Reticular Action Model (RAM) Notation Notes

Ivan Jacob Agaloos Pesigan

2021-02-14

Contents

Chapter 1

Description

This is a collection of my notes on the Reticular Action Model (RAM) notation that accompanies the `ramR` package (Pesigan, 2021) in the `R` statistical environment (R Core Team, 2020). You can install the released version of `ramR` from GitHub with:

```
remotes::install_github("jeksterslab/ramR")
```

These notes are based on the following resources:

- Boker and McArdle (2005)
- McArdle and McDonald (1984)
- McArdle (2005)

See GitHub Pages for the html deployment.

Chapter 2

Reticular Action Model (RAM) Matrix Notation

2.1 Full Model

Definition 2.1.

$$\mathbf{v} = \mathbf{A}\mathbf{v} + \mathbf{u} \tag{2.1}$$

where

- $\mathbf{v}$ and $\mathbf{u}$ are $t \times 1$ vectors of random variables
- $\mathbf{A}$ is a $t \times t$ matrix of *directed* or *asymmetric* relationship from column variable v_j to row variable v_i
  - $\mathbf{A}$ represent the regression of each of the t variables $\mathbf{v}$ on the other $t-1$ variables
  - diagonal $a_{i,i}$ is zero
  - u_i represent the residual of v_i
  - if all regression coefficients on other variables are zero (i.e., i^{th} row of $\mathbf{A}$ consists of zeros), then the variable v_i is considered the same as its own residual u_i

Definition 2.2.

$$\mathbf{S} = \mathbb{E}\left\{\mathbf{u}\mathbf{u}'\right\}, \tag{2.2}$$

where

- $\mathbf{S}$ is a $t \times t$ matrix of *undirected* or *symmetric* relationship
  - the notation Ω is used in other sources for $\mathbf{S}$
- $\mathbb{E}$ is the expectation operator

Definition 2.3.

$$\mathbf{C} = \mathbb{E}\left\{\mathbf{v}\mathbf{v}'\right\}, \tag{2.3}$$

where

- $\mathbf{C}$ is a $t \times t$ variance-covariance matrix
  - the notation Σ is used in other sources for $\mathbf{C}$

Definition 2.4.

$$\mathbf{v} = \mathbf{A}\mathbf{v} + \mathbf{u}$$

can be rewritten as

$$\begin{aligned} \mathbf{v} - \mathbf{A}\mathbf{v} &= \mathbf{u} \\ \mathbf{u} &= \mathbf{v} - \mathbf{A}\mathbf{v} \\ \mathbf{u} &= \left(\mathbf{I} - \mathbf{A}\right)\mathbf{v} \end{aligned} \tag{2.4}$$

assuming that $(\mathbf{I} - \mathbf{A})$ is non-singular,

$$\mathbf{E} = \left(\mathbf{I} - \mathbf{A}\right)^{-1} \tag{2.5}$$

then

$$\begin{aligned} \mathbf{v} &= \left(\mathbf{I} - \mathbf{A}\right)^{-1}\mathbf{u} \\ &= \mathbf{E}\mathbf{u}. \end{aligned} \tag{2.6}$$

Using the definitions above, $\mathbf{S}$ and $\mathbf{C}$ are given by

$$\begin{aligned} \mathbf{S} &= \left(\mathbf{I} - \mathbf{A}\right)\mathbf{C}\left(\mathbf{I} - \mathbf{A}\right)^{-1} \\ &= \mathbf{E}^{-1}\mathbf{C}\left(\mathbf{E}^{-1}\right)^{\intercal} \end{aligned} \tag{2.7}$$

$$\begin{aligned} \mathbf{C} &= \left(\mathbf{I} - \mathbf{A}\right)^{-1}\mathbf{S}\left[\left(\mathbf{I} - \mathbf{A}\right)^{-1}\right]^{\intercal} \\ &= \mathbf{E}\mathbf{S}\mathbf{E}^{\intercal} \end{aligned} \tag{2.8}$$

2.2 Given vs. Hidden Variables

Definition 2.5.

$$\mathbf{v} = \begin{bmatrix} \mathbf{g}_{p \times 1} \\ \mathbf{h}_{q \times 1} \end{bmatrix} \tag{2.9}$$

$$t = p + q \tag{2.10}$$

- $\mathbf{g}$ may be considered observed, manifest or *given* variables
- $\mathbf{h}$ may be considered unobserved, latent, or *hidden* variables

Definition 2.6.

$$\mathbf{F} = \left[\mathbf{I}_{p \times p} : \mathbf{0}_{p \times q}\right] \tag{2.11}$$

- the $\mathbf{F}$ matrix acts as a *filter* to select the manifest variables out of the full set of manifest and latent variables

$$\mathbf{g} = \mathbf{F}\mathbf{v} \tag{2.12}$$

$$\begin{aligned} \mathbf{g} &= \mathbf{F}\left(\mathbf{I} - \mathbf{A}\right)^{-1}\mathbf{u} \\ &= \mathbf{F}\mathbf{E}\mathbf{u} \end{aligned} \tag{2.13}$$

Definition 2.7.

$$\mathbf{M} = \mathbb{E}\left\{\mathbf{g}\mathbf{g}^\top\right\} \tag{2.14}$$

$$\begin{aligned} \mathbf{M} &= \mathbf{F}\left(\mathbf{I} - \mathbf{A}\right)^{-1}\mathbf{S}\left[\left(\mathbf{I} - \mathbf{A}\right)^{-1}\right]^\top\mathbf{F}^\top \\ &= \mathbf{F}\mathbf{E}\mathbf{S}\mathbf{E}^\top\mathbf{F}^\top \\ &= \mathbf{F}\mathbf{C}\mathbf{F}^\top \end{aligned} \tag{2.15}$$

- when components of $\mathbf{v}$ are permuted, the columns of $\mathbf{F}$ can be correspondingly permuted
- the rows and columns of $\mathbf{C}$ that are filtered out by $\mathbf{F}$ contain useful information about the latent variable structure.

The equations above completely define RAM.

Chapter 3

Reticular Action Model (RAM) Path Diagram

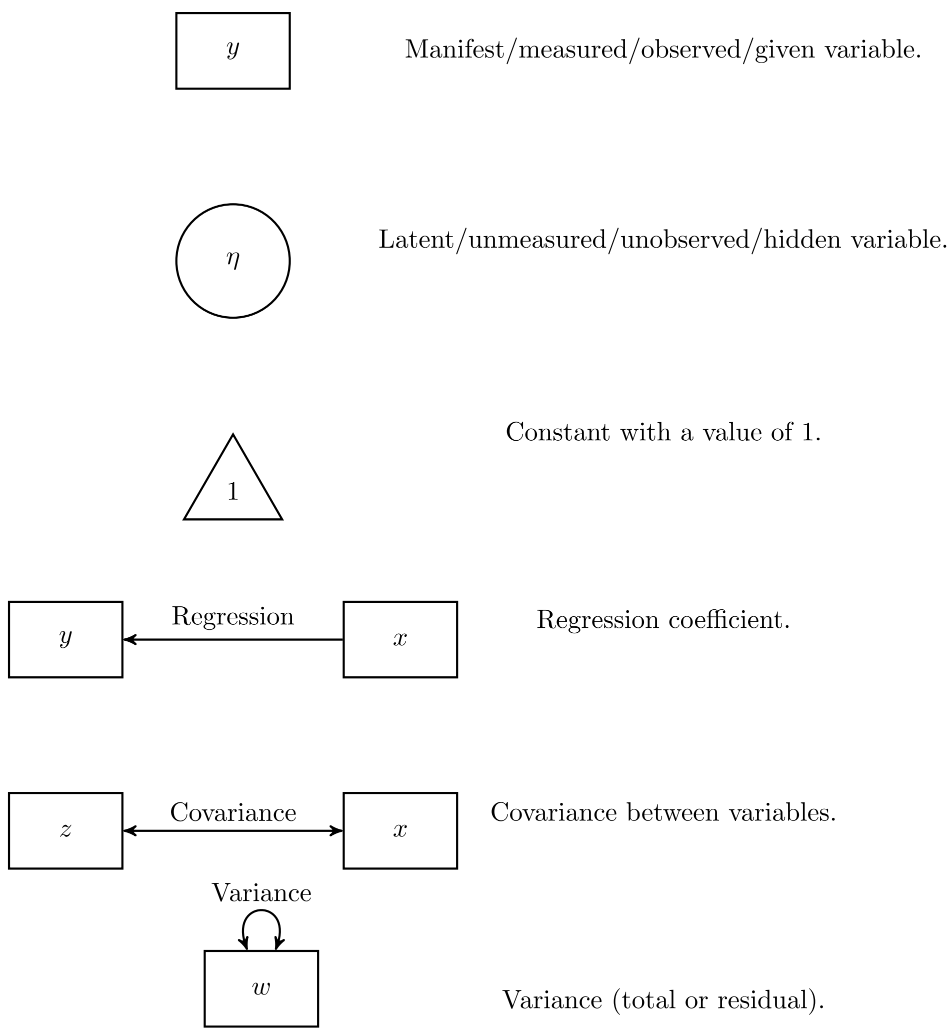
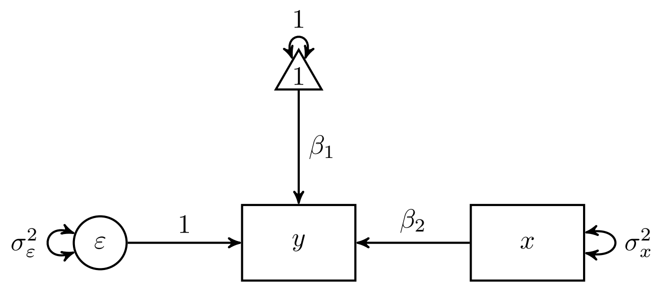


Figure 3.1: Path Diagram Elements



$$y = \alpha + \beta x + \varepsilon$$

Figure 3.2: Two-Variable Regression Model

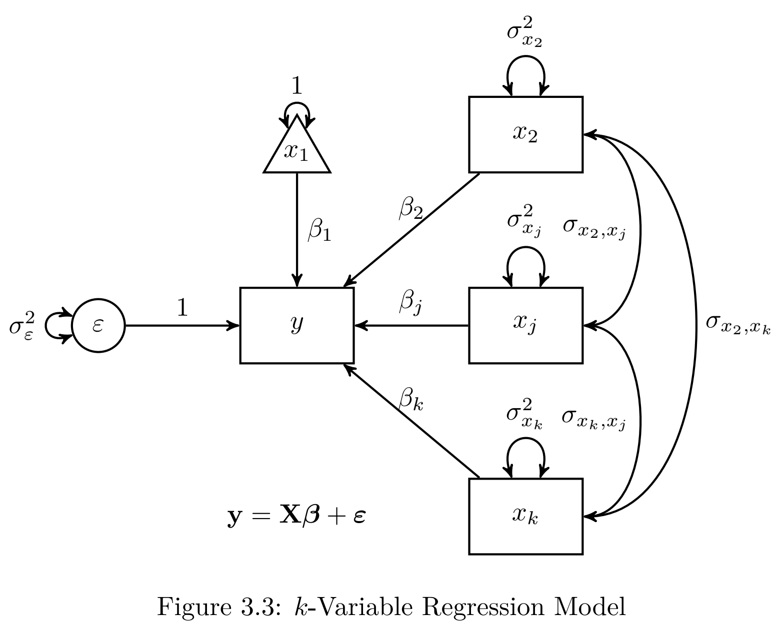
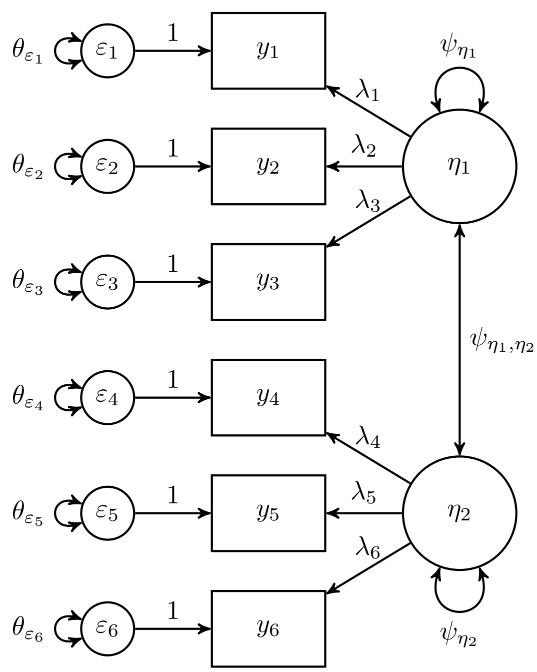


Figure 3.3: k-Variable Regression Model



$$\mathbf{y} = \boldsymbol{\Lambda}\boldsymbol{\eta} + \boldsymbol{\varepsilon}$$

Figure 3.4: Two-Factor Confirmatory Factor Analysis Model

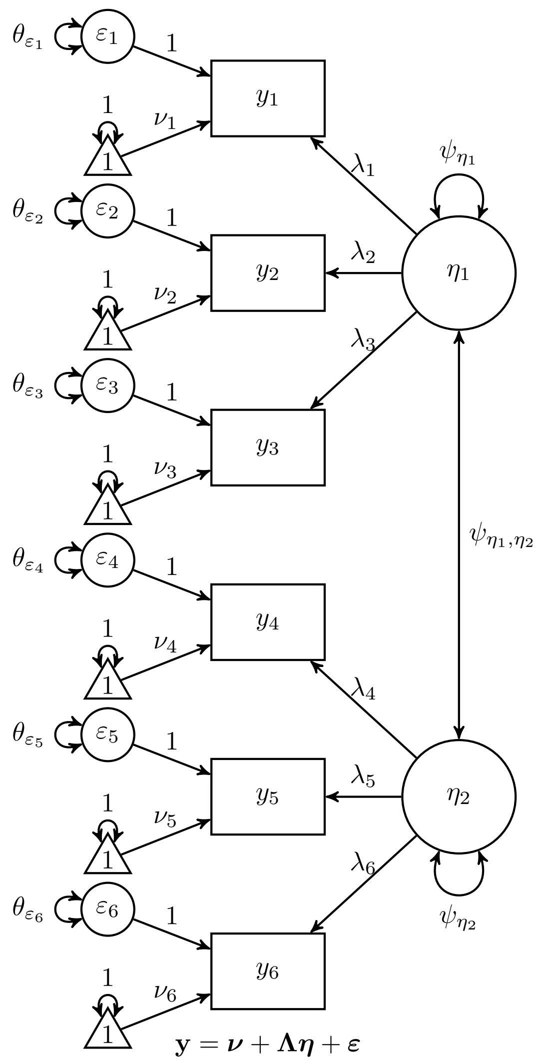
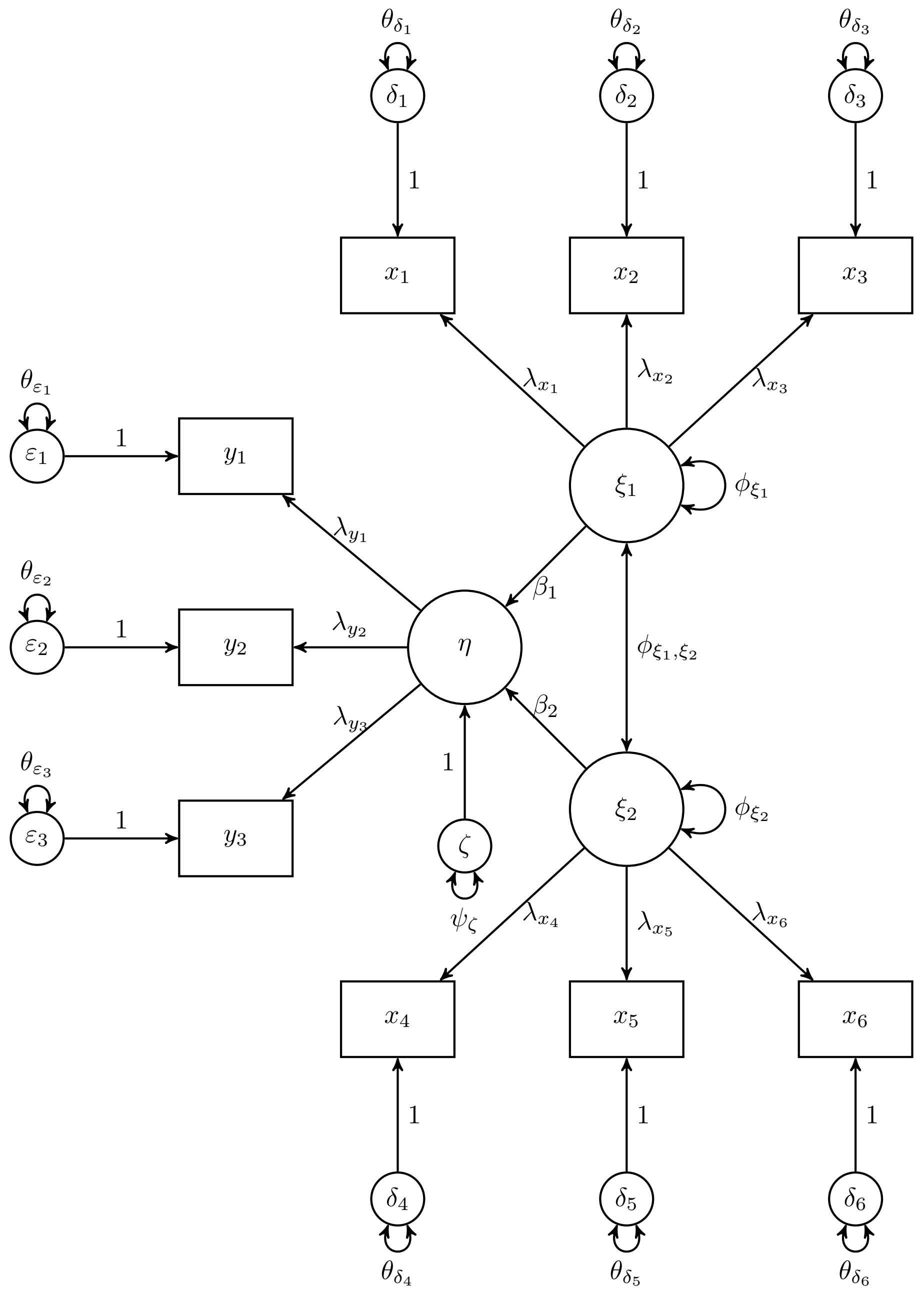


