Introduction to `pathdiagram`

Gaston **S**anchez

www.gastonsanchez.com

1   Introduction

`pathdiagram` is an accessory R package for drawing path diagrams in R, like the one below:
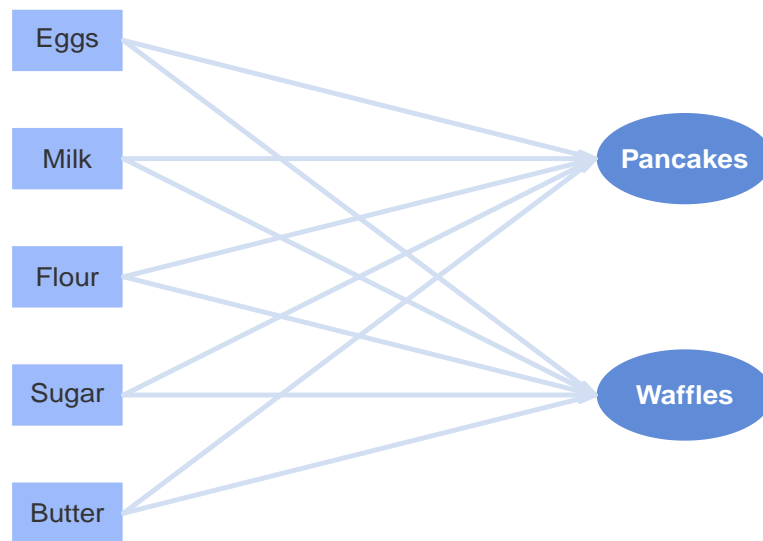

Figure 1: example of a path diagram

`pathdiagram` is intended to help you plot nice and beautiful path diagrams for visualization and illustration purposes, especially when generating reports in R with `knitr` or `Sweave`. Briefly, path diagrams are just a graphical means to visually display path models. The aim of the R package `pathdiagram` is to help you draw path diagrams the way they are usually depicted under a **Partial Least Squares Path Modeling** (PLS-PM) approach.

1.1   Motivation

The one and only reason to create `pathdiagram` was the need to plot a path diagram in R. Since I was using `knitr` (by Yihui Xie) to write the tutorials and demos for the package `plspm`, I realized that I badly needed path diagrams accompanying my examples. I first tried using the `grid` package (by Paul Murrell). It was fine but it didn't allow me to have complete control over the looking of the objects (mainly with color of lines, borders, text, etc). Then I tried to use the package `diagram` (by Karline Soetaert) but I got the same restrictions. Finally, after some short experiments, I decided to create `pathdiagram`.

1.2 Installation

Install `pathdiagram` as you would install any other package on the CRAN repository. Simply run in your R console:

```r
# installation
install.packages("pathdiagram")
```

Once `pathdiagram` has been installed, you just need to load it using the function `library()`:

```r
# load pathdiagram
library(pathdiagram)
```

1.3 Available Functions

`pathdiagram` has a very simple structure and it provides five functions:

Function	Description
`manifest()`	sets the graphical specifcations for a manifest variable
`latent()`	sets the graphical specifcations for a latent variable
`wall()`	opens a new plotting window to start drawing a path diagram
`draw()`	plots either a manifest or a latent variable
`arrow()`	plots an arrow between two specifed variables

These functions are all you need to create beautiful PLS-like path diagrams in R. This type of diagrams follow a set of semi-established rules to represent different types of variables, as well as the relationships between them. The general convention is to display manifest variables in a rectangular shape, and latent variables in a circular-elliptical way. I'm not going to describe the etiquette rules for PLS path diagrams (if you google a little bit you will find more information about this topic). Instead, I'm going to focus on describing the know-how for generating path diagrams with `pathdiagram`.

