

SEM and R

Bill

2021-04-21

Contents

1	SEM and R	5
2	Introduction	7
2.1	Definitions (Basic Concepts)	7
2.2	The path diagram	8
2.3	Lavaan syntax	8
2.4	Regression and path analysis	9
3	Real data example (Simple linear regression)	11
3.1	Read the data into the R Studio environment.	11
4	Real data example (Multiple linear regression)	15
5	Bootstrapping	17

Chapter 1

SEM and R

This is the starting point.

Chapter 2

Introduction

The following R codes and texts are from UCLA website “<https://stats.idre.ucla.edu/r/seminars/rsem/>” and I do not own the copyright of the R codes or texts. I wrote this R Markdown file for my own study purpose.

Given this consideration, please do NOT distribute this page in any way.

2.1 Definitions (Basic Concepts)

2.1.1 Observed variable

Observed variable: A variable that exists in the data (a.k.a item or manifest variable)

2.1.2 Latent variable

Latent variable: A variable that is constructed and does not exist in the data.

2.1.3 Exogenous variable

Exogenous variable: An independent variable either observed (X) or latent (ξ) that explains an endogenous variable.

2.1.4 Endogenous variable

Endogenous variable: A dependent variable, either observed (Y) or latent (η) that has a causal path leading to it.

2.1.5 Measurement model

Measurement model: A model that links observed variables with latent variables.

2.1.6 Indicator (in a measurement model)

Indicator: An observed variable in a measurement model (can be exogenous or endogenous).

2.1.7 Factor

Factor: A latent variable defined by its indicators (can be exogenous or endogenous).

2.1.8 Loading

Loading: A path between an indicator and a factor.

2.1.9 Structural model

Structural model: A model that specifies casual relationships among exogenous variables to endogenous variables (can be observed or latent).

2.1.10 Regression path

Regression path: A path between exogenous and endogenous variables (can be observed or latent).

2.2 The path diagram

Circles represent latent variables. Squares represent observed indicators. Triangles represent intercepts or means. One way arrows represent paths. Two-way arrows represent either variances or covariances.

2.3 Lavaan syntax

\sim **predict**: used for regression of observed outcome to observed predictors (e.g., $y \sim x$).

$=\sim$ **indicator**: used for latent variable to observed indicator in factor analysis measurement models (e.g., $f =\sim q + r + s$).

$\sim\sim$ **covariance**: (e.g., $x \sim\sim x$).

~ 1 **intercept or mean**: (e.g., $x \sim 1$ estimates the mean of variable x).

$1*$ **fixes parameter or loading to one**: (e.g., $f =\sim 1 * q$).

*NA** **free parameter or loading**: used to override default marker method (e.g., $f = \sim NA * q$).

*a** **labels the parameter 'a'**: used for model constraints (e.g., $f = \sim a * q$).

2.4 Regression and path analysis

$$y_1 = b_0 + b_1 x_1 + \epsilon_1$$

$$y_1 = \alpha + \gamma_1 x_1 + \zeta_1$$

x_1 single exogenous variable

y_1 single endogenous variable

b_0, α_1 intercept of y_1 (alpha)

b_1, γ_1 regression coefficient (gamma)

ϵ_1, ζ_1 residual of y_1 (epsilon, zeta)

ϕ variance or covariance of the exogenous variable (phi)

ψ residual variance or covariance of the endogenous variable (psi)

Chapter 3

Real data example (Simple linear regression)

3.1 Read the data into the R Studio environment.

It also calculates the covariance matrix among all the variables in the data.

```
dat <- read.csv("https://stats.idre.ucla.edu/wp-content/uploads/2021/02/worland5.csv")
cov(dat)
```

```
##      motiv harm stabi ppsych ses verbal read arith spell
## motiv    100   77   59   -25  25    32   53   60   59
## harm      77  100   58   -25  26    25   42   44   45
## stabi     59   58  100   -16  18    27   36   38   38
## ppsych    -25 -25  -16   100 -42   -40  -39  -24  -31
## ses       25  26   18   -42 100    40   43   37   33
## verbal    32  25   27   -40  40   100   56   49   48
## read      53  42   36   -39  43    56  100   73   87
## arith     60  44   38   -24  37    49   73  100   72
## spell     59  45   38   -31  33    48   87   72  100
```

```
var(dat$motiv)
```

```
## [1] 100
```