Figure 3.5: Two-Factor Confirmatory Factor Analysis Model with Mean Structure



$$\boldsymbol{\eta} = \mathbf{B}\boldsymbol{\eta} + \boldsymbol{\Gamma}\boldsymbol{\xi} + \boldsymbol{\zeta}, \mathbf{y} = \boldsymbol{\Lambda}_y\boldsymbol{\eta} + \boldsymbol{\varepsilon}, \mathbf{x} = \boldsymbol{\Lambda}_x\boldsymbol{\xi} + \boldsymbol{\delta}$$

Figure 3.6: Path Model with Latent Variables

Chapter 4

Student's t-test

In this section, the Student's t-test is presented as a structural equation model using the RAM notation. Let y be a continuous dependent variable, x be a dichotomous independent variable ($x = \{0, 1\}$), and ε be the stochastic error term with mean 0 and constant variance of σ^2_{ε} across the values of x. The associations of the variables are given by

$$y = \alpha + \beta x + \varepsilon$$

where

- α is the expected value of y when $x = 0$
- β is the unit change in y for unit change in x
- $\alpha + \beta$ is the expected value of y when $x = 1$

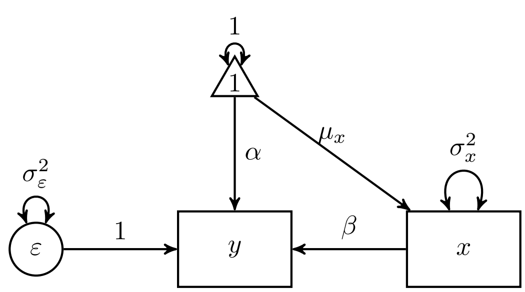


Figure 4.1: Student's t-test

4.1 Symbolic

Let $\{y, x, \varepsilon\}$ be the variables of interest.

$$\mathbf{A} = \begin{pmatrix} 0 & \beta & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$\mathbf{S} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & \sigma_x^2 & 0 \\ 0 & 0 & \sigma_\varepsilon^2 \end{pmatrix}$$

$$\begin{aligned}
\mathbf{C} &= (\mathbf{I} - \mathbf{A})^{-1}\,\mathbf{S}\left[(\mathbf{I} - \mathbf{A})^{-1}\right]^{\mathsf{T}} \\
&= \mathbf{E}\mathbf{S}\mathbf{E}^{\mathsf{T}} \\
&= \begin{pmatrix} 1 & \beta & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 \\ 0 & \sigma_x^2 & 0 \\ 0 & 0 & \sigma_\varepsilon^2 \end{pmatrix} \begin{pmatrix} 1 & \beta & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}^{\mathsf{T}} \\
&= \begin{pmatrix} \sigma_x^2\beta^2 + \sigma_\varepsilon^2 & \beta\sigma_x^2 & \sigma_\varepsilon^2 \\ \sigma_x^2\beta & \sigma_x^2 & 0 \\ \sigma_\varepsilon^2 & 0 & \sigma_\varepsilon^2 \end{pmatrix}
\end{aligned}$$

$$\mathbf{F} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

$$\begin{aligned}
\mathbf{M} &= \mathbf{F}\,(\mathbf{I} - \mathbf{A})^{-1}\,\mathbf{S}\left[(\mathbf{I} - \mathbf{A})^{-1}\right]^{\mathsf{T}}\mathbf{F}^{\mathsf{T}} \\
&= \mathbf{F}\mathbf{E}\mathbf{S}\mathbf{E}^{\mathsf{T}}\mathbf{F}^{\mathsf{T}} \\
&= \mathbf{F}\mathbf{C}\mathbf{F}^{\mathsf{T}} \\
&= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \sigma_x^2\beta^2 + \sigma_\varepsilon^2 & \beta\sigma_x^2 & \sigma_\varepsilon^2 \\ \sigma_x^2\beta & \sigma_x^2 & 0 \\ \sigma_\varepsilon^2 & 0 & \sigma_\varepsilon^2 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}^{\mathsf{T}} \\
&= \begin{pmatrix} \sigma_x^2\beta^2 + \sigma_\varepsilon^2 & \beta\sigma_x^2 \\ \sigma_x^2\beta & \sigma_x^2 \end{pmatrix}
\end{aligned}$$

$$\begin{aligned}
\mathbf{v} &= (\mathbf{I} - \mathbf{A})^{-1}\,\mathbf{u} \\
&= \left[\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} 0 & \beta & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}\right]^{-1} \begin{pmatrix} \alpha \\ \mu_x \\ 0 \end{pmatrix} \\
&= \begin{pmatrix} \alpha + \beta\mu_x \\ \mu_x \\ 0 \end{pmatrix}
\end{aligned}$$

$$
\begin{aligned}
\mathbf{u} &= (\mathbf{I} - \mathbf{A})\,\mathbf{v} \\
&= \left[\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} 0 & \beta & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \right] \begin{pmatrix} \alpha + \beta\mu_x \\ \mu_x \\ 0 \end{pmatrix} \\
&= \begin{pmatrix} \alpha + \beta\mu_x \\ \mu_x \\ 0 \end{pmatrix}
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{g} &= \mathbf{F}\,(\mathbf{I} - \mathbf{A})^{-1}\,\mathbf{u} \\
&= \left[\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} 0 & \beta & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \right]^{-1} \begin{pmatrix} \alpha \\ \mu_x \\ 0 \end{pmatrix} \\
&= \begin{pmatrix} \alpha + \beta\mu_x \\ \mu_x \end{pmatrix}
\end{aligned}
$$

4.1.1 Using the `ramR` Package

```
A
```

```
##   y   x      e
## y "0" "beta" "1"
## x "0" "0"    "0"
## e "0" "0"    "0"
```

```
S
```

```
##   y   x            e
## y "0" "0"          "0"
## x "0" "sigma[x]^2" "0"
## e "0" "0"          "sigma[varepsilon]^2"
```

```
u
```

```
##   u
## y "alpha"
## x "mu[x]"
## e "0"
```

```
Filter
```

```
##   y x e
## y 1 0 0
## x 0 1 0
```

The covariance expectations can be symbolically derived using the `ramR::C()` function with `A` of class `yac.symbol`.

```
ramR::C(Ryacas::ysym(A), S)
```

```
## {{sigma[x]^2*beta^2+sigma[varepsilon]^2,                          beta*sigma[x]^2,                sig
##  {                       sigma[x]^2*beta,                              sigma[x]^2,
##  {                    sigma[varepsilon]^2,                                     0,                sig
```

$$\mathbf{C} = \begin{pmatrix} \sigma_x^2\beta^2 + \sigma_\varepsilon^2 & \beta\sigma_x^2 & \sigma_\varepsilon^2 \\ \sigma_x^2\beta & \sigma_x^2 & 0 \\ \sigma_\varepsilon^2 & 0 & \sigma_\varepsilon^2 \end{pmatrix}$$

The covariance expectations for the observed variables can be symbolically derived using the `ramR::M()` function with `A` of class `yac.symbol`.

```
ramR::M(Ryacas::ysym(A), S, Filter)
```

```
## {{sigma[x]^2*beta^2+sigma[varepsilon]^2,                          beta*sigma[x]^2},
##  {                       sigma[x]^2*beta,                              sigma[x]^2}}
```

$$\mathbf{M} = \begin{pmatrix} \sigma_x^2\beta^2 + \sigma_\varepsilon^2 & \beta\sigma_x^2 \\ \sigma_x^2\beta & \sigma_x^2 \end{pmatrix}$$

The mean expectations can be symbolically derived using the `ramR::v()` function with `A` of class `yac.symbol`.

```
ramR::v(Ryacas::ysym(A), u)
```

```
## {{alpha+beta*mu[x]},
##  {           mu[x]},
##  {               0}}
```

$$\mathbf{v} = \begin{pmatrix} \alpha + \beta\mu_x \\ \mu_x \\ 0 \end{pmatrix}$$

The mean expectations for the observed variables can be symbolically derived using the `ramR::g()` function with `A` of class `yac.symbol`.

```
ramR::g(Ryacas::ysym(A), u, Filter)
```

```
## {{alpha+beta*mu[x]},
##  {           mu[x]}}
```

$$\mathbf{g} = \begin{pmatrix} \alpha + \beta\mu_x \\ \mu_x \end{pmatrix}$$

4.2 Numerical Example

Let df be a random sample from a population with the following parameters

Parameter	$x = 0$	$x = 1$
Sample Size	500	500
$\mathbb{E}(y \mid x)$	0	1
$\mathrm{Var}(y \mid x)$	1	1

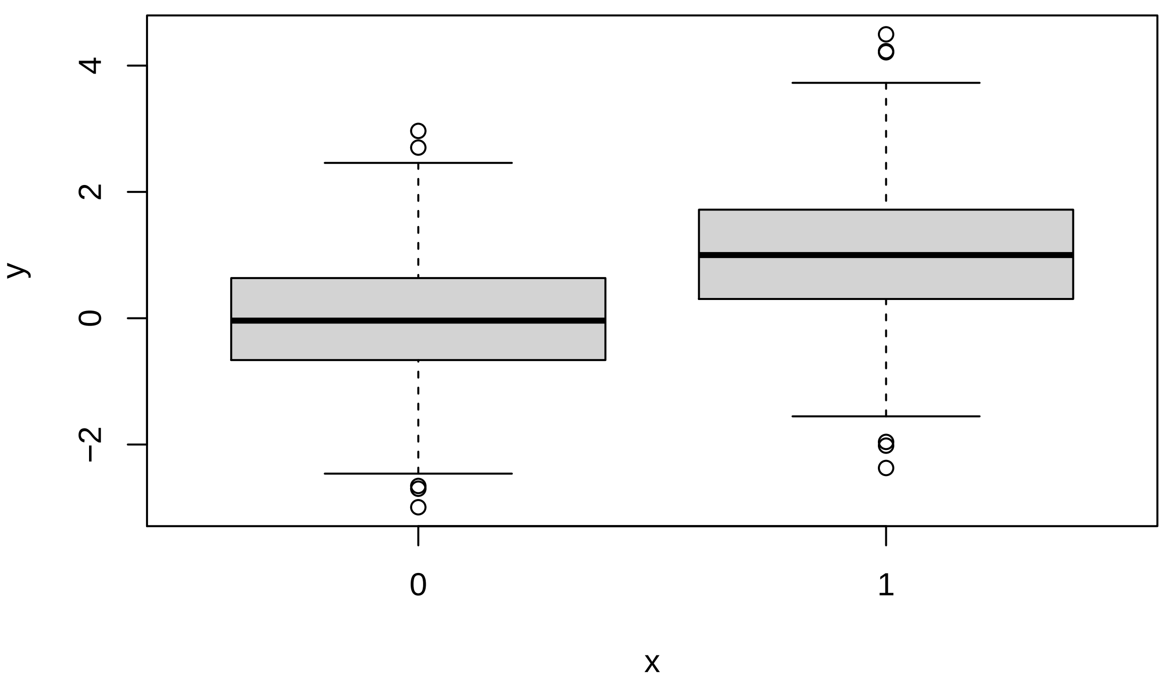
Parameter	Description	Value
α	$\mathbb{E}(y \mid x = 0)$	0
β	$\mathbb{E}(y \mid x = 1) - \mathbb{E}(y \mid x = 0)$	1

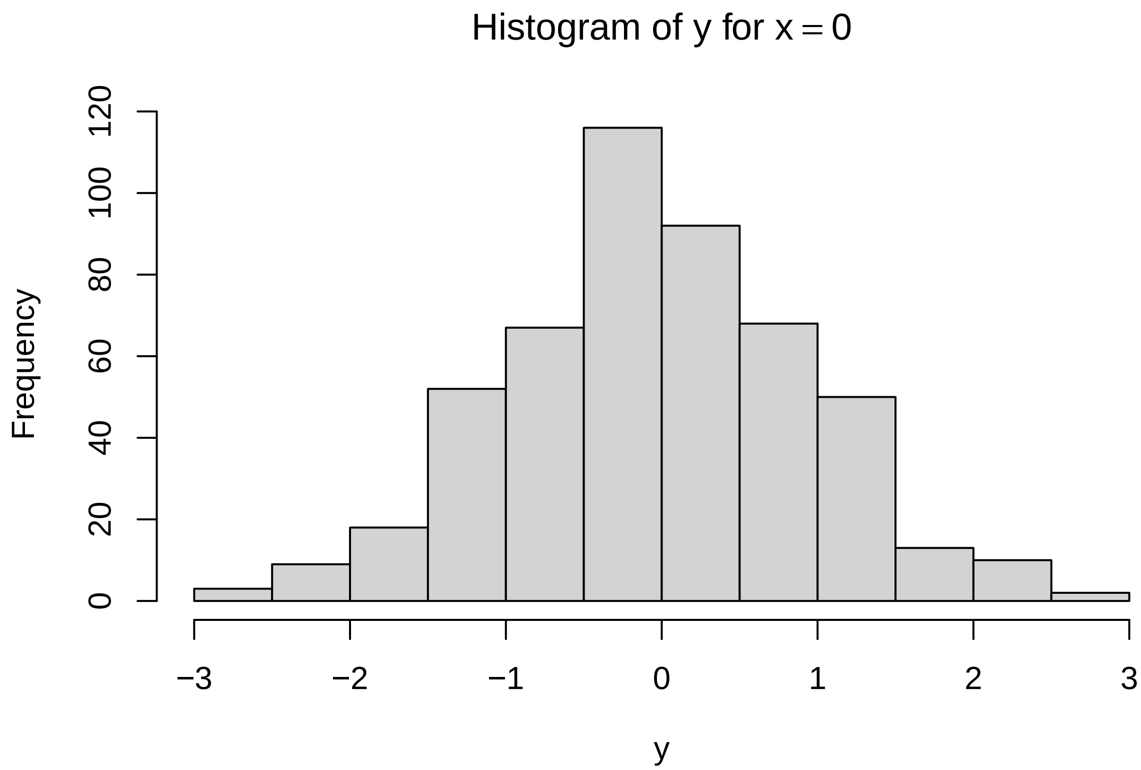
```
head(df)
```

```
##            y x
## 1  1.3709584 0
## 2 -0.5646982 0
## 3  0.3631284 0
## 4  0.6328626 0
## 5  0.4042683 0
## 6 -0.1061245 0
```