1.4 `pathdiagram` basics

When we draw a path model, we basically use three types of geometric objects:

- **rectangles** (or squares) for representing manifest variables

- **ellipses** (or circles) for representing latent variables

- **arrows** to establich the connections between (manifest and latent) variable

The function `manifest()` lets you define the graphical specifications for drawing manifest variables in a rectangular shape. The function `latent()` lets you define the graphical specifications for drawing latent variables in an elliptical format. And the function `arrow()` allows you to set the graphical specifications and draw an arrow between two given variables.

In addition, we also have two more functions. `wall()` is just a conveniente way to open a plot window so we can start drawing a path diagram. By default it opens a window in a $x - y$ coordinate region of $[0, 1]$x$[0, 1]$, but you could change the predetermined limits. The other function is `draw()` which, as its name suggests, lets you draw the manifest and latent variables on a plotting window.

2 Toy Example

Let's start by considering an informal simple path model for representing the ingredients of two of my favorite breakfast plates: *waffles* and *pancakes*. Both dishes share the same ingredients: eggs, milk, flour, sugar, and butter. But the difference between waffles and pancakes is in the way they are cooked. The important thing for this example is that the ingredients will play the role of manifest variables, while the produced plates will play the role of latent constructs.

2.1 `manifest()` variables

We begin by defining the manifest variables: *eggs, milk, flour, sugar, butter*. For illustration purposes, I'm going to define each ingredient separately inside a list. Each variable is defined with the function `manifest()`. The first argument is the label that will be displayed inside the rectangle. The rest of the parameters set the graphical characteristics like the coordinates of the rectangle's center, its width and height, the fill color, the border color, the type of font for the label, etc.

```
# manifest variables
ingredients = list(
  eggs = manifest("Eggs", x = 0.25, y = 0.8, width = 0.1, height = 0.08),
  milk = manifest("Milk", x = 0.25, y = 0.65, width = 0.1, height = 0.08),
  flour = manifest("Flour", x = 0.25, y = 0.5, width = 0.1, height = 0.08),
  sugar = manifest("Sugar", x = 0.25, y = 0.35, width = 0.1, height = 0.08),
  butter = manifest("Butter", x = 0.25, y = 0.2, width = 0.1, height = 0.08)
)
```

2.2 `latent()` variables

The following step is to define the latent variables by using the function `latent()` which has pretty much the same parameters of `manifest()`. The only difference is that `latent()` does not have a `width` and a `height` parameters. Instead, it has `rx` and `ry` which are the major and minor radius of the ellipse.

```
# latent variables
pancakes = latent("Pancakes", x = 0.8, y = 0.65, rx = 0.08, ry = 0.06)
waffles = latent("Waffles", x = 0.8, y = 0.35, rx = 0.08, ry = 0.06)
```

The real challenge when using `manifest()` and `latent()` is in the values for the parameters `x, y, width, height, rx` and `ry`. To be honest, you will need to try different values until you find the best location and looking shapes for your manifest and latent variables.

2.3 `draw()`ing the diagram

Once we have defined the variables, the next step is to draw them in a plot window. `pathdiagram` comes with the function `wall()` that opens a white canvas to start drawing your diagram. But you could as well use the typical `plot()` function to open a plotting window.

```
# open a new wall
wall(xlim = c(0.1, 0.9), ylim = c(0.1, 0.9))

# draw latent variables
draw(pancakes)
draw(waffles)

# draw ingredients
for (i in 1:5) {
    draw(ingredients[[i]])
}
```