In the following, we conduct a simple linear regression.

$$\text{sample variance - covariance matrix } \hat{\Sigma} = \mathbf{S}$$

12 CHAPTER 3. REAL DATA EXAMPLE (SIMPLE LINEAR REGRESSION)

```

m1a <- lm(read ~ motiv, data=dat)
(summary(m1a))

##
## Call:
## lm(formula = read ~ motiv, data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -26.0995  -6.1109   0.2342   5.2237  24.0183
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.232e-07  3.796e-01   0.00    1
## motiv       5.300e-01  3.800e-02  13.95 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.488 on 498 degrees of freedom
## Multiple R-squared:  0.2809, Adjusted R-squared:  0.2795
## F-statistic: 194.5 on 1 and 498 DF, p-value: < 2.2e-16

library(lavaan)
#simple regression using lavaan
m1b <- '
# regressions
read ~ 1* motiv
# variance (optional)
motiv ~~ motiv
'

fit1b <- sem(m1b, data=dat)
summary(fit1b)

## lavaan 0.6-8 ended normally after 14 iterations
##
## Estimator                      ML
## Optimization method            NLMINB
## Number of model parameters      5
##
## Number of observations          500
##
## 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|)
##      read ~
##      motiv           0.530    0.038   13.975    0.000
##
## Intercepts:
##              Estimate  Std.Err  z-value  P(>|z|)
##      .read          -0.000    0.379   -0.000    1.000
##      motiv           0.000    0.447    0.000    1.000
##
## Variances:
##              Estimate  Std.Err  z-value  P(>|z|)
##      motiv          99.800    6.312   15.811    0.000
##      .read          71.766    4.539   15.811    0.000
```


Chapter 4

Real data example (Multiple linear regression)

```
m2 <- '
# regressions
read ~ 1 + ppsych + motiv
# covariance
ppsyach ~~ motiv
'
fit2 <- sem(m2, data=dat)
summary(fit2)
```

```
## lavaan 0.6-8 ended normally after 34 iterations
##
##      Estimator                      ML
##      Optimization method          NLMINB
##      Number of model parameters      9
##
##      Number of observations          500
##
## Model Test User Model:
##
##      Test statistic                  0.000
##      Degrees of freedom              0
##
## Parameter Estimates:
##
##      Standard errors                Standard
##      Information                    Expected
```

16 CHAPTER 4. REAL DATA EXAMPLE (MULTIPLE LINEAR REGRESSION)

```
## Information saturated (h1) model          Structured
##
## Regressions:
##           Estimate Std.Err  z-value  P(>|z|)
##   read ~
##     ppsych      -0.275    0.037   -7.385    0.000
##     motiv       0.461    0.037   12.404    0.000
##
## Covariances:
##           Estimate Std.Err  z-value  P(>|z|)
##     ppsych ~~
##       motiv     -24.950    4.601   -5.423    0.000
##
## Intercepts:
##           Estimate Std.Err  z-value  P(>|z|)
##     .read        0.000    0.360    0.000    1.000
##     ppsych       -0.000    0.447   -0.000    1.000
##     motiv        0.000    0.447    0.000    1.000
##
## Variances:
##           Estimate Std.Err  z-value  P(>|z|)
##     .read        64.708    4.092   15.811    0.000
##     ppsych       99.800    6.312   15.811    0.000
##     motiv       99.800    6.312   15.811    0.000
```


Chapter 5

Bootstrapping

The following note is made when I was studying Bret Larget's note posted online.
<http://pages.stat.wisc.edu/~larget/stat302/chap3.pdf>