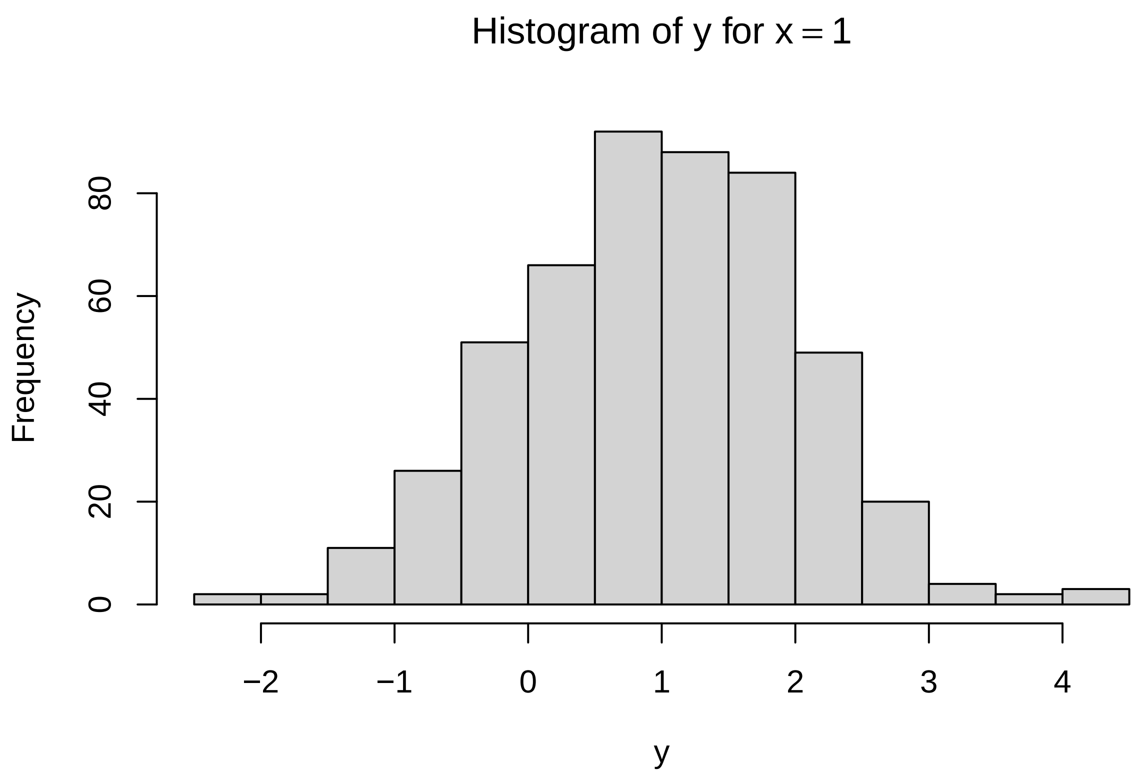
```
summary(df)
```

```
##        y                  x
##  Min.   :-2.9931   Min.   :0.0
##  1st Qu.:-0.2770   1st Qu.:0.0
##  Median : 0.4503   Median :0.5
##  Mean   : 0.4742   Mean   :0.5
##  3rd Qu.: 1.2492   3rd Qu.:1.0
##  Max.   : 4.4953   Max.   :1.0
```





Histogram of y for x = 0
Frequency
0
20
40
60
80
100
120
−3
−2
−1
0
1
2
3
y



Histogram of y for x = 1
Frequency
0
20
40
60
80
−2
−1
0
1
2
3
4
y

4.2.1 t-test

```r
t.test(y ~ x, data = df)
```

```
##
##  Welch Two Sample t-test
##
## data:  y by x
## t = -15.897, df = 994.36, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.1329278 -0.8839594
## sample estimates:
## mean in group 0 mean in group 1
##     -0.03004622      0.97839737
```

4.2.2 Linear Regression

```r
summary(lm(y ~ x, data = df))
```

```
##
## Call:
## lm(formula = y ~ x, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3501 -0.6517  0.0086  0.6858  3.5169
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.03005    0.04486   -0.67    0.503
## x            1.00844    0.06344   15.90   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.003 on 998 degrees of freedom
## Multiple R-squared:  0.2021, Adjusted R-squared:  0.2013
## F-statistic: 252.7 on 1 and 998 DF,  p-value: < 2.2e-16
```

4.2.3 Structural Equation Modeling

4.2.3.1 lavaan (Rosseel, 2012)

```r
model <- "
  y ~ x
"
fit <- lavaan::sem(
  model,
  data = df,
  meanstructure = TRUE,
```

```
  fixed.x = FALSE
)
lavaan::summary(fit)
```

```
## lavaan 0.6-7 ended normally after 12 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of free parameters                          5
##
##   Number of observations                          1000
##
## Model Test User Model:
##
##   Test statistic                                 0.000
##   Degrees of freedom                                 0
##
## Parameter Estimates:
##
##   Standard errors                             Standard
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##
## Regressions:
##                    Estimate  Std.Err  z-value  P(>|z|)
##   y ~
##     x                 1.008    0.063   15.913    0.000
##
## Intercepts:
##                    Estimate  Std.Err  z-value  P(>|z|)
##    .y                -0.030    0.045   -0.671    0.503
##     x                 0.500    0.016   31.623    0.000
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)
##    .y                 1.004    0.045   22.361    0.000
##     x                 0.250    0.011   22.361    0.000
```

4.2.3.2 OpenMx (Boker et al., 2020)

RAM matrices can be used to specify models in `OpenMx`. Note, however, that the `u` vector in the RAM notation is `M` in the `OpenMx` notation.

```
mxData <- OpenMx::mxData(
  observed = df,
  type = "raw"
)
mxA <- OpenMx::mxMatrix(
  type = "Full",
  nrow = 3,
  ncol = 3,
  free = c(
```

```
    F, T, F,
    F, F, F,
    F, F, F
  ),
  values = c(
    0, 0.20, 1,
    0, 0, 0,
    0, 0, 0
  ),
  labels = c(
    NA, "beta", NA,
    NA, NA, NA,
    NA, NA, NA
  ),
  byrow = TRUE,
  name = "mxA"
)
mxS <- OpenMx::mxMatrix(
  type = "Symm",
  nrow = 3,
  ncol = 3,
  free = c(
    F, F, F,
    F, T, F,
    F, F, T
  ),
  values = c(
    0, 0, 0,
    0, 0.20, 0,
    0, 0, 0.20
  ),
  labels = c(
    NA, NA, NA,
    NA, "sigma2x", NA,
    NA, NA, "sigma2e"
  ),
  byrow = TRUE,
  name = "mxS"
)
mxM <- OpenMx::mxMatrix(
  type = "Full",
  nrow = 1,
  ncol = 3,
  free = c(
    T, T, F
  ),
  values = c(
    0.20,
    0.20,
    0
  ),
  labels = c(
    "alpha",
```

```
    "mux",
    NA
  ),
  byrow = TRUE,
  name = "mxM"
)
mxF <- OpenMx::mxMatrix(
  type = "Full",
  nrow = 2,
  ncol = 3,
  free = FALSE,
  values = c(
    1, 0, 0,
    0, 1, 0
  ),
  byrow = TRUE,
  name = "mxF"
)
expRAM <- OpenMx::mxExpectationRAM(
  A = "mxA",
  S = "mxS",
  F = "mxF",
  M = "mxM",
  dimnames = c(
    "y",
    "x",
    "e"
  )
)
objML <- OpenMx::mxFitFunctionML()
mxMod <- OpenMx::mxModel(
  name = "Student's t test",
  data = mxData,
  matrices = list(
    mxA,
    mxS,
    mxF,
    mxM
  ),
  expectation = expRAM,
  fitfunction = objML
)
fit <- OpenMx::mxRun(mxMod)
```

```
## Running Student's t test with 5 parameters
```

```
summary(fit)
```

```
## Summary of Student's t test
##
## free parameters:
##      name matrix row col    Estimate  Std.Error A
```

```
## 1    beta    mxA   1   2  1.00844356 0.06337369
## 2 sigma2x    mxS   2   2  0.25000000 0.01118034
## 3 sigma2e    mxS   3   3  1.00402596 0.04490152
## 4   alpha    mxM   1   y -0.03004621 0.04481202
## 5     mux    mxM   1   x  0.49999999 0.01581140
##
## Model Statistics:
##                |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
##        Model:             5                   1995             4293.478
##    Saturated:             5                   1995                   NA
## Independence:             4                   1996                   NA
## Number of observations/statistics: 1000/2000
##
## Information Criteria:
##       |  df Penalty  |  Parameters Penalty  |  Sample-Size Adjusted
## AIC:       303.4776              4303.478                 4303.538
## BIC:     -9487.4941              4328.016                 4312.136
## CFI: NA
## TLI: 1   (also known as NNFI)
## RMSEA:  0  [95% CI (NA, NA)]
## Prob(RMSEA <= 0.05): NA
## To get additional fit indices, see help(mxRefModels)
## timestamp: 2021-02-14 03:27:22
## Wall clock time: 0.04273891 secs
## optimizer:  SLSQP
## OpenMx version number: 2.18.1
## Need help?  See help(mxSummary)
```

4.2.4 Using the `ramR` Package

```
A
```

```
##   y        x e
## y 0 1.008444 1
## x 0 0.000000 0
## e 0 0.000000 0
```

```
S
```

```
##   y         x        e
## y 0 0.0000000 0.000000
## x 0 0.2502503 0.000000
## e 0 0.0000000 1.006038
```

```
u
```

```
##             u
## y -0.03004622
## x  0.50000000
## e  0.00000000
```

```
Filter
```

```
##   y x e
## y 1 0 0
## x 0 1 0
```

The covariance expectations can be numerically derived using the `ramR::C()` function.

```
ramR::C(A, S)
```

```
##           y         x        e
## y 1.2605321 0.2523633 1.006038
## x 0.2523633 0.2502503 0.000000
## e 1.0060380 0.0000000 1.006038
```

The covariance expectations for the observed variables can be numerically derived using the `ramR::M()` function.

```
ramR::M(A, S, Filter)
```

```
##           y         x
## y 1.2605321 0.2523633
## x 0.2523633 0.2502503
```

The mean expectations can be numerically derived using the `ramR::v()` function.

```
ramR::v(A, u)
```

```
##           v
## y 0.4741756
## x 0.5000000
## e 0.0000000
```

The mean expectations for the observed variables can be numerically derived using the `ramR::g()` function.

```
ramR::g(A, u, Filter)
```

```
##           g
## y 0.4741756
## x 0.5000000
```

4.3 Equations to RAM

The `ramR` package has a utility function to convert structural equations to RAM notation. The Student's *t*-test can be expressed in the following equations

```
eq <- "
  # LHS OPERATION RHS LABEL
  e     by        y   1
  y     on        x   beta
  e     with      e   sigma[varepsilon]^2
  x     with      x   sigma[x]^2
  y     on        1   alpha
  x     on        1   mu[x]
"
```

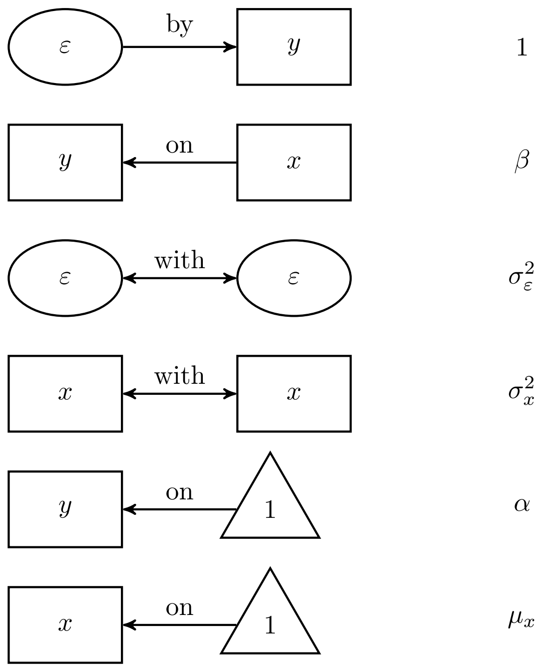


Figure 4.2: Student's t-test's Structural Equations

The error term is treated as a latent variable and defined with the operation `by`. Its value is constrained to 1. The regression of y on x is defined by operation `on`. It is labeled as `beta`. The variance of x and the error variance are defined using the operation `with`. These are labeled `sigma[x]^2` and `sigma[varepsilon]^2` respectively. The intercept and the mean of x are defined using the operation `on 1`. These are labeled `alpha` and `mu[x]` respectively.

The `ramR::Eq2RAM` converts the equations to RAM notation.

```
ramR::Eq2RAM(eq)
```

```
## $par.table
##   lhs   op rhs           par.label par.index
## 1   e   by   y                   1         1
## 2   y   on   x                beta        p1
## 3   e with   e sigma[varepsilon]^2        p2
## 4   x with   x          sigma[x]^2        p3
## 5   y   on   1               alpha        p4
## 6   x   on   1               mu[x]        p5
##
```

```
## $variables
## [1] "y" "x" "e"
##
## $g.variables
## [1] "y" "x"
##
## $h.variables
## [1] "e"
##
## $A
##   y   x      e
## y "0" "beta" "1"
## x "0" "0"    "0"
## e "0" "0"    "0"
##
## $S
##   y   x            e
## y "0" "0"          "0"
## x "0" "sigma[x]^2" "0"
## e "0" "0"          "sigma[varepsilon]^2"
##
## $u
##   u
## y "alpha"
## x "mu[x]"
## e "0"
##
## $Filter
##   y x e
## y 1 0 0
## x 0 1 0
```

4.4 Equations to Expectations

The ramR package has a utility function to convert structural equations to expectations both symbolically and numerically.

```
eq <- "
  # LHS OPERATION RHS LABEL
  e     by        y   1
  y     on        x   beta
  e     with      e   sigma[varepsilon]^2
  x     with      x   sigma[x]^2
  y     on        1   alpha
  x     on        1   mu[x]
"
```

```
ramR::Eq2Expectations(eq)
```

```
## $par.table
##   lhs   op rhs           par.label par.index
## 1   e   by   y                   1         1
```

```
## 2   y   on   x              beta        p1
## 3   e with   e sigma[varepsilon]^2        p2
## 4   x with   x          sigma[x]^2        p3
## 5   y   on   1               alpha        p4
## 6   x   on   1               mu[x]        p5
##
## $variables
## [1] "y" "x" "e"
##
## $g.variables
## [1] "y" "x"
##
## $h.variables
## [1] "e"
##
## $A
## {{   0, beta,    1},
##  {   0,    0,    0},
##  {   0,    0,    0}}
##
## $S
## {{                  0,                   0,                   0},
##  {                  0,          sigma[x]^2,                   0},
##  {                  0,                   0, sigma[varepsilon]^2}}
##
## $u
## {{alpha},
##  {mu[x]},
##  {    0}}
##
## $Filter
## {{1, 0, 0},
##  {0, 1, 0}}
##
## $v
## {{alpha+beta*mu[x]},
##  {           mu[x]},
##  {               0}}
##
## $g
## {{alpha+beta*mu[x]},
##  {           mu[x]}}
##
## $C
## {{sigma[x]^2*beta^2+sigma[varepsilon]^2,                       beta*sigma[x]^2,                   si
##  {                     sigma[x]^2*beta,                            sigma[x]^2,
##  {                 sigma[varepsilon]^2,                                     0,                   si
##
## $M
## {{sigma[x]^2*beta^2+sigma[varepsilon]^2,                       beta*sigma[x]^2},
##  {                     sigma[x]^2*beta,                            sigma[x]^2}}
```

```
eq <- "
  # LHS OPERATION RHS VALUE
```

```
  e     by        y   1.00
  y     on        x   1.00
  e     with      e   1.00
  x     with      x   0.25
  y     on        1   0.00
  x     on        1   0.50
"
```

```
ramR::Eq2Expectations(eq)
```

```
## $par.table
##   lhs   op rhs par.label par.index
## 1   e   by   y      1.00      1.00
## 2   y   on   x      1.00      1.00
## 3   e with   e      1.00      1.00
## 4   x with   x      0.25      0.25
## 5   y   on   1      0.00      0.00
## 6   x   on   1      0.50      0.50
##
## $variables
## [1] "y" "x" "e"
##
## $g.variables
## [1] "y" "x"
##
## $h.variables
## [1] "e"
##
## $A
##   y x e
## y 0 1 1
## x 0 0 0
## e 0 0 0
##
## $S
##   y    x e
## y 0 0.00 0
## x 0 0.25 0
## e 0 0.00 1
##
## $u
##     u
## y 0.0
## x 0.5
## e 0.0
##
## $Filter
##   y x e
## y 1 0 0
## x 0 1 0
##
## $v
##     v
```

```
## y 0.5
## x 0.5
## e 0.0
##
## $g
##     g
## y 0.5
## x 0.5
##
## $C
##      y    x e
## y 1.25 0.25 1
## x 0.25 0.25 0
## e 1.00 0.00 1
##
## $M
##      y    x
## y 1.25 0.25
## x 0.25 0.25
```

Chapter 5

One-Way Analysis of Variance

In this section, one-way analysis of variance is presented as a structural equation model using the RAM notation. Let y be a continuous dependent variable, x be a categorical independent variable with three levels ($x = \{0, 1, 2\}$). The dependent variable x can be dummy coded as

x	x_1	x_2
$x = 0$	0	0
$x = 1$	1	0
$x = 2$	0	1

ε is the stochastic error term with mean 0 and constant variance of σ^2_{ε} across the values of the regressors. The associations of the variables are given by

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \varepsilon$$

where

- β_0 is the expected value of y when $x = 0$
- β_1 is the unit change in y for unit change in x_1 while x_2 is constant
- β_2 is the unit change in y for unit change in x_2 while x_1 is constant
- $\beta_0 + \beta_1$ is the expected value of y when $x = 1$
- $\beta_0 + \beta_2$ is the expected value of y when $x = 2$