The last step is to add the connecting arrows with the function `arrow()`. Basically, you just need to tell `arrow()` from which variable to which variable there is a connection, and what are the starting and ending positions for the arrow according to the four cardinal options: `"north"`, `"south"`, `"east"`, and `"west"`.

```
# arrows
for (i in 1:5) {
    arrow(from = ingredients[[i]], to = pancakes, start = "east", end = "west")
    arrow(from = ingredients[[i]], to = waffles, start = "east", end = "west")
}
```
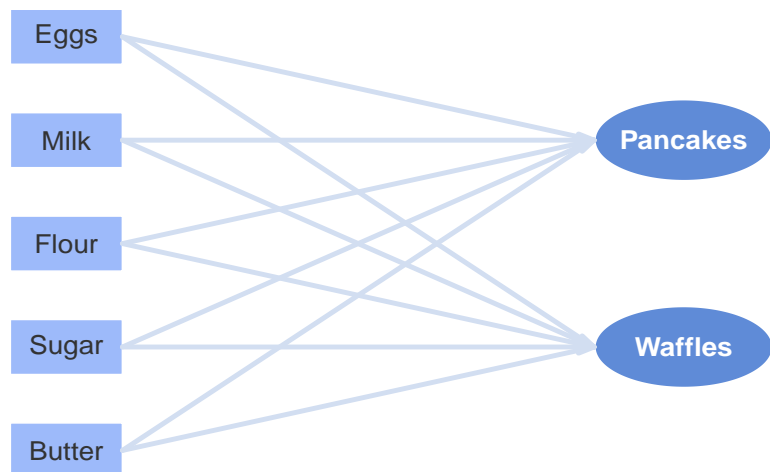
3   Some Path Diagram Examples

The best way to learn about `pathdiagram` is to show you some examples. In this section you will find four different diagrams (and their corresponding code snippets) that show some of the plots you can create with `pathdiagram`.

3.1   Barcelona FC

The first example consists of a path diagram of the FC Barcelona soccer team. This model has 3 blocks of variables: `attack`, `defense`, and `success`. Each block corresponds to a latent construct, and each of them contains 3 manifest variables.
The first step is to define the graphical specifications for both the manifest and the latent variables as follows:

```
# define Attack block
attack = list(
  att1 = manifest("Messi", x=0.15, y=0.9, width=0.09, height=0.08, fill="#d199a4"),
  att2 = manifest("Xavi", x=0.15, y=0.75, width=0.09, height=0.08, fill="#d199a4"),
  att3 = manifest("Iniesta", x=0.15, y=0.6, width=0.09, height=0.08, fill="#d199a4"))
ATTACK = latent("ATTACK", x=0.35, y=0.75, rx=0.08, ry=0.07, fill="#a12b43", font=1)

# define Defense block
defense = list(
  def1 = manifest("Puyol", x=0.15, y=0.4, width=0.09, height=0.08, fill="#a0bee1"),
  def2 = manifest("Pique", x=0.15, y=0.25, width=0.09, height=0.08, fill="#a0bee1"),
  def3 = manifest("Abidal", x=0.15, y=0.1, width=0.09, height=0.08, fill="#a0bee1"))
DEFENSE = latent("DEFENSE", x=0.35, y=0.25, rx=0.08, ry=0.07, fill="#1e67ba", font=1)

# define Success block
success = list(
  suc1 = manifest("2008-2009", x=0.85, y=0.65, width=0.14, height=0.08, fill="gold2"),
  suc2 = manifest("2009-2010", x=0.85, y=0.5, width=0.14, height=0.08, fill="gold2"),
  suc3 = manifest("2010-2011", x=0.85, y=0.35, width=0.14, height=0.08, fill="gold2"))
SUCCESS = latent("SUCCESS", x=0.65, y=0.5, rx=0.08, ry=0.07, fill="gold2", font=1)
```

