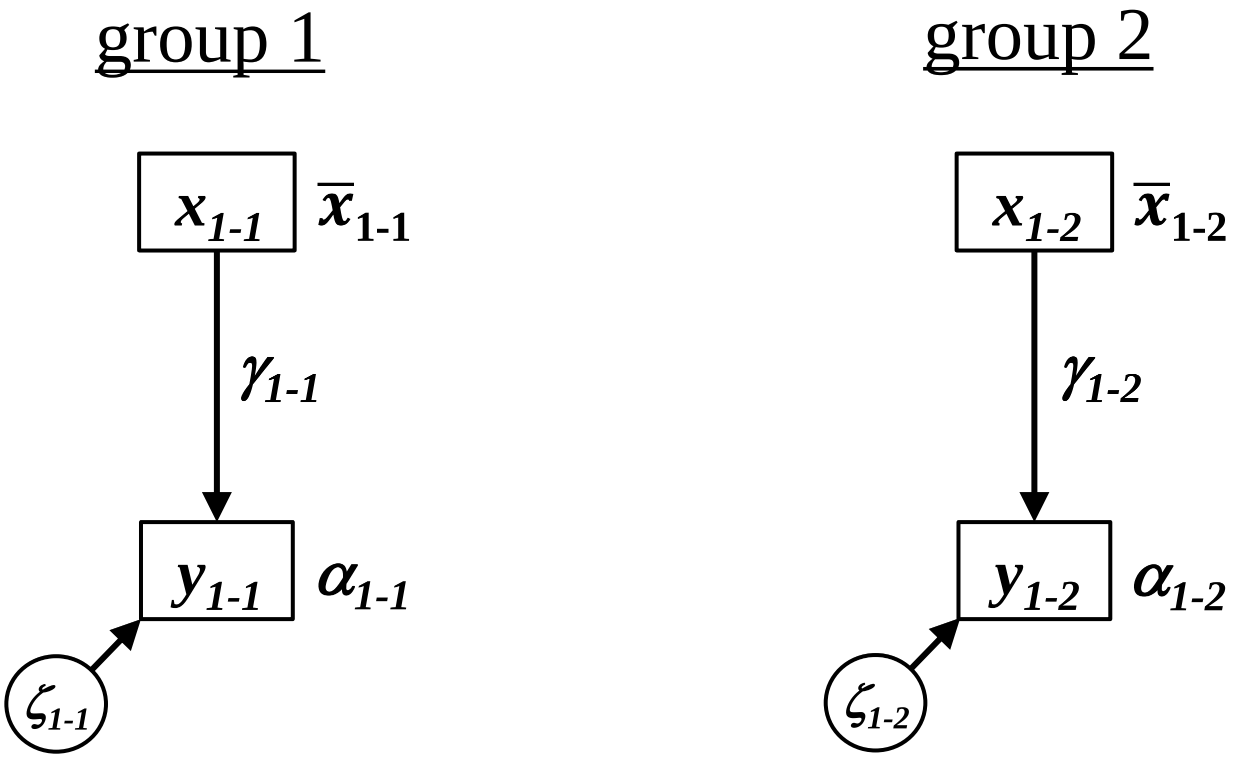
1. Multigroup modeling involves situations where there are discrete groups in the data that you want to compare.

- males versus females in a population
- treated versus control plots
- areas with different disturbance histories

In multigroup modeling, we develop a common model for different groups and then ask what parameters are the same or different between groups.

We can use the classical likelihood-based measures (e.g., model chi-square) to test for constraints across groups.

2. There are several parameters that can be compared.



A number of hypotheses we can test:

(1) equal slopes: $\gamma_{1-1} = \gamma_{1-2}$

(2) equal intercepts: $\alpha_{1-1} = \alpha_{1-2}$

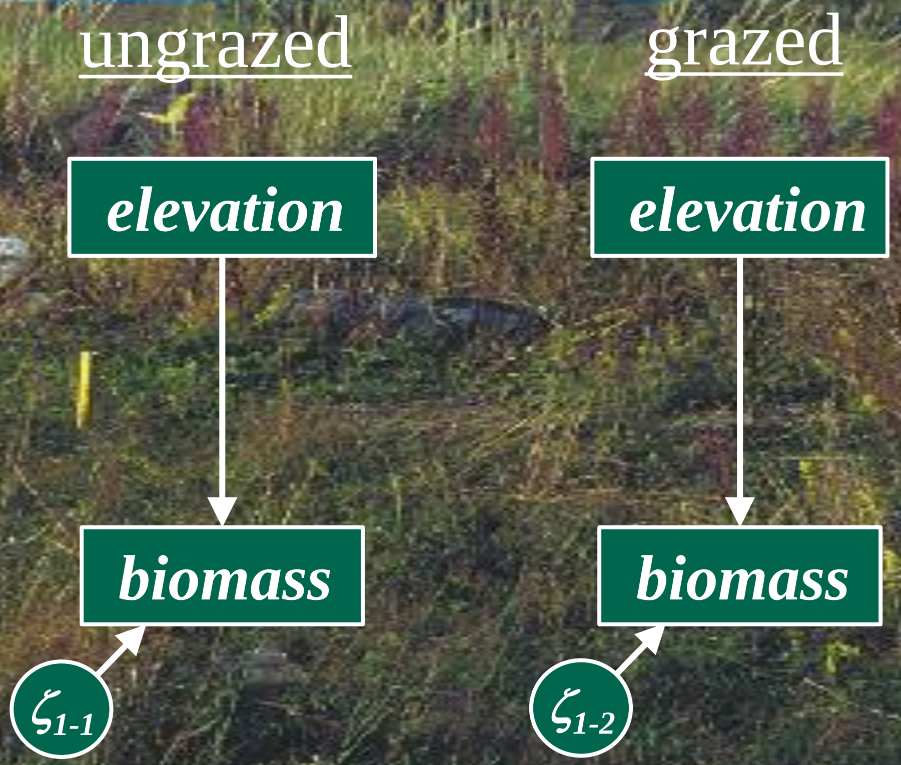
(3) equal means: $\bar{x}_{1-1} = \bar{x}_{1-2}$