5.1 Symbolic

Let $\{y, x_1, x_2, \varepsilon\}$ be the variables of interest.

$$\mathbf{A} = \begin{pmatrix} 0 & \beta_1 & \beta_2 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\mathbf{S} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \sigma^2_{x_1} & 0 & 0 \\ 0 & 0 & \sigma^2_{x_2} & 0 \\ 0 & 0 & 0 & \sigma^2_{\varepsilon} \end{pmatrix}$$

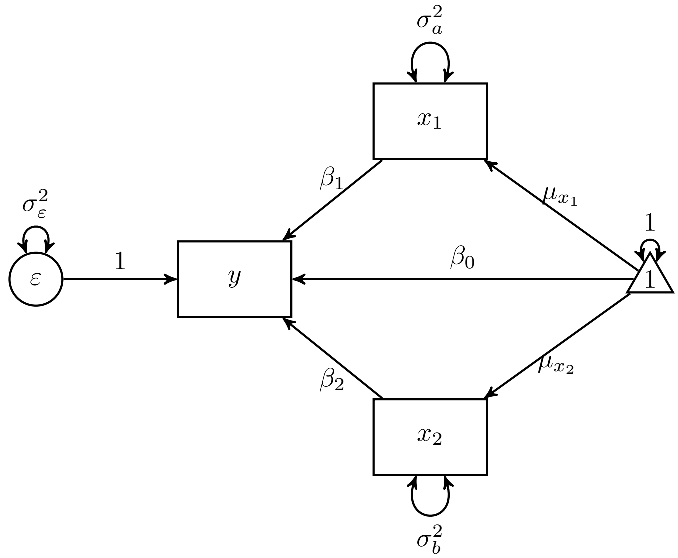


Figure 5.1: One-Way Analysis of Variance

$$
\begin{aligned}
\mathbf{C} &= (\mathbf{I}-\mathbf{A})^{-1}\,\mathbf{S}\left[(\mathbf{I}-\mathbf{A})^{-1}\right]^{\mathsf{T}} \\
&= \mathbf{E}\mathbf{S}\mathbf{E}^{\mathsf{T}} \\
&= \begin{pmatrix} 1 & \beta_1 & \beta_2 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}
\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \sigma_{x_1}^2 & 0 & 0 \\ 0 & 0 & \sigma_{x_2}^2 & 0 \\ 0 & 0 & 0 & \sigma_\varepsilon^2 \end{pmatrix}
\begin{pmatrix} 1 & \beta_1 & \beta_2 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}^{\mathsf{T}} \\
&= \begin{pmatrix} \sigma_{x_1}^2\beta_1^2 + \sigma_{x_2}^2\beta_2^2 + \sigma_\varepsilon^2 & \beta_1\sigma_{x_1}^2 & \beta_2\sigma_{x_2}^2 & \sigma_\varepsilon^2 \\ \sigma_{x_1}^2\beta_1 & \sigma_{x_1}^2 & 0 & 0 \\ \sigma_{x_2}^2\beta_2 & 0 & \sigma_{x_2}^2 & 0 \\ \sigma_\varepsilon^2 & 0 & 0 & \sigma_\varepsilon^2 \end{pmatrix}
\end{aligned}
$$

$$
\mathbf{F} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}
$$

$$
\begin{aligned}
\mathbf{M} &= \mathbf{F}\left(\mathbf{I}-\mathbf{A}\right)^{-1}\mathbf{S}\left[\left(\mathbf{I}-\mathbf{A}\right)^{-1}\right]^{\mathsf{T}}\mathbf{F}^{\mathsf{T}} \\
&= \mathbf{FESE}^{\mathsf{T}}\mathbf{F}^{\mathsf{T}} \\
&= \mathbf{FCF}^{\mathsf{T}} \\
&= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}
\begin{pmatrix} \sigma^2_{x_1}\beta_1^2 + \sigma^2_{x_2}\beta_2^2 + \sigma^2_{\varepsilon} & \beta_1\sigma^2_{x_1} & \beta_2\sigma^2_{x_2} & \sigma^2_{\varepsilon} \\ \sigma^2_{x_1}\beta_1 & \sigma^2_{x_1} & 0 & 0 \\ \sigma^2_{x_2}\beta_2 & 0 & \sigma^2_{x_2} & 0 \\ \sigma^2_{\varepsilon} & 0 & 0 & \sigma^2_{\varepsilon} \end{pmatrix}
\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}^{\mathsf{T}} \\
&= \begin{pmatrix} \sigma^2_{x_1}\beta_1^2 + \sigma^2_{x_2}\beta_2^2 + \sigma^2_{\varepsilon} & \beta_1\sigma^2_{x_1} & \beta_2\sigma^2_{x_2} \\ \sigma^2_{x_1}\beta_1 & \sigma^2_{x_1} & 0 \\ \sigma^2_{x_2}\beta_2 & 0 & \sigma^2_{x_2} \end{pmatrix}
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{v} &= \left(\mathbf{I}-\mathbf{A}\right)^{-1}\mathbf{u} \\
&= \left[\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} 0 & \beta_1 & \beta_2 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}\right]^{-1} \begin{pmatrix} \beta_0 \\ \mu_{x_1} \\ \mu_{x_2} \\ 0 \end{pmatrix} \\
&= \begin{pmatrix} \beta_0 + \beta_1\mu_{x_1} + \beta_2\mu_{x_2} \\ \mu_{x_1} \\ \mu_{x_2} \\ 0 \end{pmatrix}
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{u} &= \left(\mathbf{I}-\mathbf{A}\right)\mathbf{v} \\
&= \left[\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} 0 & \beta_1 & \beta_2 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}\right] \begin{pmatrix} \beta_0 + \beta_1\mu_{x_1} + \beta_2\mu_{x_2} \\ \mu_{x_1} \\ \mu_{x_2} \\ 0 \end{pmatrix} \\
&= \begin{pmatrix} \beta_0 + \beta_1\mu_{x_1} + \beta_2\mu_{x_2} \\ \mu_{x_1} \\ \mu_{x_2} \\ 0 \end{pmatrix}
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{g} &= \mathbf{F}\left(\mathbf{I}-\mathbf{A}\right)^{-1}\mathbf{u} \\
&= \left[\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} 0 & \beta_1 & \beta_2 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}\right]^{-1} \begin{pmatrix} \beta_0 \\ \mu_{x_1} \\ \mu_{x_2} \\ 0 \end{pmatrix} \\
&= \begin{pmatrix} \beta_0 + \beta_1\mu_{x_1} + \beta_2\mu_{x_2} \\ \mu_{x_1} \\ \mu_{x_2} \end{pmatrix}
\end{aligned}
$$

5.1.1 Using the `ramR` Package

```
A
```

```
##    y   x1        x2        e
## y  "0" "beta[1]" "beta[2]" "1"
## x1 "0" "0"       "0"       "0"
## x2 "0" "0"       "0"       "0"
## e  "0" "0"       "0"       "0"
```

```
S
```

```
##    y   x1               x2               e
## y  "0" "0"              "0"              "0"
## x1 "0" "sigma[x[1]]^2" "0"              "0"
## x2 "0" "0"              "sigma[x[2]]^2" "0"
## e  "0" "0"              "0"              "sigma[varepsilon]^2"
```

```
u
```

```
##    u
## y  "beta[0]"
## x1 "mu[x[1]]"
## x2 "mu[x[2]]"
## e  "0"
```

```
Filter
```

```
##    y x1 x2 e
## y  1  0  0 0
## x1 0  1  0 0
## x2 0  0  1 0
```

The covariance expectations can be symbolically derived using the `ramR::C()` function with `A` of class `yac.symbol`.

```
ramR::C(Ryacas::ysym(A), S)
```

```
## {{sigma[x[1]]^2*beta[1]^2+sigma[x[2]]^2*beta[2]^2+sigma[varepsilon]^2,
##  {                                              sigma[x[1]]^2*beta[1],
##  {                                              sigma[x[2]]^2*beta[2],
##  {                                                sigma[varepsilon]^2,
```

$$
\mathbf{C} = \begin{pmatrix} \sigma_{x_1}^2\beta_1^2 + \sigma_{x_2}^2\beta_2^2 + \sigma_{\varepsilon}^2 & \beta_1\sigma_{x_1}^2 & \beta_2\sigma_{x_2}^2 & \sigma_{\varepsilon}^2 \\ \sigma_{x_1}^2\beta_1 & \sigma_{x_1}^2 & 0 & 0 \\ \sigma_{x_2}^2\beta_2 & 0 & \sigma_{x_2}^2 & 0 \\ \sigma_{\varepsilon}^2 & 0 & 0 & \sigma_{\varepsilon}^2 \end{pmatrix}
$$

The covariance expectations for the observed variables can be symbolically derived using the `ramR::M()` function with `A` of class `yac.symbol`.

```
ramR::M(Ryacas::ysym(A), S, Filter)
```

```
## {{sigma[x[1]]^2*beta[1]^2+sigma[x[2]]^2*beta[2]^2+sigma[varepsilon]^2,
##  {                                                sigma[x[1]]^2*beta[1],
##  {                                                sigma[x[2]]^2*beta[2],
```

$$\mathbf{M} = \begin{pmatrix} \sigma_{x_1}^2\beta_1^2 + \sigma_{x_2}^2\beta_2^2 + \sigma_{\varepsilon}^2 & \beta_1\sigma_{x_1}^2 & \beta_2\sigma_{x_2}^2 \\ \sigma_{x_1}^2\beta_1 & \sigma_{x_1}^2 & 0 \\ \sigma_{x_2}^2\beta_2 & 0 & \sigma_{x_2}^2 \end{pmatrix}$$

The mean expectations can be symbolically derived using the `ramR::v()` function with `A` of class `yac.symbol`.

```
ramR::v(Ryacas::ysym(A), u)
```

```
## {{beta[0]+beta[1]*mu[x[1]]+beta[2]*mu[x[2]]},
##  {                              mu[x[1]]},
##  {                              mu[x[2]]},
##  {                                     0}}
```

$$\mathbf{v} = \begin{pmatrix} \beta_0 + \beta_1\mu_{x_1} + \beta_2\mu_{x_2} \\ \mu_{x_1} \\ \mu_{x_2} \\ 0 \end{pmatrix}$$

The mean expectations for the observed variables can be symbolically derived using the `ramR::g()` function with `A` of class `yac.symbol`.

```
ramR::g(Ryacas::ysym(A), u, Filter)
```

```
## {{beta[0]+beta[1]*mu[x[1]]+beta[2]*mu[x[2]]},
##  {                              mu[x[1]]},
##  {                              mu[x[2]]}}
```

$$\mathbf{g} = \begin{pmatrix} \beta_0 + \beta_1\mu_{x_1} + \beta_2\mu_{x_2} \\ \mu_{x_1} \\ \mu_{x_2} \end{pmatrix}$$

5.2 Numerical Example

Let `df` be a random sample from a population with the following parameters

Parameter	$x = 0$	$x = 1$	$x = 2$
Sample Size	500	500	500
$\mathbb{E}(y \mid x)$	0	2	1
$\mathrm{Var}(y \mid x)$	1	1	1

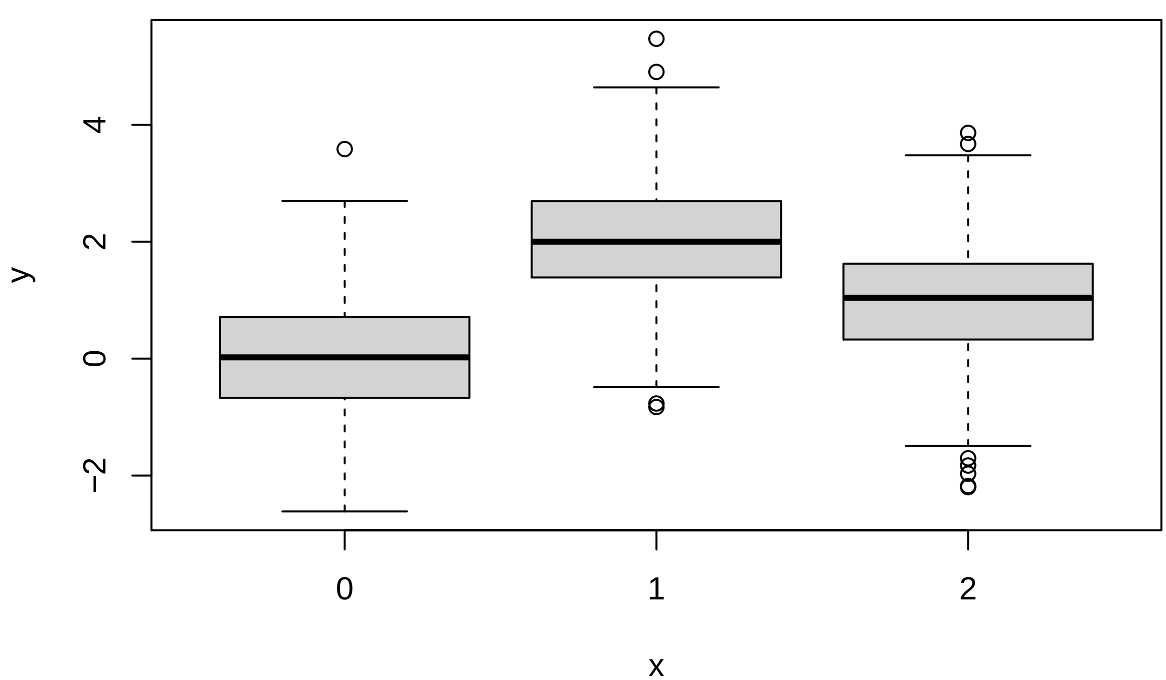
Parameter	Description	Value
β_0	$\mathbb{E}(y \mid x = 0)$	0
β_1	$\mathbb{E}(y \mid x = 1) - \mathbb{E}(y \mid x = 0)$	2
β_2	$\mathbb{E}(y \mid x = 2) - \mathbb{E}(y \mid x = 0)$	1