He used the data from LOck5data as an example.

```
library(Lock5Data)
data(CommuteAtlanta)
str(CommuteAtlanta)

## 'data.frame':    500 obs. of  5 variables:
## $ City      : Factor w/ 1 level "Atlanta": 1 1 1 1 1 1 1 1 1 1 ...
## $ Age       : int  19 55 48 45 48 43 48 41 47 39 ...
## $ Distance: int  10 45 12 4 15 33 15 4 25 1 ...
## $ Time      : int  15 60 45 10 30 60 45 10 25 15 ...
## $ Sex       : Factor w/ 2 levels "F","M": 2 2 2 1 1 2 2 1 2 1 ...

time.mean = with(CommuteAtlanta, mean(Time))

time.mean

## [1] 29.11
```

Now, he sampled a (b times n) table. Note that, the Atlanta data has 500 row, as it has 500 observations (or, people). But, in the following new matrix, it is a (1000 times 500) table. Also, it should be noted that the logic of sample function in R. This webpage provides some insight into this function. Basically, the following R code randomly sample a bigger sample of (1000 times 500) from those 500 data points. After that, the matrix function put such (1000 times 500) data points into a matrix of (1000 times 500).

```
B = 1000
n = nrow(CommuteAtlanta)
```

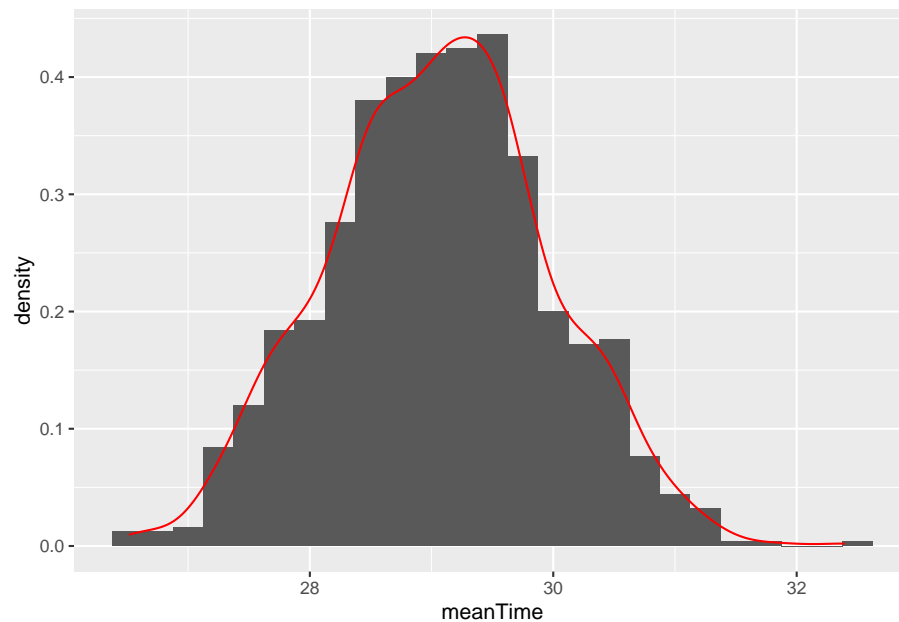
```
boot.samples = matrix(sample(CommuteAtlanta$Time, size = B * n, replace = TRUE),
                      B, n)
```

Next, we need to calculate the mean for each row. Remember, we have 1000 rows. Note that, 1 in the apply function indicates that we calculate means on each row, whereas 2 indicates to each column.

```
boot.statistics = apply(boot.samples, 1, mean)
```

We can then plot all the means.

```
require(ggplot2)
ggplot(data.frame(meanTime = boot.statistics), aes(x=meanTime)) +
  geom_histogram(binwidth=0.25, aes(y=..density..)) +
  geom_density(color="red")
```



```
time.se = sd(boot.statistics)
time.se
```

```
## [1] 0.9091176
```

```
me = ceiling(10 * 2 * time.se)/10
me
```

```
## [1] 1.9
```

```
round(time.mean, 1) + c(-1, 1) * me
```

```
## [1] 27.2 31.0
```

Note, if we do not use bootstrapping, we can use the standard CI formula (<https://www.mathsisfun.com/data/confidence-interval.html>). This formula assumes normal distribution. As we can see, this is close to the result based on the bootstrapping method.