(4) equal errors: $\zeta_{1-1} = \zeta_{1-2}$

↓ increasingly strong constraints

Effects of Grazing on Finnish Coastal Meadows*

How does biomass respond to the interaction between grazing & elevation?



<u>ungrazed</u>

elevation

biomass

ζ_{1-1}

<u>grazed</u>

elevation

biomass

ζ_{1-2}

Data from 1-m^2 plots arrayed along an elevation gradient in each of several paired grazed and ungrazed meadows in SW Finland.

Grace, J.B. and Jutila, H. (1999) The relationship between species density and community biomass in grazed and ungrazed coastal meadows. *Oikos*, 85:398-408.

1. Lavaan uses a ‘group=’ command to invoke a multigroup analysis.

```
#lavaan code for basic model
mod1 <-'biomass ~ elev'

#fit the model, specifying groups
mod1.fit <- sem(mod1, data=sem.dat, group="grazed")

#request output
summary(mod1.fit)
```

2. Default allows all parameters to differ between groups.

```
> summary(mod1.fit)
lavaan (0.5-12) converged normally after  24 iterations

  Number of observations per group
  1                                                 165
  0                                                 189

  Estimator                                          ML
  Minimum Function Test Statistic                 0.000
  Degrees of freedom                                  0
  P-value (Chi-square)                            0.000

Chi-square for each group:

  1                                               0.000
  0                                               0.000
```

There are no equality constraints and therefore no chi-square tests.

3. Default allows all parameters to differ between groups.

```
Group 1 [1]:  (notice Group=1 = grazed)
                   Estimate  Std.err  Z-value  P(>|z|)
Regressions:
  biomass ~
    elev             -0.474    0.205   -2.311    0.021
Intercepts:
    biomass           5.263    0.117   45.072    0.000
Variances:
    biomass           0.534    0.059

Group 2 [0]:
                   Estimate  Std.err  Z-value  P(>|z|)
Regressions:
  biomass ~
    elev             -0.798    0.145   -5.523    0.000
Intercepts:
    biomass           5.926    0.065   91.069    0.000
Variances:
    biomass           0.296    0.030    0.000
```

4. We can test equality constraints by labeling parameters.

```
#lavaan code naming the path coefficient “b1”
mod2 <-'biomass ~ c(“b1”,”b1”)*elev'

#fit the model, specifying groups
mod2.fit <- sem(mod2, data=sem.dat, group="grazed")
```

When you label a parameter across groups, you have to pass to lavaan a vector of labels, one for each group. Here, **`c(“b1”,”b1”)`** is a vector of labels.

5. With constraints imposed, we can test if models sig. different.

```
> summary(mod2.fit)
lavaan (0.5-12) converged normally after  19 iterations

  Estimator                                         ML
  Minimum Function Test Statistic                1.668
  Degrees of freedom                                 1
  P-value (Chi-square)                           0.197

Chi-square for each group:

  1                                              1.116
  0                                              0.552
```

The overall model chi-square of 1.668 with 1 df would traditionally be interpreted as a non-significant difference between models.

6. With constraints imposed, we can test if parameters different.

```
Group 1 [1]:
                   Estimate  Std.err  Z-value  P(>|z|)
Regressions:
  biomass ~
    elev      (b1)   -0.691    0.118   -5.836    0.000
Intercepts:
    biomass           5.371    0.082   65.493    0.000
Variances:
    biomass           0.538    0.059

Group 2 [0]:
                   Estimate  Std.err  Z-value  P(>|z|)
Regressions:
  biomass ~
    elev      (b1)   -0.691    0.118   -5.836    0.000
Intercepts:
    biomass           5.888    0.058  101.552    0.000
Variances:
    biomass           0.297    0.031
```

We get one best estimate for both groups.

7. We can also use an ‘equal’ or a ‘group.equal’ command.

```
mod2a <-'biom.log ~ equal("b1")*elev.m'

mod2a.fit <- sem(mod2a, data=sem.dat, group="grazed")
```

Produces exactly the same results as the previous command.

```
mod3 <-'biom.log ~ elev.m`

mod3.fit <- sem(mod3, data=sem.dat, group="grazed",
group.equal="regressions")
```

8. There are a number of helpful options for the ‘group.equal’ command.

```
group.equal=c(
"intercepts",
"means",
"regressions",
"residuals",
"residual.covariances")
```

This gives you a taste of the possibilities for testing equality constraints across groups.

For more, consult the tutorial

http://lavaan.ugent.be/tutorial/tutorial.pdf

Discussion