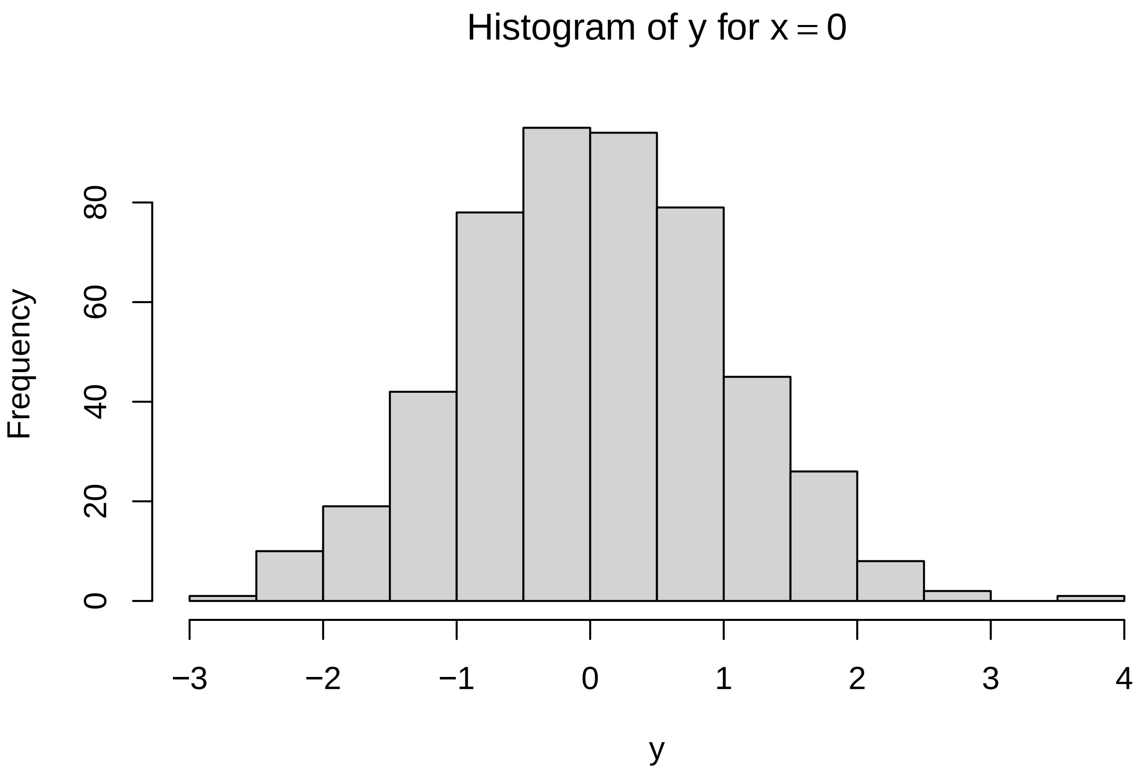
```
head(df)
```

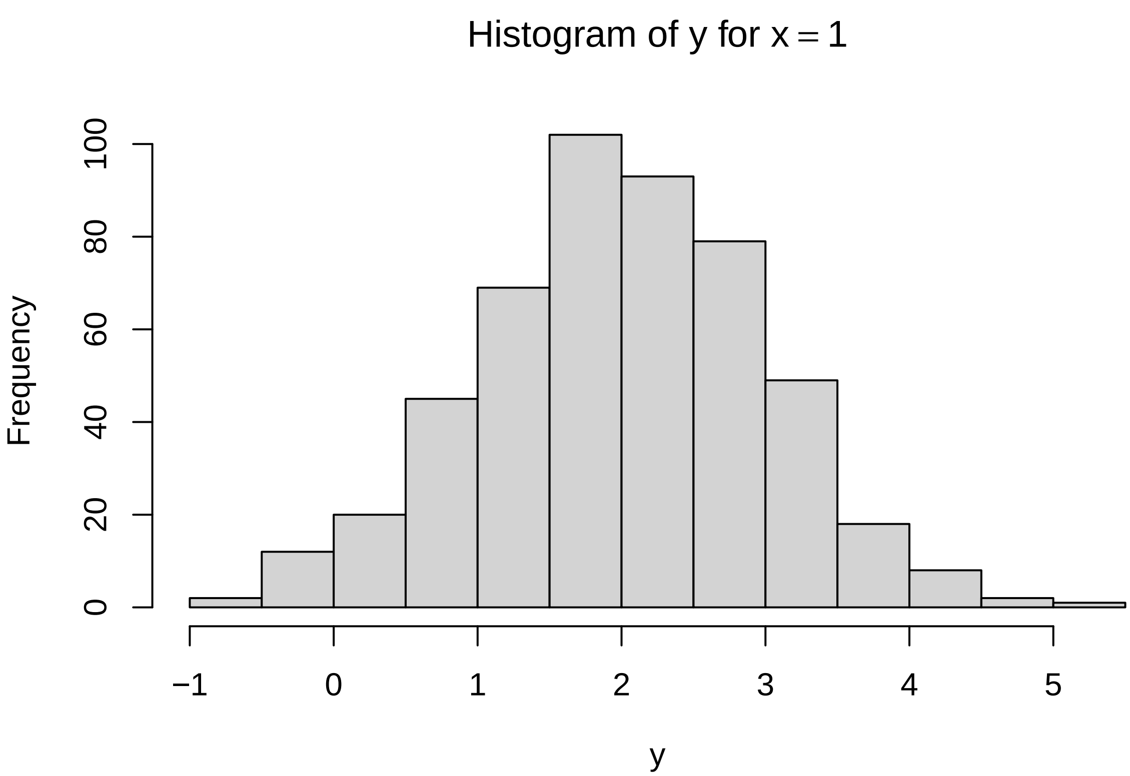
```
##            y x
## 1 -0.6013830 0
## 2 -0.1358161 0
## 3 -0.9872728 0
## 4  0.8319250 0
## 5 -0.7950595 0
## 6  0.3404646 0
```

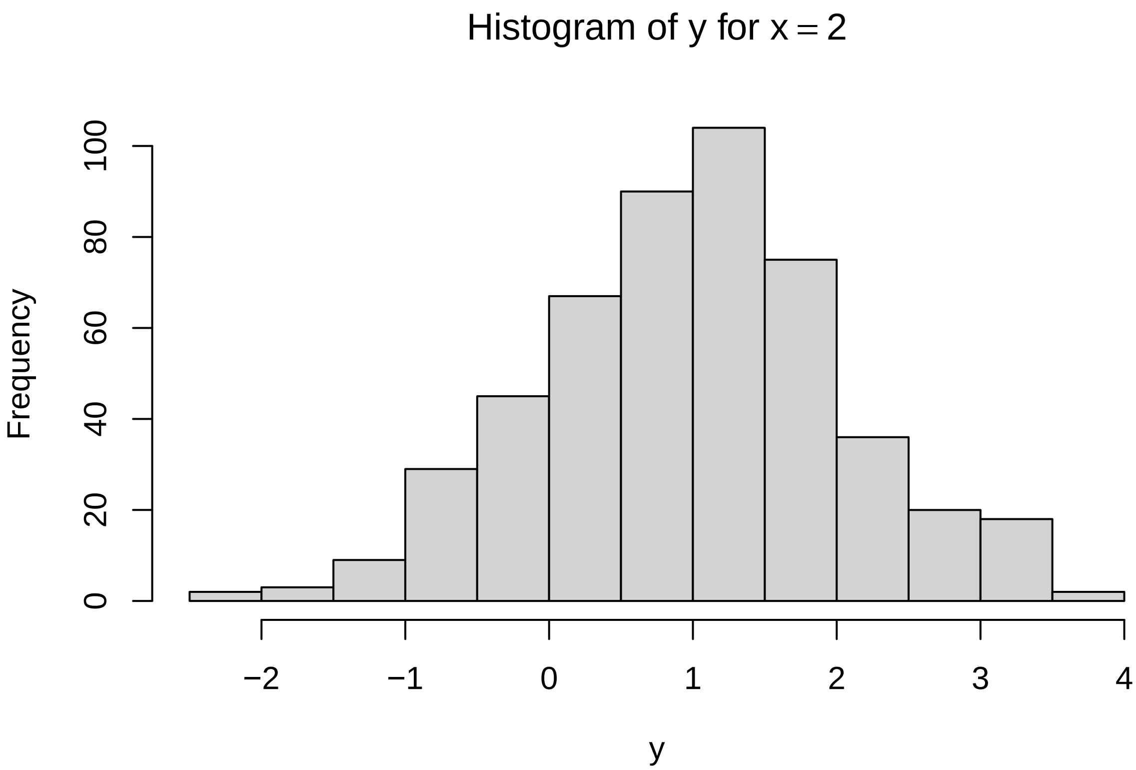
```
summary(df)
```

```
##        y             x
##  Min.   :-2.61364   0:500
##  1st Qu.: 0.08094   1:500
##  Median : 1.02617   2:500
##  Mean   : 1.00814
##  3rd Qu.: 1.90112
##  Max.   : 5.47091
```




Histogram of y for x = 0
Frequency
0
20
40
60
80
−3
−2
−1
0
1
2
3
4
y


Histogram of y for x = 1
Frequency
0
20
40
60
80
100
−1
0
1
2
3
4
5
y



5.2.1 One-Way Analysis of Variance

Make sure that x is of class `factor` for `lm` and `aov` to treat it as a categorical variable.

```
str(df)
```

```
## 'data.frame':    1500 obs. of  2 variables:
##  $ y: num  -0.601 -0.136 -0.987 0.832 -0.795 ...
##  $ x: Factor w/ 3 levels "0","1","2": 1 1 1 1 1 1 1 1 1 1 ...
```

```
summary(aov(y ~ x, data = df))
```

```
##               Df Sum Sq Mean Sq F value Pr(>F)
## x              2  983.8   491.9   471.4 <2e-16 ***
## Residuals   1497 1562.2     1.0
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

5.2.2 Linear Regression

```
summary(lm(y ~ x, data = df))
```

```
##
## Call:
```

```
## lm(formula = y ~ x, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.1792 -0.6469  0.0021  0.6751  3.5538
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.03083    0.04569   0.675      0.5
## x1           1.98309    0.06461  30.694   <2e-16 ***
## x2           0.94884    0.06461  14.686   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.022 on 1497 degrees of freedom
## Multiple R-squared:  0.3864, Adjusted R-squared:  0.3856
## F-statistic: 471.4 on 2 and 1497 DF,  p-value: < 2.2e-16
```

5.2.3 Structural Equation Modeling

We have to dummy code the data set first before fitting the model. The `model.matrix` function which is used to create a design matrix can be used to dummy code `x`. Make sure that `x` is a `factor`. The first column of the design matrix is a matrix of ones. Since we do not need this column, we can replace this column with the values of `y`. Make sure to name rename the first column as `lavaan` relies on the column names.

```
df_dummy <- model.matrix(y ~ x, data = df)
df_dummy[, 1] <- df$y
colnames(df_dummy)[1] <- "y"
head(df_dummy)
```

```
##            y x1 x2
## 1 -0.6013830  0  0
## 2 -0.1358161  0  0
## 3 -0.9872728  0  0
## 4  0.8319250  0  0
## 5 -0.7950595  0  0
## 6  0.3404646  0  0
```

5.2.3.1 lavaan (Rosseel, 2012)

```
model <- "
  y ~ x1 + x2
"
fit <- lavaan::sem(
  model,
  data = df_dummy,
  meanstructure = TRUE,
  fixed.x = FALSE
)
lavaan::summary(fit)
```

```
## lavaan 0.6-7 ended normally after 22 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of free parameters                          9
##
##   Number of observations                          1500
##
## Model Test User Model:
##
##   Test statistic                                 0.000
##   Degrees of freedom                                 0
##
## Parameter Estimates:
##
##   Standard errors                             Standard
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##
## Regressions:
##                    Estimate  Std.Err  z-value  P(>|z|)
##   y ~
##     x1                1.983    0.065   30.725    0.000
##     x2                0.949    0.065   14.701    0.000
##
## Covariances:
##                    Estimate  Std.Err  z-value  P(>|z|)
##   x1 ~~
##     x2               -0.111    0.006  -17.321    0.000
##
## Intercepts:
##                    Estimate  Std.Err  z-value  P(>|z|)
##    .y                 0.031    0.046    0.676    0.499
##     x1                0.333    0.012   27.386    0.000
##     x2                0.333    0.012   27.386    0.000
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)
##    .y                 1.041    0.038   27.386    0.000
##     x1                0.222    0.008   27.386    0.000
##     x2                0.222    0.008   27.386    0.000
```

5.2.3.2 OpenMx (Boker et al., 2020)

RAM matrices can be used to specify models in `OpenMx`. Note, however, that the `u` vector in the RAM notation is `M` in the `OpenMx` notation.

```
mxData <- OpenMx::mxData(
  observed = df_dummy,
  type = "raw"
)
mxA <- OpenMx::mxMatrix(
  type = "Full",
```

```
  nrow = 4,
  ncol = 4,
  free = c(
    F, T, T, F,
    F, F, F, F,
    F, F, F, F,
    F, F, F, F
  ),
  values = c(
    0, 0.20, 0.20, 1,
    0, 0, 0, 0,
    0, 0, 0, 0,
    0, 0, 0, 0
  ),
  labels = c(
    NA, "beta1", "beta2", NA,
    NA, NA, NA, NA,
    NA, NA, NA, NA,
    NA, NA, NA, NA
  ),
  byrow = TRUE,
  name = "mxA"
)
mxS <- OpenMx::mxMatrix(
  type = "Symm",
  nrow = 4,
  ncol = 4,
  free = c(
    F, F, F, F,
    F, T, F, F,
    F, F, T, F,
    F, F, F, T
  ),
  values = c(
    0,    0,    0,    0,
    0, 0.20,    0,    0,
    0,    0, 0.20,    0,
    0,    0,    0, 0.20
  ),
  labels = c(
    NA, NA, NA, NA,
    NA, "sigma2x1", NA, NA,
    NA, NA, "sigma2x2", NA,
    NA, NA, NA, "sigma2e"
  ),
  byrow = TRUE,
  name = "mxS"
)
mxM <- OpenMx::mxMatrix(
  type = "Full",
  nrow = 1,
  ncol = 4,
  free = c(
```

```
    T, T, T, F
  ),
  values = c(
    0.20,
    0.20,
    0.20,
    0
  ),
  labels = c(
    "beta0",
    "mux1",
    "mux2",
    NA
  ),
  byrow = TRUE,
  name = "mxM"
)
mxF <- OpenMx::mxMatrix(
  type = "Full",
  nrow = 3,
  ncol = 4,
  free = FALSE,
  values = c(
    1, 0, 0, 0,
    0, 1, 0, 0,
    0, 0, 1, 0
  ),
  byrow = TRUE,
  name = "mxF"
)
expRAM <- OpenMx::mxExpectationRAM(
  A = "mxA",
  S = "mxS",
  F = "mxF",
  M = "mxM",
  dimnames = c(
    "y",
    "x1",
    "x2",
    "e"
  )
)
objML <- OpenMx::mxFitFunctionML()
mxMod <- OpenMx::mxModel(
  name = "One Way Analysis of Variance",
  data = mxData,
  matrices = list(
    mxA,
    mxS,
    mxF,
    mxM
  ),
  expectation = expRAM,
```

```
  fitfunction = objML
)
fit <- OpenMx::mxRun(mxMod)
```

```
## Running One Way Analysis of Variance with 8 parameters
```

```
summary(fit)
```

```
## Summary of One Way Analysis of Variance
##
## free parameters:
##       name matrix row col   Estimate   Std.Error A
## 1    beta1    mxA   1   2 1.98308662 0.064543779
## 2    beta2    mxA   1   3 0.94883814 0.064543143
## 3 sigma2x1    mxS   2   2 0.22222230 0.008114416
## 4 sigma2x2    mxS   3   3 0.22222238 0.008114420
## 5  sigma2e    mxS   4   4 1.04147460 0.038029458
## 6    beta0    mxM   1   y 0.03083127 0.045639092
## 7     mux1    mxM   1  x1 0.33333343 0.012171613
## 8     mux2    mxM   1  x2 0.33333344 0.012171612
##
## Model Statistics:
##                |  Parameters  |  Degrees of Freedom  |  Fit (-2lnL units)
##        Model:             8                   4492              8319.17
##    Saturated:             9                   4491                   NA
## Independence:             6                   4494                   NA
## Number of observations/statistics: 1500/4500
##
## Information Criteria:
##       |  df Penalty  |  Parameters Penalty  |  Sample-Size Adjusted
## AIC:      -664.8302               8335.170                 8335.266
## BIC:    -24531.8162               8377.676                 8352.262
## To get additional fit indices, see help(mxRefModels)
## timestamp: 2021-02-14 03:27:24
## Wall clock time: 0.03658605 secs
## optimizer:  SLSQP
## OpenMx version number: 2.18.1
## Need help?  See help(mxSummary)
```

5.2.4 Using the ramR Package

```
A
```

```
##    y       x1        x2 e
## y  0 2.008444 0.9885797 1
## x1 0 0.000000 0.0000000 0
## x2 0 0.000000 0.0000000 0
## e  0 0.000000 0.0000000 0
```

```r
S
```

```
##    y        x1        x2         e
## y  0 0.0000000 0.0000000 0.0000000
## x1 0 0.2223705 0.0000000 0.0000000
## x2 0 0.0000000 0.2223705 0.0000000
## e  0 0.0000000 0.0000000 0.9823083
```

```r
u
```

```
##            u
## y  0.3333333
## x1 0.3333333
## x2 0.3333333
## e  0.0000000
```

```r
Filter
```

```
##    y x1 x2 e
## y  1  0  0 0
## x1 0  1  0 0
## x2 0  0  1 0
```

The covariance expectations can be numerically derived using the `ramR::C()` function.

```r
ramR::C(A, S)
```

```
##            y        x1        x2         e
## y  2.0966368 0.4466185 0.2198309 0.9823083
## x1 0.4466185 0.2223705 0.0000000 0.0000000
## x2 0.2198309 0.0000000 0.2223705 0.0000000
## e  0.9823083 0.0000000 0.0000000 0.9823083
```

The covariance expectations for the observed variables can be numerically derived using the `ramR::M()` function.

```r
ramR::M(A, S, Filter)
```

```
##            y        x1        x2
## y  2.0966368 0.4466185 0.2198309
## x1 0.4466185 0.2223705 0.0000000
## x2 0.2198309 0.0000000 0.2223705
```

The mean expectations can be numerically derived using the `ramR::v()` function.

```r
ramR::v(A, u)
```

```
##            v
## y  1.3323411
## x1 0.3333333
## x2 0.3333333
## e  0.0000000
```

The mean expectations for the observed variables can be numerically derived using the `ramR::v()` function.

```
ramR::g(A, u, Filter)
```

```
##            g
## y  1.3323411
## x1 0.3333333
## x2 0.3333333
```

5.3 Equations to RAM

The `ramR` package has a utility function to convert structural equations to RAM notation. One-way analysis of variance with three levels can be expressed in the following equations

```
eq <- "
  # VARIABLE1 OPERATION VARIABLE2 LABEL
  e           by        y         1
  y           on        x1        beta[1]
  y           on        x2        beta[2]
  e           with      e         sigma[varepsilon]^2
  x1          with      x1        sigma[x[1]]^2
  x2          with      x2        sigma[x[2]]^2
  y           on        1         beta[0]
  x1          on        1         mu[x[1]]
  x2          on        1         mu[x[2]]
"
```

The error term is treated as a latent variable and defined with the operation `by`. Its value is constrained to 1. The regression of y on x_1 and x_2 is defined by operation `on`. The coefficients are labeled as `beta[1]` and `beta[2]` respectively. The variance of x_1, x_2 and the error variance are defined using the operation `with`. These are labeled `sigma[x[1]]^2`, `sigma[x[2]]^2`, and `sigma[varepsilon]^2` respectively. The intercept and the mean of x_1 and x_2 are defined using the operation `on 1`. These are labeled `beta[0]`, `mu[x[1]]`, and `mu[x[2]]` respectively.

The `ramR::Eq2RAM` converts the equations to RAM notation.