After defining the graphical parameters, we plot it with the functions `wall()` and `draw()` like this:

```
# open plot window
wall(ylim = c(0.1, 0.9))

# draw latent variables
draw(ATTACK)
draw(DEFENSE)
draw(SUCCESS)

# draw manifest variables
for (i in 1:3) {
    draw(attack[[i]])
    arrow(from = attack[[i]], to = ATTACK, start = "east", end = "west", col = "#d199a4")
    draw(defense[[i]])
    arrow(from = defense[[i]], to = DEFENSE, start = "east", end = "west", col = "#a0bee1")
    draw(success[[i]])
    arrow(from = SUCCESS, to = success[[i]], start = "east", end = "west", col = "gold1")
}
```

```
# arrows of inner model
arrow(from = ATTACK, to = SUCCESS, start = "east", end = "west", col = "#d199a4")
arrow(from = DEFENSE, to = SUCCESS, start = "east", end = "west", col = "#a0bee1")
```
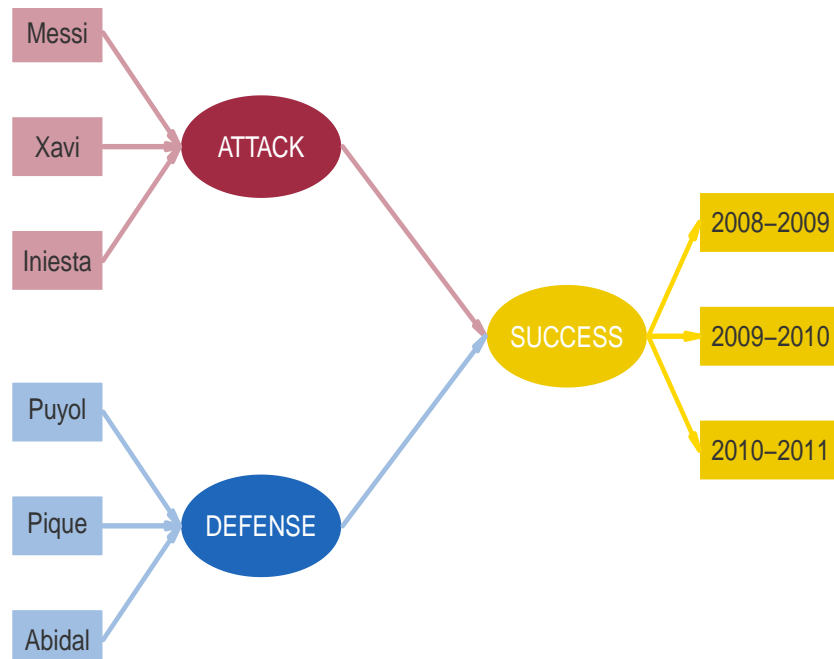
Figure 2: FC Barcelona path diagram

3.2 Harry Potter

The second example is a path diagram based on Harry Potter's books. The model illustrates the competition between `Gryffindor` and `Slytherin`.

```r
# Gryffindor block
gryff = list(
  harry = manifest("Harry \nPotter", x=0.15, y=0.8, width=0.12, height=0.08,
    cex=0.8, fill="#f2d22e", col="#7c4f87", family="serif"),

  ron = manifest("Ron\nWeasley", x=0.15, y=0.7, width=0.12, height=0.08,
    cex=0.8, fill="#f2d22e", col="#7c4f87", family="serif"),

  hermione = manifest("Hermione\nGranger", x=0.15, y=0.6, width=0.12, height=0.08,
    cex=0.8, fill="#f2d22e", col="#7c4f87", family="serif"),

  albus = manifest("Albus\nDumbledore", x=0.15, y=0.5, width=0.12, height=0.08,
    cex=0.8, fill="#f2d22e", col="#7c4f87", family="serif"),

  neville = manifest("Neville\nLongbottom", x=0.15, y=0.4, width=0.12, height=0.08,
    cex=0.8, fill="#f2d22e", col="#7c4f87", family="serif"),

  sirius = manifest("Sirius\nBlack", x=0.15, y=0.3, width=0.12, height=0.08,
    cex=0.8, fill="#f2d22e", col="#7c4f87", family="serif"),

  rubeus = manifest("Rubeus\nHagrid", x=0.15, y=0.2, width=0.12, height=0.08,
    cex=0.8, fill="#f2d22e", col="#7c4f87", family="serif"))
```

```r
# Slytherin block
slyth = list(
  tom = manifest("Tom\nRiddle", x=0.85, y=0.8, width=0.12, height=0.08,
    cex=0.8, fill="gray70", col="#467d70", family="serif"),

  severus = manifest("Severus\nSnape", x=0.85, y=0.7, width=0.12, height=0.08,
    cex=0.8, fill="gray70", col="#467d70", family="serif"),

  bella = manifest("Bellatrix\nLestrange", x=0.85, y=0.6, width=0.12, height=0.08,
    cex=0.8, fill="gray70", col="#467d70", family="serif"),

  regulus = manifest("Regulus\nBlack", x=0.85, y=0.5, width=0.12, height=0.08,
    cex=0.8, fill="gray70", col="#467d70", family="serif"),

  phineas = manifest("Phineas\nBlack", x=0.85, y=0.4, width=0.12, height=0.08,
    cex=0.8, fill="gray70", col="#467d70", family="serif"),

  draco = manifest("Draco\nMalfoy", x=0.85, y=0.3, width=0.12, height=0.08,
    cex=0.8, fill="gray70", col="#467d70", family="serif"),

  horace = manifest("Horace\nSlughorn", x=0.85, y=0.2, width=0.12, height=0.08,
    cex=0.8, fill="gray70", col="#467d70", family="serif"))
```

```
# latent variables
gry = latent("Gryffindor", x=0.375, y=0.5, rx=0.07, ry=0.06, cex=0.85,
    fill="#7c4f87", family="serif")

sly = latent("Slytherin", x=0.625, y=0.5, rx=0.07, ry=0.06, cex=0.85,
    fill="#467d70", family="serif")
```