```
ramR::Eq2RAM(eq)
```

```
## $par.table
##   lhs   op rhs           par.label par.index
## 1   e   by   y                   1         1
## 2   y   on  x1             beta[1]        p1
## 3   y   on  x2             beta[2]        p2
## 4   e with   e sigma[varepsilon]^2        p3
## 5  x1 with  x1       sigma[x[1]]^2        p4
## 6  x2 with  x2       sigma[x[2]]^2        p5
## 7   y   on   1             beta[0]        p6
## 8  x1   on   1            mu[x[1]]        p7
## 9  x2   on   1            mu[x[2]]        p8
##
## $variables
```

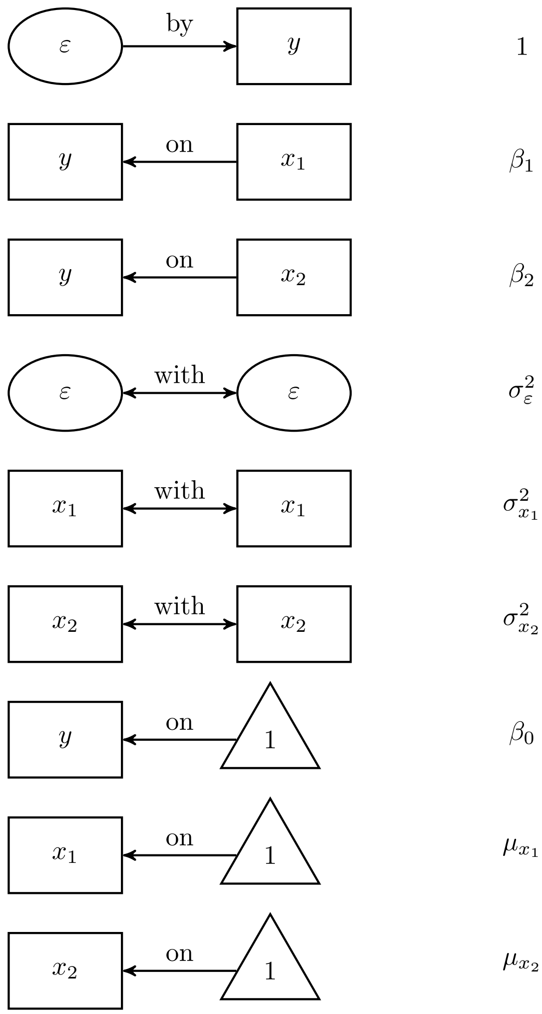


Figure 5.2: One-Way Analysis of Variance's Structural Equations

```
## [1] "y"  "x1" "x2" "e"
##
## $g.variables
## [1] "y"  "x1" "x2"
##
## $h.variables
## [1] "e"
##
## $A
##    y   x1        x2        e
## y  "0" "beta[1]" "beta[2]" "1"
## x1 "0" "0"       "0"       "0"
## x2 "0" "0"       "0"       "0"
## e  "0" "0"       "0"       "0"
##
## $S
##    y   x1              x2              e
## y  "0" "0"             "0"             "0"
## x1 "0" "sigma[x[1]]^2" "0"             "0"
## x2 "0" "0"             "sigma[x[2]]^2" "0"
## e  "0" "0"             "0"             "sigma[varepsilon]^2"
##
## $u
##    u
## y  "beta[0]"
## x1 "mu[x[1]]"
## x2 "mu[x[2]]"
## e  "0"
##
## $Filter
##    y x1 x2 e
## y  1  0  0 0
## x1 0  1  0 0
## x2 0  0  1 0
```

5.4 Equations to Expectations

The `ramR` package has a utility function to convert structural equations to expectations both symbolically and numerically.

```
eq <- "
  # VARIABLE1 OPERATION VARIABLE2 LABEL
  e         by        y         1
  y         on        x1        beta[1]
  y         on        x2        beta[2]
  e         with      e         sigma[varepsilon]^2
  x1        with      x1        sigma[x[1]]^2
  x2        with      x2        sigma[x[2]]^2
  y         on        1         beta[0]
  x1        on        1         mu[x[1]]
  x2        on        1         mu[x[2]]
"
```

```
ramR::Eq2Expectations(eq)
```

```
## $par.table
##   lhs   op rhs           par.label par.index
## 1   e   by   y                   1         1
## 2   y   on  x1             beta[1]        p1
## 3   y   on  x2             beta[2]        p2
## 4   e with   e sigma[varepsilon]^2        p3
## 5  x1 with  x1       sigma[x[1]]^2        p4
## 6  x2 with  x2       sigma[x[2]]^2        p5
## 7   y   on   1             beta[0]        p6
## 8  x1   on   1            mu[x[1]]        p7
## 9  x2   on   1            mu[x[2]]        p8
##
## $variables
## [1] "y"  "x1" "x2" "e"
##
## $g.variables
## [1] "y"  "x1" "x2"
##
## $h.variables
## [1] "e"
##
## $A
## {{      0, beta[1], beta[2],       1},
##  {      0,       0,       0,       0},
##  {      0,       0,       0,       0},
##  {      0,       0,       0,       0}}
##
## $S
## {{                  0,                   0,                   0,                   0},
##  {                  0,       sigma[x[1]]^2,                   0,                   0},
##  {                  0,                   0,       sigma[x[2]]^2,                   0},
##  {                  0,                   0,                   0, sigma[varepsilon]^2}}
##
## $u
## {{ beta[0]},
##  {mu[x[1]]},
##  {mu[x[2]]},
##  {       0}}
##
## $Filter
## {{1, 0, 0, 0},
##  {0, 1, 0, 0},
##  {0, 0, 1, 0}}
##
## $v
## {{beta[0]+beta[1]*mu[x[1]]+beta[2]*mu[x[2]]},
##  {                                mu[x[1]]},
##  {                                mu[x[2]]},
##  {                                       0}}
##
## $g
```

```
## {{beta[0]+beta[1]*mu[x[1]]+beta[2]*mu[x[2]]},
##  {                                mu[x[1]]},
##  {                                mu[x[2]]}}
##
## $C
## {{sigma[x[1]]^2*beta[1]^2+sigma[x[2]]^2*beta[2]^2+sigma[varepsilon]^2,
##  {                                              sigma[x[1]]^2*beta[1],
##  {                                              sigma[x[2]]^2*beta[2],
##  {                                                sigma[varepsilon]^2,
##
## $M
## {{sigma[x[1]]^2*beta[1]^2+sigma[x[2]]^2*beta[2]^2+sigma[varepsilon]^2,
##  {                                              sigma[x[1]]^2*beta[1],
##  {                                              sigma[x[2]]^2*beta[2],
```

```
eq <- "
  # VARIABLE1 OPERATION VARIABLE2 LABEL
  e           by        y         1
  y           on        x1        2
  y           on        x2        1
  e           with      e         1
  x1          with      x1        0.22222222222
  x2          with      x2        0.22222222222
  y           on        1         0
  x1          on        1         0.33333333333
  x2          on        1         0.33333333333
"
```

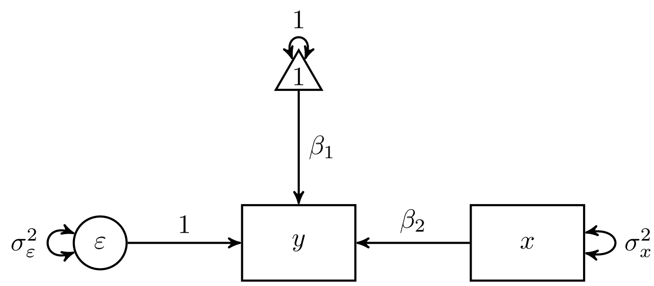
```
ramR::Eq2Expectations(eq)
```

```
## $par.table
##   lhs   op rhs par.label par.index
## 1   e   by   y 1.0000000 1.0000000
## 2   y   on  x1 2.0000000 2.0000000
## 3   y   on  x2 1.0000000 1.0000000
## 4   e with   e 1.0000000 1.0000000
## 5  x1 with  x1 0.2222222 0.2222222
## 6  x2 with  x2 0.2222222 0.2222222
## 7   y   on   1 0.0000000 0.0000000
## 8  x1   on   1 0.3333333 0.3333333
## 9  x2   on   1 0.3333333 0.3333333
##
## $variables
## [1] "y"  "x1" "x2" "e"
##
## $g.variables
## [1] "y"  "x1" "x2"
##
## $h.variables
## [1] "e"
##
## $A
##    y x1 x2 e
```

```
## y  0  2  1 1
## x1 0  0  0 0
## x2 0  0  0 0
## e  0  0  0 0
##
## $S
##    y        x1        x2 e
## y  0 0.0000000 0.0000000 0
## x1 0 0.2222222 0.0000000 0
## x2 0 0.0000000 0.2222222 0
## e  0 0.0000000 0.0000000 1
##
## $u
##            u
## y  0.0000000
## x1 0.3333333
## x2 0.3333333
## e  0.0000000
##
## $Filter
##    y x1 x2 e
## y  1  0  0 0
## x1 0  1  0 0
## x2 0  0  1 0
##
## $v
##            v
## y  1.0000000
## x1 0.3333333
## x2 0.3333333
## e  0.0000000
##
## $g
##            g
## y  1.0000000
## x1 0.3333333
## x2 0.3333333
##
## $C
##            y        x1        x2 e
## y  2.1111111 0.4444444 0.2222222 1
## x1 0.4444444 0.2222222 0.0000000 0
## x2 0.2222222 0.0000000 0.2222222 0
## e  1.0000000 0.0000000 0.0000000 1
##
## $M
##            y        x1        x2
## y  2.1111111 0.4444444 0.2222222
## x1 0.4444444 0.2222222 0.0000000
## x2 0.2222222 0.0000000 0.2222222
```

Chapter 6

Two-Variable Regression Model



$$y = \alpha + \beta x + \varepsilon$$

Figure 6.1: Two-Variable Regression Model

Chapter 7

k-Variable Regression Model

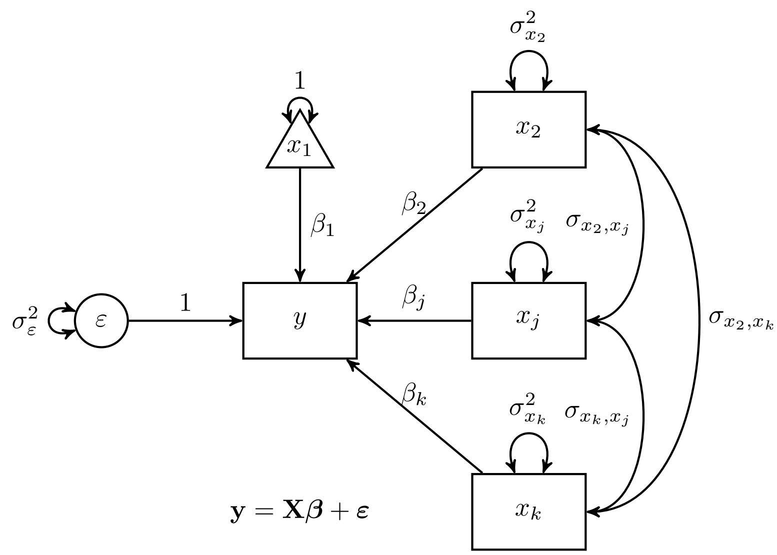


Figure 7.1: k-Variable Regression Model

Chapter 8

The Simple Mediation Model

Let y, m, x, ε_y, and ε_m be random variables whose associations are given by

$$y = \beta_0 + \beta_1 x + \beta_2 m + \varepsilon_y \tag{8.1}$$

$$m = \alpha_0 + \alpha_1 x + \varepsilon_m \tag{8.2}$$

or combined

$$\begin{aligned} y &= \beta_0 + \beta_1 x + \beta_2(\alpha_0 + \alpha_1 x + \varepsilon_m) + \varepsilon_y \\ &= \beta_0 + \beta_1 x + \beta_2 \alpha_0 + \alpha_1 \beta_2 x + \beta_2 \varepsilon_m + \varepsilon_y \end{aligned} \tag{8.3}$$

where

- β_1 is the path from x on y
- β_2 is the path from m to y
- α_1 is the path from x to m
- ε_y and ε_m are uncorrelated error terms with means of zero and constant variances of $\sigma^2_{\varepsilon_y}$ and $\sigma^2_{\varepsilon_m}$ respectively
- β_0 and α_0 are intercepts

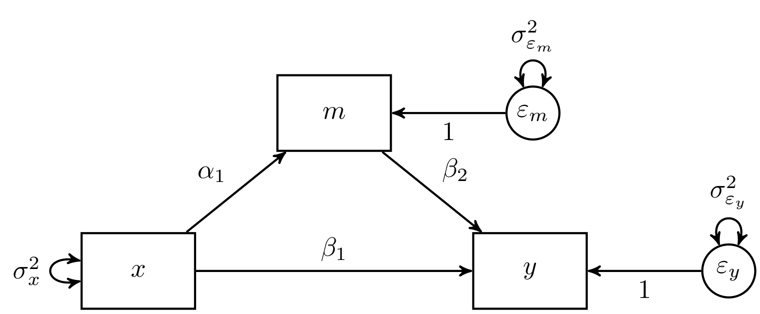


Figure 8.1: The Simple Mediation Model

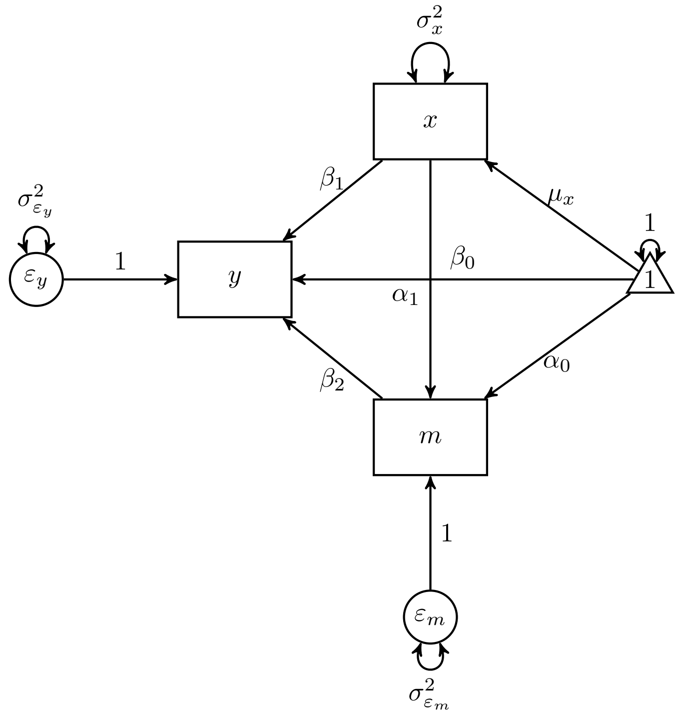


Figure 8.2: The Simple Mediation Model with Mean Structure

8.1 Symbolic

Let $\{y, m, x, \varepsilon_y, \varepsilon_m\}$ be the variables of interest.

$$\mathbf{A} = \begin{pmatrix} 0 & \beta_2 & \beta_1 & 1 & 0 \\ 0 & 0 & \alpha_1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\mathbf{S} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma^2_x & 0 & 0 \\ 0 & 0 & 0 & \sigma^2_{\varepsilon_y} & 0 \\ 0 & 0 & 0 & 0 & \sigma^2_{\varepsilon_m} \end{pmatrix}$$

$$
\begin{aligned}
\mathbf{C} &= (\mathbf{I}-\mathbf{A})^{-1}\,\mathbf{S}\left[(\mathbf{I}-\mathbf{A})^{-1}\right]^{\mathsf{T}} \\
&= \mathbf{E}\mathbf{S}\mathbf{E}^{\mathsf{T}} \\
&= \begin{pmatrix} 1 & \beta_2 & \beta_1+\beta_2\alpha_1 & 1 & \beta_2 \\ 0 & 1 & \alpha_1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}
\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_x^2 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{\varepsilon_y}^2 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{\varepsilon_m}^2 \end{pmatrix}
\begin{pmatrix} 1 & \beta_2 & \beta_1+\beta_2\alpha_1 & 1 & \beta_2 \\ 0 & 1 & \alpha_1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}^{\mathsf{T}} \\
&= \begin{pmatrix}
\sigma_x^2\beta_1^2 + 2\sigma_x^2\beta_1\beta_2\alpha_1 + \sigma_x^2\beta_2^2\alpha_1^2 + \beta_2^2\sigma_{\varepsilon_m}^2 + \sigma_{\varepsilon_y}^2 & \beta_1\alpha_1\sigma_x^2 + \beta_2\alpha_1^2\sigma_x^2 + \beta_2\sigma_{\varepsilon_m}^2 & \beta_1\sigma_x^2 + \beta_2\alpha_1\sigma_x^2 & \sigma_{\varepsilon_y}^2 & \beta_2\sigma_{\varepsilon_m}^2 \\
\alpha_1^2\sigma_x^2\beta_2 + \alpha_1\sigma_x^2\beta_1 + \beta_2\sigma_{\varepsilon_m}^2 & \sigma_x^2\alpha_1^2 + \sigma_{\varepsilon_m}^2 & \alpha_1\sigma_x^2 & 0 & \sigma_{\varepsilon_m}^2 \\
\sigma_x^2\beta_1 + \sigma_x^2\beta_2\alpha_1 & \sigma_x^2\alpha_1 & \sigma_x^2 & 0 & 0 \\
\sigma_{\varepsilon_y}^2 & 0 & 0 & \sigma_{\varepsilon_y}^2 & 0 \\
\sigma_{\varepsilon_m}^2\beta_2 & \sigma_{\varepsilon_m}^2 & 0 & 0 & \sigma_{\varepsilon_m}^2
\end{pmatrix}
\end{aligned}
$$

$$
\mathbf{F} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}
$$

$$
\begin{aligned}
\mathbf{M} &= \mathbf{F}\,(\mathbf{I}-\mathbf{A})^{-1}\,\mathbf{S}\left[(\mathbf{I}-\mathbf{A})^{-1}\right]^{\mathsf{T}}\mathbf{F}^{\mathsf{T}} \\
&= \mathbf{F}\mathbf{E}\mathbf{S}\mathbf{E}^{\mathsf{T}}\mathbf{F}^{\mathsf{T}} \\
&= \mathbf{F}\mathbf{C}\mathbf{F}^{\mathsf{T}} \\
&= \begin{pmatrix}
\sigma_x^2\beta_1^2 + 2\sigma_x^2\beta_1\beta_2\alpha_1 + \sigma_x^2\beta_2^2\alpha_1^2 + \beta_2^2\sigma_{\varepsilon_m}^2 + \sigma_{\varepsilon_y}^2 & \beta_1\alpha_1\sigma_x^2 + \beta_2\alpha_1^2\sigma_x^2 + \beta_2\sigma_{\varepsilon_m}^2 & \beta_1\sigma_x^2 + \beta_2\alpha_1\sigma_x^2 \\
\alpha_1^2\sigma_x^2\beta_2 + \alpha_1\sigma_x^2\beta_1 + \beta_2\sigma_{\varepsilon_m}^2 & \sigma_x^2\alpha_1^2 + \sigma_{\varepsilon_m}^2 & \alpha_1\sigma_x^2 \\
\sigma_x^2\beta_1 + \sigma_x^2\beta_2\alpha_1 & \sigma_x^2\alpha_1 & \sigma_x^2
\end{pmatrix}
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{v} &= (\mathbf{I}-\mathbf{A})^{-1}\,\mathbf{u} \\
&= \left[\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} 0 & \beta_2 & \beta_1 & 1 & 0 \\ 0 & 0 & \alpha_1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}\right]^{-1} \begin{pmatrix} \beta_0 \\ \alpha_0 \\ \mu_x \\ 0 \\ 0 \end{pmatrix} \\
&= \begin{pmatrix} \beta_0 + \beta_2\alpha_0 + \beta_2\alpha_1\mu_x + \beta_1\mu_x \\ \alpha_0 + \alpha_1\mu_x \\ \mu_x \\ 0 \\ 0 \end{pmatrix}
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{u} &= \left(\mathbf{I} - \mathbf{A}\right) \mathbf{v} \\
&= \left[\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} 0 & \beta_2 & \beta_1 & 1 & 0 \\ 0 & 0 & \alpha_1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \right] \begin{pmatrix} \beta_0 + \beta_1 \alpha_0 + \beta_1 \alpha_1 \mu_x + \beta_2 \mu_x \\ \alpha_0 + \alpha_1 \mu_x \\ \mu_x \\ 0 \\ 0 \end{pmatrix} \\
&= \begin{pmatrix} \beta_0 + \beta_1 \alpha_0 + \beta_1 \alpha_1 \mu_x + \mu_x \beta_2 \\ \alpha_0 + \alpha_1 \mu_x \\ \mu_x \\ 0 \\ 0 \end{pmatrix}
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{g} &= \mathbf{F} \left(\mathbf{I} - \mathbf{A}\right)^{-1} \mathbf{u} \\
&= \left[\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} 0 & \beta_2 & \beta_1 & 1 & 0 \\ 0 & 0 & \alpha_1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \right]^{-1} \begin{pmatrix} \beta_0 \\ \alpha_0 \\ \mu_x \\ 0 \\ 0 \end{pmatrix} \\
&= \begin{pmatrix} \beta_0 + \beta_2 \alpha_0 + \beta_2 \alpha_1 \mu_x + \beta_1 \mu_x \\ \alpha_0 + \alpha_1 \mu_x \\ \mu_x \end{pmatrix}
\end{aligned}
$$

8.1.1 Using the `ramR` Package

```
A
```

```
##    y   m        x          ey  em
## y  "0" "beta[2]" "beta[1]"  "1" "0"
## m  "0" "0"       "alpha[1]" "0" "1"
## x  "0" "0"       "0"        "0" "0"
## ey "0" "0"       "0"        "0" "0"
## em "0" "0"       "0"        "0" "0"
```

```
S
```

```
##    y   m   x            ey                         em
## y  "0" "0" "0"          "0"                        "0"
## m  "0" "0" "0"          "0"                        "0"
## x  "0" "0" "sigma[x]^2" "0"                        "0"
## ey "0" "0" "0"          "sigma[varepsilon[y]]^2" "0"
## em "0" "0" "0"          "0"                        "sigma[varepsilon[m]]^2"
```

```
u
```

```
##    u
## y  "beta[0]"
```

```
## m  "alpha[0]"
## x  "mu[x]"
## ey "0"
## em "0"
```

```
filter
```

```
##   y m x ey em
## y 1 0 0  0  0
## m 0 1 0  0  0
## x 0 0 1  0  0
```

The covariance expectations can be symbolically derived using the `ramR::C()` function with `A` of class `yac.symbol`.

```
ramR::C(Ryacas::ysym(A), S, simplify = TRUE)
```

```
## {{sigma[x]^2*beta[1]^2+2*sigma[x]^2*beta[1]*beta[2]*alpha[1]+sigma[x]^2*beta[2]^2*alpha[1]^2+beta[2]
##  {                                                             alpha[1]^2*sigma[x]^2*beta[2]+alpha[1]*s
##  {
##  {
##  {
```

$$\mathbf{C} = \begin{pmatrix} \sigma_x^2\beta_1^2 + 2\sigma_x^2\beta_1\beta_2\alpha_1 + \sigma_x^2\beta_2^2\alpha_1^2 + \beta_2^2\sigma_{\varepsilon_m}^2 + \sigma_{\varepsilon_y}^2 & \beta_1\alpha_1\sigma_x^2 + \beta_2\alpha_1^2\sigma_x^2 + \beta_2\sigma_{\varepsilon_m}^2 & \beta_1\sigma_x^2 + \beta_2\alpha_1\sigma_x^2 & \sigma_{\varepsilon_y}^2 & \beta_2\sigma_{\varepsilon_m}^2 \\ \alpha_1^2\sigma_x^2\beta_2 + \alpha_1\sigma_x^2\beta_1 + \beta_2\sigma_{\varepsilon_m}^2 & \sigma_x^2\alpha_1^2 + \sigma_{\varepsilon_m}^2 & \alpha_1\sigma_x^2 & 0 & \sigma_{\varepsilon_m}^2 \\ \sigma_x^2\beta_1 + \sigma_x^2\beta_2\alpha_1 & \sigma_x^2\alpha_1 & \sigma_x^2 & 0 & 0 \\ \sigma_{\varepsilon_y}^2 & 0 & 0 & \sigma_{\varepsilon_y}^2 & 0 \\ \sigma_{\varepsilon_m}^2\beta_2 & \sigma_{\varepsilon_m}^2 & 0 & 0 & \sigma_{\varepsilon_m}^2 \end{pmatrix}$$

The covariance expectations for the observed variables can be symbolically derived using the `ramR::M()` function with `A` of class `yac.symbol`.

```
ramR::M(Ryacas::ysym(A), S, filter, simplify = TRUE)
```

```
## {{sigma[x]^2*beta[1]^2+2*sigma[x]^2*beta[1]*beta[2]*alpha[1]+sigma[x]^2*beta[2]^2*alpha[1]^2+beta[2]
##  {                                                             alpha[1]^2*sigma[x]^2*beta[2]+alpha[1]*s
##  {
```

$$\mathbf{M} = \begin{pmatrix} \sigma_x^2\beta_1^2 + 2\sigma_x^2\beta_1\beta_2\alpha_1 + \sigma_x^2\beta_2^2\alpha_1^2 + \beta_2^2\sigma_{\varepsilon_m}^2 + \sigma_{\varepsilon_y}^2 & \beta_1\alpha_1\sigma_x^2 + \beta_2\alpha_1^2\sigma_x^2 + \beta_2\sigma_{\varepsilon_m}^2 & \beta_1\sigma_x^2 + \beta_2\alpha_1\sigma_x^2 \\ \alpha_1^2\sigma_x^2\beta_2 + \alpha_1\sigma_x^2\beta_1 + \beta_2\sigma_{\varepsilon_m}^2 & \sigma_x^2\alpha_1^2 + \sigma_{\varepsilon_m}^2 & \alpha_1\sigma_x^2 \\ \sigma_x^2\beta_1 + \sigma_x^2\beta_2\alpha_1 & \sigma_x^2\alpha_1 & \sigma_x^2 \end{pmatrix}$$

The mean expectations can be symbolically derived using the `ramR::v()` function with `A` of class `yac.symbol`.

```
ramR::v(Ryacas::ysym(A), u, simplify = TRUE)
```

```
## {{beta[0]+beta[2]*alpha[0]+beta[2]*alpha[1]*mu[x]+beta[1]*mu[x]},
##  {                                        alpha[0]+alpha[1]*mu[x]},
##  {                                                          mu[x]},
##  {                                                              0},
##  {                                                              0}}
```

$$\mathbf{v} = \begin{pmatrix} \beta_0 + \beta_2\alpha_0 + \beta_2\alpha_1\mu_x + \beta_1\mu_x \\ \alpha_0 + \alpha_1\mu_x \\ \mu_x \\ 0 \\ 0 \end{pmatrix}$$

The mean expectations for the observed variables can be symbolically derived using the `ramR::g()` function with `A` of class `yac.symbol`.

```
ramR::g(Ryacas::ysym(A), u, filter, simplify = TRUE)
```

```
## {{beta[0]+beta[2]*alpha[0]+beta[2]*alpha[1]*mu[x]+beta[1]*mu[x]},
##  {                                        alpha[0]+alpha[1]*mu[x]},
##  {                                                          mu[x]}}
```

$$\mathbf{g} = \begin{pmatrix} \beta_0 + \beta_2\alpha_0 + \beta_2\alpha_1\mu_x + \beta_1\mu_x \\ \alpha_0 + \alpha_1\mu_x \\ \mu_x \end{pmatrix}$$

8.2 Numerical Example

Let `df` be a random sample from a population with the following parameters

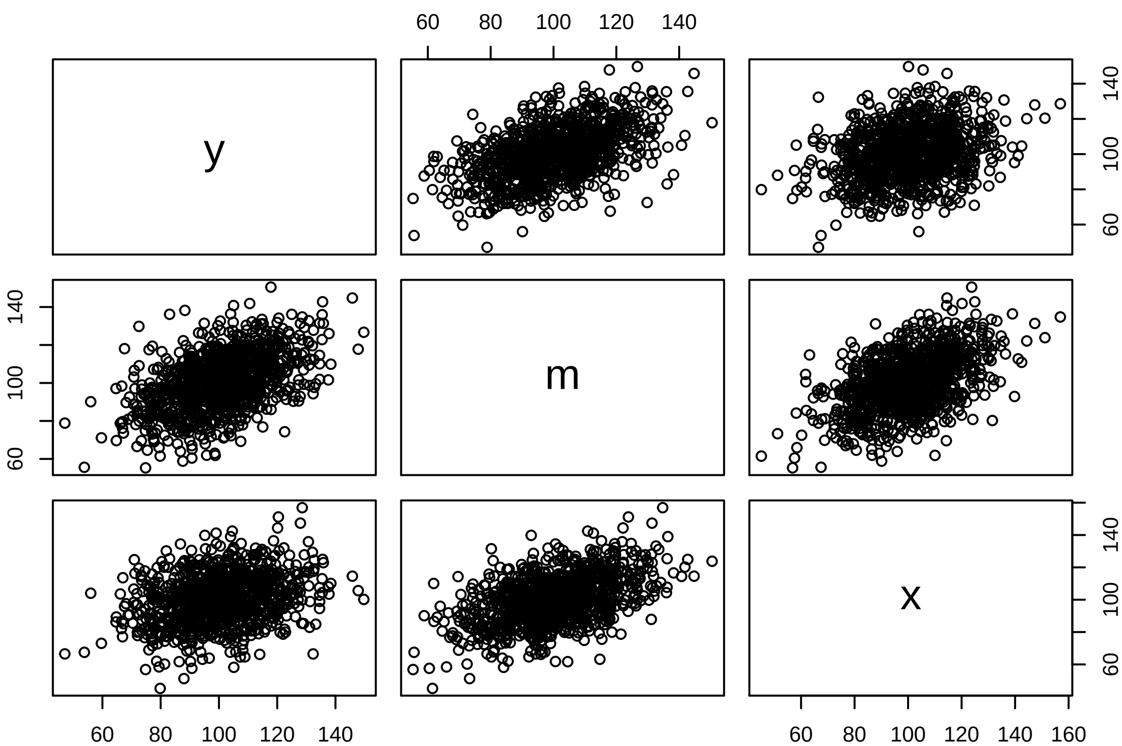
Parameter	Value
β_1	0
β_2	0.5
α_1	0.5
$\sigma^2_{\varepsilon_y}$	168.75
$\sigma^2_{\varepsilon_m}$	168.75
σ^2_x	225
β_0	50
α_0	50
μ_x	100

```
head(df)
```

```
##           y         m         x
## 1 107.32121  80.22215  64.60716
## 2 109.83608 109.42737 100.20734
## 3  97.41131 107.89265  79.57780
## 4  87.75538 106.43004  80.82994
## 5  80.77481 104.18033  99.07128
## 6  97.27327  98.70669 108.24974
```

```
summary(df)
```

```
##        y                m                x         
##  Min.   : 47.09   Min.   : 55.30   Min.   : 45.16  
##  1st Qu.: 90.02   1st Qu.: 90.39   1st Qu.: 90.09  
##  Median :100.34   Median : 99.84   Median :100.14  
##  Mean   :100.22   Mean   :100.36   Mean   :100.32  
##  3rd Qu.:110.30   3rd Qu.:110.40   3rd Qu.:110.29  
##  Max.   :149.73   Max.   :150.48   Max.   :156.91
```



8.2.1 Linear Regression

```
summary(lm(y ~ x + m, data = df))
```

```
## 
## Call:
## lm(formula = y ~ x + m, data = df)
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -42.413  -8.837  -0.045   8.804  39.148 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 48.47917    3.19559  15.171   <2e-16 ***
## x            0.04486    0.03101   1.447    0.148
```

```
## m            0.47072    0.03136  15.012   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.98 on 997 degrees of freedom
## Multiple R-squared:  0.2464, Adjusted R-squared:  0.2449
## F-statistic:   163 on 2 and 997 DF,  p-value: < 2.2e-16
```

```
summary(lm(m ~ x, data = df))
```

```
##
## Call:
## lm(formula = m ~ x, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -43.182  -9.426   0.280   8.705  38.824
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 52.04456    2.77350   18.77   <2e-16 ***
## x            0.48163    0.02734   17.62   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.1 on 998 degrees of freedom
## Multiple R-squared:  0.2372, Adjusted R-squared:  0.2365
## F-statistic: 310.4 on 1 and 998 DF,  p-value: < 2.2e-16
```

8.2.2 Structural Equation Modeling

8.2.2.1 lavaan (Rosseel, 2012)

```
# Covariance Structure
model <- "
  y ~ beta1 * x + beta2 * m
  m ~ alpha1 * x
  indirect := alpha1 * beta2
"
# With Mean Structure
model <- "
  y ~ beta1 * x + beta2 * m
  m ~ alpha1 * x
  y ~~ sigma2ey * y
  m ~~ sigma2em * m
  x ~~ sigma2x * x
  y ~ beta0 * 1
  m ~ alpha0 * 1
  x ~ mux * 1
  indirect := alpha1 * beta2
"
```

```
fit <- lavaan::sem(
  model,
  data = df,
  meanstructure = TRUE,
  fixed.x = FALSE
)
lavaan::summary(fit)
```

```
## lavaan 0.6-7 ended normally after 28 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of free parameters                          9
##
##   Number of observations                          1000
##
## Model Test User Model:
##
##   Test statistic                                 0.000
##   Degrees of freedom                                 0
##
## Parameter Estimates:
##
##   Standard errors                             Standard
##   Information                                 Expected
##   Information saturated (h1) model          Structured
##
## Regressions:
##                    Estimate  Std.Err  z-value  P(>|z|)
##   y ~
##     x       (bet1)    0.045    0.031    1.449    0.147
##     m       (bet2)    0.471    0.031   15.034    0.000
##   m ~
##     x       (alp1)    0.482    0.027   17.636    0.000
##
## Intercepts:
##                    Estimate  Std.Err  z-value  P(>|z|)
##    .y       (bet0)   48.479    3.191   15.193    0.000
##    .m       (alp0)   52.045    2.771   18.784    0.000
##     x        (mux)  100.320    0.479  209.300    0.000
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)
##    .y      (sgm2y)  167.964    7.512   22.361    0.000
##    .m      (sgm2m)  171.335    7.662   22.361    0.000
##     x      (sgm2x)  229.740   10.274   22.361    0.000
##
## Defined Parameters:
##                    Estimate  Std.Err  z-value  P(>|z|)
##     indirect          0.227    0.020   11.441    0.000
```

8.2.3 Using the `ramR` Package

```
A
```

```
##     y         m          x ey em
## y   0 0.4707226 0.04486201  1  0
## m   0 0.0000000 0.48162865  0  1
## x   0 0.0000000 0.00000000  0  0
## ey  0 0.0000000 0.00000000  0  0
## em  0 0.0000000 0.00000000  0  0
```

```
S
```

```
##     y m        x      ey       em
## y   0 0   0.0000   0.000   0.0000
## m   0 0   0.0000   0.000   0.0000
## x   0 0 229.9701   0.000   0.0000
## ey  0 0   0.0000 168.469   0.0000
## em  0 0   0.0000   0.000 171.6784
```

```
u
```

```
##             u
## y    48.47917
## m    52.04456
## x   100.31982
## ey    0.00000
## em    0.00000
```

```
filter
```

```
##   y m x ey em
## y 1 0 0  0  0
## m 0 1 0  0  0
## x 0 0 1  0  0
```

The covariance expectations can be numerically derived using the `ramR::C()` function.