Now we plot it:

```
# open plot window
wall(xlim = c(0.1, 0.9), ylim = c(0.15, 0.85))

# draw variables
for (i in 1:7) {
    # arrows between each block and its latent
    arrow(from = gryff[[i]], to = gry, start = "east", end = "west", col = "#b095b7",
        angle = 5, lwd = 1)
    arrow(from = slyth[[i]], to = sly, start = "west", end = "east", col = "#90b1a9",
        angle = 5, lwd = 1)

    # variables
    draw(gryff[[i]])
    draw(slyth[[i]])
    draw(gry)
    draw(sly)

    # arrows between latent variables
    arrow(from = gry, to = sly, start = "east", end = "west", col = "#dddddd",
        angle = 20)
    arrow(from = sly, to = gry, start = "west", end = "east", col = "#dddddd",
        angle = 20)
}
```
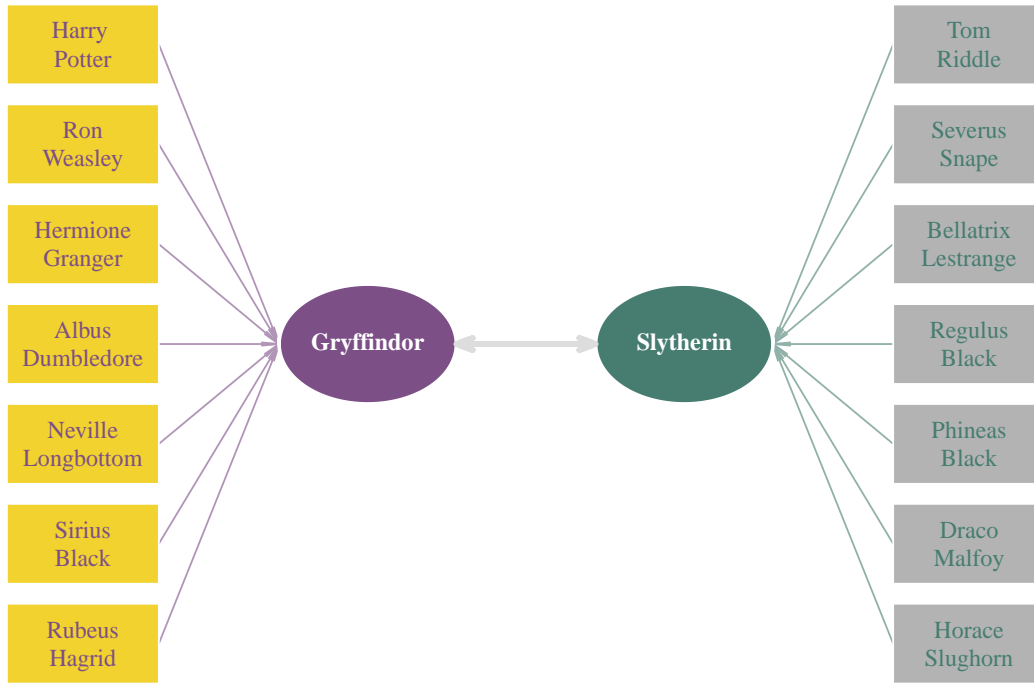
Figure 3: Gryffindor vs Slytherin path diagram