```
ramR::C(A, S)
```

```
##             y        m         x      ey        em
## y   223.47047 110.8927  62.45425 168.469  80.81291
## m   110.89267 225.0237 110.76020   0.000 171.67843
## x    62.45425 110.7602 229.97013   0.000   0.00000
## ey  168.46896   0.0000   0.00000 168.469   0.00000
## em   80.81291 171.6784   0.00000   0.000 171.67843
```

The covariance expectations for the observed variables can be numerically derived using the `ramR::M()` function.

```
ramR::M(A, S, filter)
```

```
##           y        m         x
## y 223.47047 110.8927  62.45425
## m 110.89267 225.0237 110.76020
## x  62.45425 110.7602 229.97013
```

The mean expectations can be numerically derived using the `ramR::v()` function.

```
ramR::v(A, u)
```

```
##           v
## y  100.2221
## m  100.3615
## x  100.3198
## ey   0.0000
## em   0.0000
```

The mean expectations for the observed variables can be numerically derived using the `ramR::v()` function.

```
ramR::g(A, u, filter)
```

```
##          g
## y 100.2221
## m 100.3615
## x 100.3198
```

8.3 Equations to RAM

The `ramR` package has a utility function to convert structural equations to RAM notation. The simple mediation model can be expressed in the following equations

```
eq <- "
  # VARIABLE1 OPERATION VARIABLE2 LABEL
  ey          by        y         1
  em          by        m         1
  y           on        x         beta[1]
  y           on        m         beta[2]
  m           on        x         alpha[1]
  ey          with      ey        sigma[varepsilon[y]]^2
  em          with      em        sigma[varepsilon[m]]^2
  x           with      x         sigma[x]^2
  y           on        1         beta[0]
  m           on        1         alpha[0]
  x           on        1         mu[x]
"
```

The `ramR::Eq2RAM` converts the equations to RAM notation.

```
ramR::Eq2RAM(eq)
```

```
## $par.table
##    lhs   op rhs              par.label par.index
## 1   ey   by   y                      1         1
## 2   em   by   m                      1         1
## 3    y   on   x                beta[1]        p1
## 4    y   on   m                beta[2]        p2
## 5    m   on   x               alpha[1]        p3
## 6   ey with  ey sigma[varepsilon[y]]^2        p4
## 7   em with  em sigma[varepsilon[m]]^2        p5
## 8    x with   x             sigma[x]^2        p6
## 9    y   on   1                beta[0]        p7
## 10   m   on   1               alpha[0]        p8
## 11   x   on   1                  mu[x]        p9
##
## $variables
## [1] "y"  "m"  "x"  "ey" "em"
##
## $g.variables
## [1] "y" "m" "x"
##
## $h.variables
## [1] "ey" "em"
##
## $A
##    y   m         x          ey  em
## y  "0" "beta[2]" "beta[1]"  "1" "0"
## m  "0" "0"       "alpha[1]" "0" "1"
## x  "0" "0"       "0"        "0" "0"
## ey "0" "0"       "0"        "0" "0"
## em "0" "0"       "0"        "0" "0"
##
## $S
##    y   m   x            ey                       em
## y  "0" "0" "0"          "0"                      "0"
## m  "0" "0" "0"          "0"                      "0"
## x  "0" "0" "sigma[x]^2" "0"                      "0"
## ey "0" "0" "0"          "sigma[varepsilon[y]]^2" "0"
## em "0" "0" "0"          "0"                      "sigma[varepsilon[m]]^2"
##
## $u
##    u
## y  "beta[0]"
## m  "alpha[0]"
## x  "mu[x]"
## ey "0"
## em "0"
##
## $Filter
##   y m x ey em
## y 1 0 0  0  0
## m 0 1 0  0  0
```

```
## x 0 0 1  0  0
```

8.4 Equations to Expectations

The `ramR` package has a utility function to convert structural equations to expectations both symbolically and numerically.

```
eq <- "
  # VARIABLE1 OPERATION VARIABLE2 LABEL
  ey          by        y         1
  em          by        m         1
  y           on        x         beta[1]
  y           on        m         beta[2]
  m           on        x         alpha[1]
  ey          with      ey        sigma[varepsilon[y]]^2
  em          with      em        sigma[varepsilon[m]]^2
  x           with      x         sigma[x]^2
  y           on        1         beta[0]
  m           on        1         alpha[0]
  x           on        1         mu[x]
"
```

```
ramR::Eq2Expectations(eq)
```

```
## $par.table
##    lhs   op rhs              par.label par.index
## 1   ey   by   y                      1         1
## 2   em   by   m                      1         1
## 3    y   on   x                beta[1]        p1
## 4    y   on   m                beta[2]        p2
## 5    m   on   x               alpha[1]        p3
## 6   ey with  ey sigma[varepsilon[y]]^2        p4
## 7   em with  em sigma[varepsilon[m]]^2        p5
## 8    x with   x             sigma[x]^2        p6
## 9    y   on   1                beta[0]        p7
## 10   m   on   1               alpha[0]        p8
## 11   x   on   1                  mu[x]        p9
##
## $variables
## [1] "y"  "m"  "x"  "ey" "em"
##
## $g.variables
## [1] "y" "m" "x"
##
## $h.variables
## [1] "ey" "em"
##
## $A
## {{       0,  beta[2],  beta[1],        1,        0},
##  {       0,        0, alpha[1],        0,        1},
##  {       0,        0,        0,        0,        0},
##  {       0,        0,        0,        0,        0},
```

```
##  {       0,       0,       0,       0,       0}}
##
## $S
## {{                    0,                    0,                    0,                    0,
##  {                    0,                    0,                    0,                    0,
##  {                    0,                    0,           sigma[x]^2,                    0,
##  {                    0,                    0,                    0, sigma[varepsilon[y]]^2,
##  {                    0,                    0,                    0,                    0, sig
##
## $u
## {{ beta[0]},
##  {alpha[0]},
##  {   mu[x]},
##  {       0},
##  {       0}}
##
## $Filter
## {{1, 0, 0, 0, 0},
##  {0, 1, 0, 0, 0},
##  {0, 0, 1, 0, 0}}
##
## $v
## {{beta[0]+beta[2]*alpha[0]+(beta[1]+beta[2]*alpha[1])*mu[x]},
##  {                                  alpha[0]+alpha[1]*mu[x]},
##  {                                                    mu[x]},
##  {                                                        0},
##  {                                                        0}}
##
## $g
## {{beta[0]+beta[2]*alpha[0]+(beta[1]+beta[2]*alpha[1])*mu[x]},
##  {                                  alpha[0]+alpha[1]*mu[x]},
##  {                                                    mu[x]}}
##
## $C
## {{sigma[x]^2*(beta[1]+beta[2]*alpha[1])^2+sigma[varepsilon[y]]^2+sigma[varepsilon[m]]^2*beta[2]^2,
##  {                  alpha[1]*sigma[x]^2*(beta[1]+beta[2]*alpha[1])+sigma[varepsilon[m]]^2*beta[2],
##  {                                                        sigma[x]^2*(beta[1]+beta[2]*alpha[1]),
##  {                                                                       sigma[varepsilon[y]]^2,
##  {                                                                sigma[varepsilon[m]]^2*beta[2],
##
## $M
## {{sigma[x]^2*(beta[1]+beta[2]*alpha[1])^2+sigma[varepsilon[y]]^2+sigma[varepsilon[m]]^2*beta[2]^2,
##  {                  alpha[1]*sigma[x]^2*(beta[1]+beta[2]*alpha[1])+sigma[varepsilon[m]]^2*beta[2],
##  {                                                        sigma[x]^2*(beta[1]+beta[2]*alpha[1]),
```

```r
eq <- "
  # VARIABLE1 OPERATION VARIABLE2 LABEL
  ey          by        y         1
  em          by        m         1
  y           on        x         0.00
  y           on        m         0.50
  m           on        x         0.50
  ey          with      ey        168.75
  em          with      em        168.75
```

```
  x          with      x         225
  y          on        1         50
  m          on        1         50
  x          on        1         100
"
```

```
ramR::Eq2Expectations(eq)
```

```
## $par.table
##    lhs   op rhs par.label par.index
## 1   ey   by   y      1.00      1.00
## 2   em   by   m      1.00      1.00
## 3    y   on   x      0.00      0.00
## 4    y   on   m      0.50      0.50
## 5    m   on   x      0.50      0.50
## 6   ey with  ey    168.75    168.75
## 7   em with  em    168.75    168.75
## 8    x with   x    225.00    225.00
## 9    y   on   1     50.00     50.00
## 10   m   on   1     50.00     50.00
## 11   x   on   1    100.00    100.00
##
## $variables
## [1] "y"  "m"  "x"  "ey" "em"
##
## $g.variables
## [1] "y" "m" "x"
##
## $h.variables
## [1] "ey" "em"
##
## $A
##    y   m   x ey em
## y  0 0.5 0.0  1  0
## m  0 0.0 0.5  0  1
## x  0 0.0 0.0  0  0
## ey 0 0.0 0.0  0  0
## em 0 0.0 0.0  0  0
##
## $S
##    y m   x     ey     em
## y  0 0   0   0.00   0.00
## m  0 0   0   0.00   0.00
## x  0 0 225   0.00   0.00
## ey 0 0   0 168.75   0.00
## em 0 0   0   0.00 168.75
##
## $u
##      u
## y   50
## m   50
## x  100
## ey   0
```

```
## em   0
##
## $Filter
##   y m x ey em
## y 1 0 0  0  0
## m 0 1 0  0  0
## x 0 0 1  0  0
##
## $v
##      v
## y  100
## m  100
## x  100
## ey   0
## em   0
##
## $g
##     g
## y 100
## m 100
## x 100
##
## $C
##          y      m      x     ey      em
## y  225.000 112.50  56.25 168.75  84.375
## m  112.500 225.00 112.50   0.00 168.750
## x   56.250 112.50 225.00   0.00   0.000
## ey 168.750   0.00   0.00 168.75   0.000
## em  84.375 168.75   0.00   0.00 168.750
##
## $M
##        y     m      x
## y 225.00 112.5  56.25
## m 112.50 225.0 112.50
## x  56.25 112.5 225.00
```

Chapter 9

The Standardized Simple Mediation Model

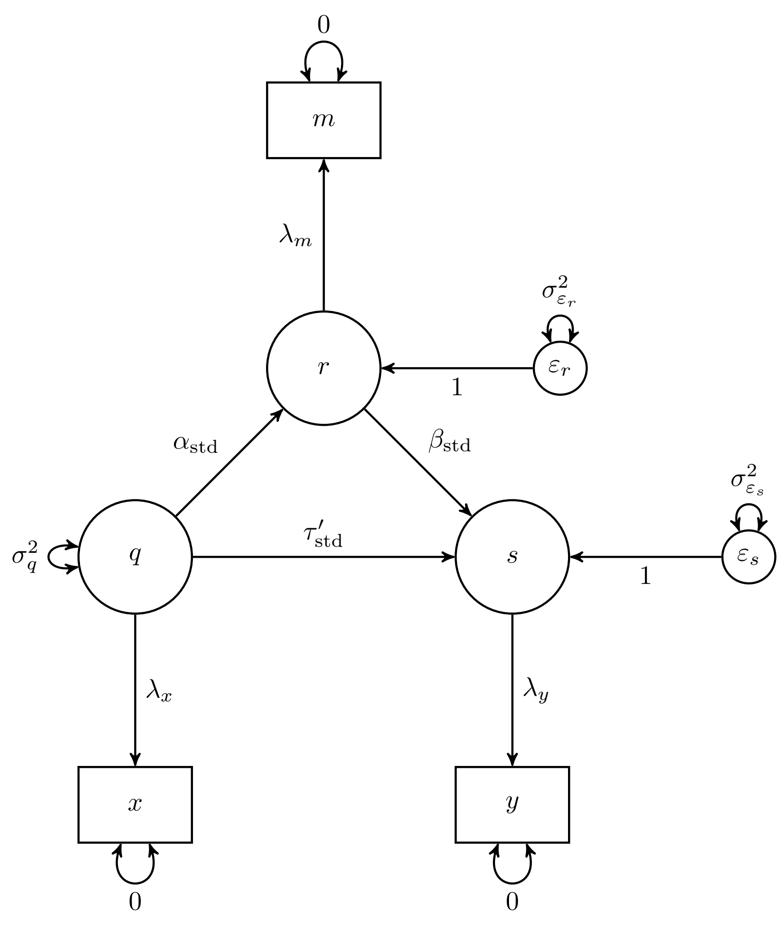


Figure 9.1: The Standardized Simple Mediation Model

Bibliography

Boker, S. M. and McArdle, J. J. (2005). Path analysis and path diagrams. In Everitt, B. S. and Howell, D. C., editors, *Encyclopedia of Statistics in Behavioral Science*, pages 1529–1531. John Wiley & Sons, Ltd, Chichester, UK.

Boker, S. M., Neale, M. C., Maes, H. H., Wilde, M. J., Spiegel, M., Brick, T. R., Estabrook, R., Bates, T. C., Mehta, P., von Oertzen, T., Gore, R. J., Hunter, M. D., Hackett, D. C., Karch, J., Brandmaier, A. M., Pritikin, J. N., Zahery, M., Kirkpatrick, R. M., Wang, Y., Goodrich, B., Driver, C., Massachusetts Institute of Technology, Johnson, S. G., Association for Computing Machinery, Kraft, D., Wilhelm, S., Medland, S., Falk, C. F., Keller, M., Manjunath B G, The Regents of the University of California, Ingber, L., Shao Voon, W., Palacios, J., Yang, J., Guennebaud, G., and Niesen, J. (2020). *OpenMx 2.18.1 User Guide*.

McArdle, J. J. (2005). The development of the RAM rules for latent variable structural equation modeling. In Maydeu-Olivares, A. and McArdle, J. J., editors, *Contemporary psychometrics: A festschrift for Roderick P. McDonald*, Multivariate applications book series, pages 225–273. Lawrence Erlbaum Associates, Mahwah, NJ.

McArdle, J. J. and McDonald, R. P. (1984). Some algebraic properties of the reticular action model for moment structures. *British Journal of Mathematical and Statistical Psychology*, 37(2):234–251.

Pesigan, I. J. A. (2021). *ramR: Reticular Action Model (RAM) Notation*. R package version 0.9.0.

R Core Team (2020). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Rosseel, Y. (2012). lavaan: An R package for structural equation modeling. *Journal of Statistical Software*, 48(2):1–36.