3.3   My Model

The next example consists of a path diagram for my achievement model. This model has 7 latent variables:
`optimism`, `dedication`, `sacrifice`, `work`, `achievement`, and `luck`.

```
# latent variables
optimism = latent("Optimism", x=0.35, y=0.75, rx=0.08, ry=0.06,
                  fill="gray90", col="#1B9E77", font=1)


dedication = latent("Dedication", x=0.2, y=0.6, rx=0.08, ry=0.06,
                    fill="gray90", col="#D95F02", font=1)


patience = latent("Patience", x=0.2, y=0.4, rx=0.08, ry=0.06,
                  fill="gray90", col="#7570B3", font=1)


sacrifice = latent("Sacrifice", x=0.35, y=0.25, rx=0.08, ry=0.06,
                   fill="gray90", col="#E7298A", font=1)


work = latent("Work", x=0.5, y=0.5, rx=0.08, ry=0.06,
              fill="gray90", col="#1F78B4", font=1)


achievement = latent("Achievement", x=0.8, y=0.5, rx=0.10, ry=0.075,
                     fill="gray90", col="tomato", font=1)


luck = latent("Luck", x=0.85, y=0.7, rx=0.065, ry=0.06,
              fill="gray90", col="#E6AB02", font=1)
```

After defining the graphical specifications, we plot the diagram like so:

```
# open wall to plot
wall(ylim = c(0.15, 0.85))

# draw latent variables
draw(optimism)
draw(dedication)
draw(patience)
draw(sacrifice)
draw(work)
draw(achievement)
draw(luck)

# add arrows
arrow(from = optimism, to = work, start = "east", end = "north", col = "gray90")
arrow(from = dedication, to = work, start = "east", end = "west", col = "gray90")
arrow(from = patience, to = work, start = "east", end = "west", col = "gray90")
arrow(from = sacrifice, to = work, start = "east", end = "south", col = "gray90")
arrow(from = work, to = achievement, start = "east", end = "west", col = "gray90")
arrow(from = luck, to = achievement, start = "south", end = "north", col = "gray90")
```
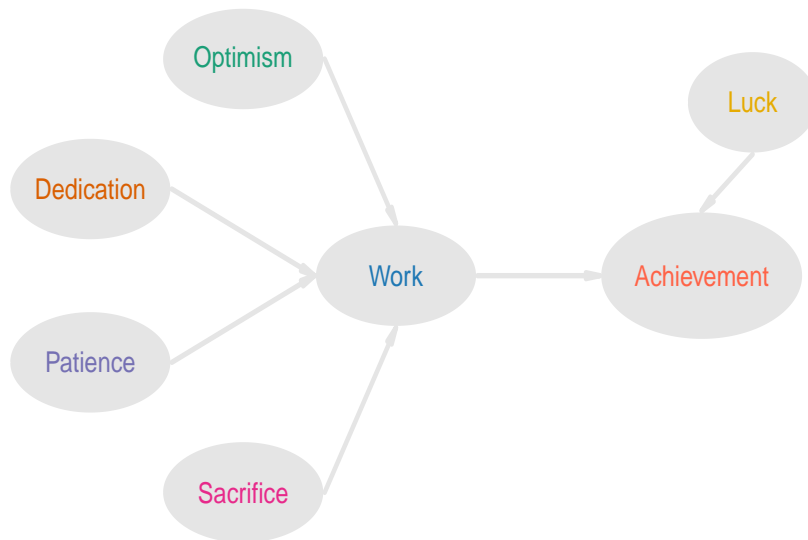

Figure 4: Personal Achievement Model path diagram

3.4 Model with Formal Notation

The following example shows a path diagram using a more formal notation. It consists of 5 latent variables measured in a reflective way. Moreover, using the function `expression()` it is possible to define the labels of the variables with mathematical as well as greek symbols.

```
# Block 1
X1 = list(
  x11 = manifest(expression(x[11]), x=0.4, y=0.9, width=0.06, height=0.06,
                 cex=0.8, fill="gray90", col="gray50"),
  x12 = manifest(expression(x[12]), x=0.47, y=0.9, width=0.06, height=0.06,
                 cex=0.8, fill="gray90", col="gray50"),
  x13 = manifest(expression(x[13]), x=0.53, y=0.9, width=0.06, height=0.06,
                 cex=0.8, fill="gray90", col="gray50"),
  x14 = manifest(expression(x[14]), x=0.6, y=0.9, width=0.06, height=0.06,
                 cex=0.8, fill="gray90", col="gray50")
)

LV1 = latent(expression(xi[1]), x=0.5, y=0.75, rx=0.06, ry=0.06,
             cex=1.2, fill="gray50")


# Block 2
X2 = list(
  x21 = manifest(expression(x[21]), x=0.1, y=0.63, width=0.06, height=0.06,
                 cex=0.8, fill="gray90", col="gray50"),
  x22 = manifest(expression(x[22]), x=0.1, y=0.55, width=0.06, height=0.06,
                 cex=0.8, fill="gray90", col="gray50"),
  x23 = manifest(expression(x[23]), x=0.1, y=0.47, width=0.06, height=0.06,
                 cex=0.8, fill="gray90", col="gray50")
)

LV2 = latent(expression(xi[2]), x=0.25, y=0.55, rx=0.06, ry=0.06,
             cex=1.2, fill="gray50")


# Block 3
X3 = list(
  x31 = manifest(expression(x[31]), x=0.1, y=0.33, width=0.06, height=0.06,
                 cex=0.8, fill="gray90", col="gray50"),
  x32 = manifest(expression(x[32]), x=0.1, y=0.25, width=0.06, height=0.06,
                 cex=0.8, fill="gray90", col="gray50"),
  x33 = manifest(expression(x[33]), x=0.1, y=0.17, width=0.06, height=0.06,
                 cex=0.8, fill="gray90", col="gray50")
)

LV3 = latent(expression(xi[3]), x=0.25, y=0.25, rx=0.06, ry=0.06,
             cex=1.2, fill="gray50")


# Block 4
X4 = list(
  x41 = manifest(expression(x[41]), x=0.4, y=0.45, width=0.06, height=0.06,
                 cex=0.8, fill="gray90", col="gray50"),
  x42 = manifest(expression(x[42]), x=0.4, y=0.35, width=0.06, height=0.06,
```

```
                        cex=0.8, fill="gray90", col="gray50")
)

LV4 = latent(expression(xi[4]), x=0.55, y=0.4, rx=0.06, ry=0.06,
             cex=1.2, fill="gray50")


# Block 5
X5 = list(
  x51 = manifest(expression(x[51]), x=0.95, y=0.47, width=0.06, height=0.06,
                 cex=0.8, fill="gray90", col="gray50"),
  x52 = manifest(expression(x[52]), x=0.95, y=0.4, width=0.06, height=0.06,
                 cex=0.8, fill="gray90", col="gray50"),
  x53 = manifest(expression(x[53]), x=0.95, y=0.33, width=0.06, height=0.06,
                 cex=0.8, fill="gray90", col="gray50")
)

LV5 = latent(expression(xi[5]), x=0.8, y=0.4, rx=0.06, ry=0.06,
             cex=1.2, fill="gray50")
```

Now we plot it:

```
# open plot window
wall(xlim = c(0, 1), ylim = c(0.05, 0.95))

# block 1
draw(LV1)
for (i in 1:length(X1)) {
    draw(X1[[i]])
    arrow(from = X1[[i]], to = LV1, start = "south", end = "north", lwd = 1,
        col = "gray80")
}

# block 2
draw(LV2)
for (i in 1:length(X2)) {
    draw(X2[[i]])
    arrow(from = X2[[i]], to = LV2, start = "east", end = "west", lwd = 1, col = "gray80")
}

# block 3
draw(LV3)
for (i in 1:length(X3)) {
    draw(X3[[i]])
    arrow(from = X3[[i]], to = LV3, start = "east", end = "west", lwd = 1, col = "gray80")
}

# block 4
draw(LV4)
for (i in 1:length(X4)) {
    draw(X4[[i]])
    arrow(from = X4[[i]], to = LV4, start = "east", end = "west", lwd = 1, col = "gray80")
}
```

```
# block 5
draw(LV5)
for (i in 1:length(X5)) {
    draw(X5[[i]])
    arrow(from = X5[[i]], to = LV5, start = "west", end = "east", lwd = 1, col = "gray80")
}

# arrows between latent variables
arrow(from = LV1, to = LV2, start = "west", end = "north", col = "gray80")
arrow(from = LV1, to = LV5, start = "east", end = "north", col = "gray80")
arrow(from = LV2, to = LV3, start = "south", end = "north", col = "gray80")
arrow(from = LV2, to = LV4, start = "east", end = "north", col = "gray80")
arrow(from = LV3, to = LV4, start = "east", end = "south", col = "gray80")
arrow(from = LV4, to = LV5, start = "east", end = "west", col = "gray80")
```
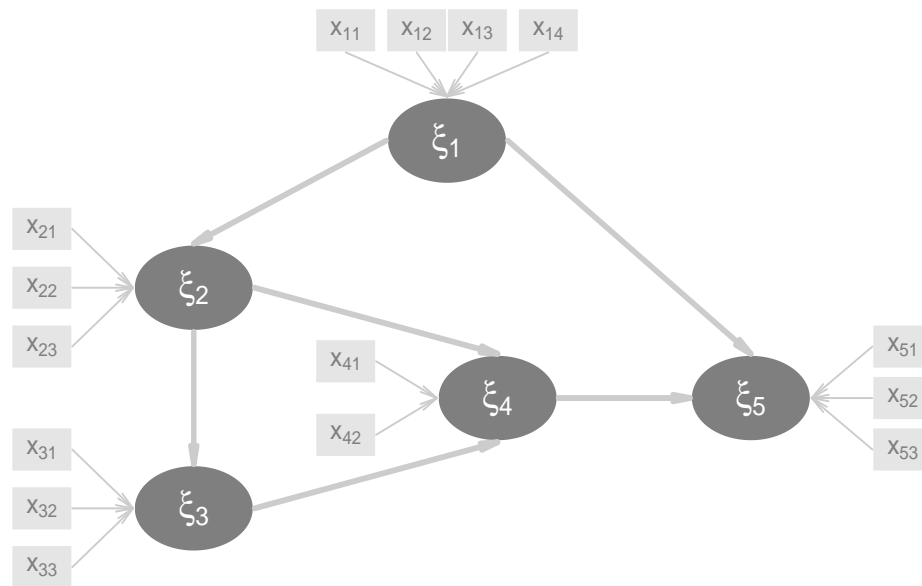

Figure 5: Formal Notation Model path diagram