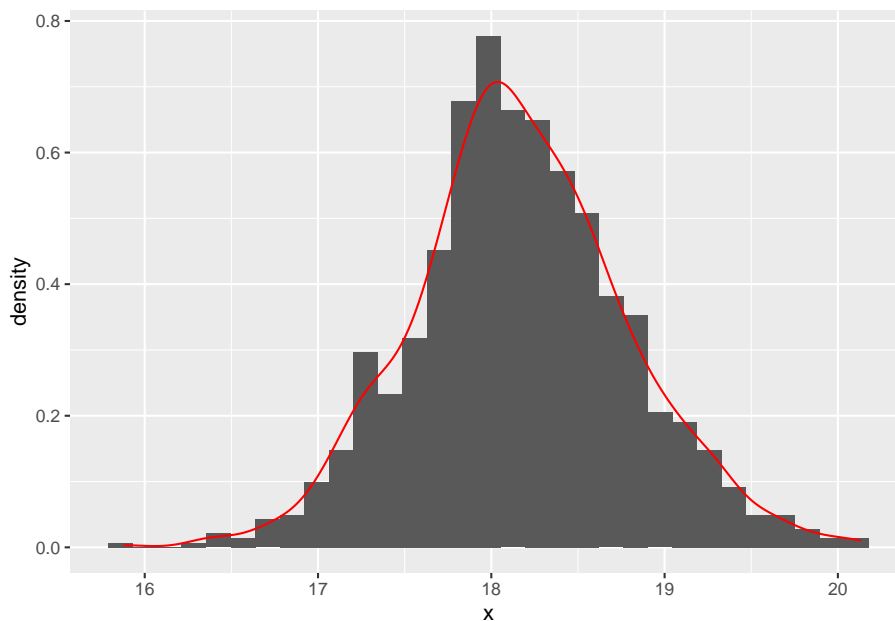
$$\bar{X} \pm Z \frac{S}{\sqrt{n}} = 29.11 \pm 1.96 \frac{20.72}{\sqrt{500}} = 27.29, 30.93$$

Note that, in the following, the author used 2 times SE to calculate the CI. The relationship between SD and SE:

“Now the sample mean will vary from sample to sample; the way this variation occurs is described by the “sampling distribution” of the mean. We can estimate how much sample means will vary from the standard deviation of this sampling distribution, which we call the standard error (SE) of the estimate of the mean. As the standard error is a type of standard deviation, confusion is understandable. Another way of considering the standard error is as a measure of the precision of the sample mean.” (<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1255808/>)

```
boot.mean = function(x,B,binwidth=NULL)
{
  n = length(x)
  boot.samples = matrix( sample(x,size=n*B,replace=TRUE), B, n)
  boot.statistics = apply(boot.samples,1,mean)
  se = sd(boot.statistics)
  require(ggplot2)
  if ( is.null(binwidth) )
    binwidth = diff(range(boot.statistics))/30
  p = ggplot(data.frame(x=boot.statistics),aes(x=x)) +
    geom_histogram(aes(y=..density..),binwidth=binwidth) + geom_density(color="red")
  plot(p)
  interval = mean(x) + c(-1,1)*2*se
  print( interval )
  return( list(boot.statistics = boot.statistics, interval=interval, se=se, plot=p) )
}

out = with(CommuteAtlanta, boot.mean(Distance, B = 1000))
```



```
## [1] 16.9218 19.3902
```

To understand the function of sample in R.

```
sample(20,replace = TRUE)
```

```
## [1] 17 17 14 17 14 10 13 17 7 1 12 7 12 12 8 15 4 18 19 7
```

The following uses loop to do the resampling. It uses sample function to index the numbers that they want to sample from the original sample. That is, [] suggests the indexing.

```
n = length(CommuteAtlanta$Distance)
B = 1000
result = rep(NA, B)
for (i in 1:B)
{
  boot.sample = sample(n, replace = TRUE)
  result[i] = mean(CommuteAtlanta$Distance[boot.sample])
}

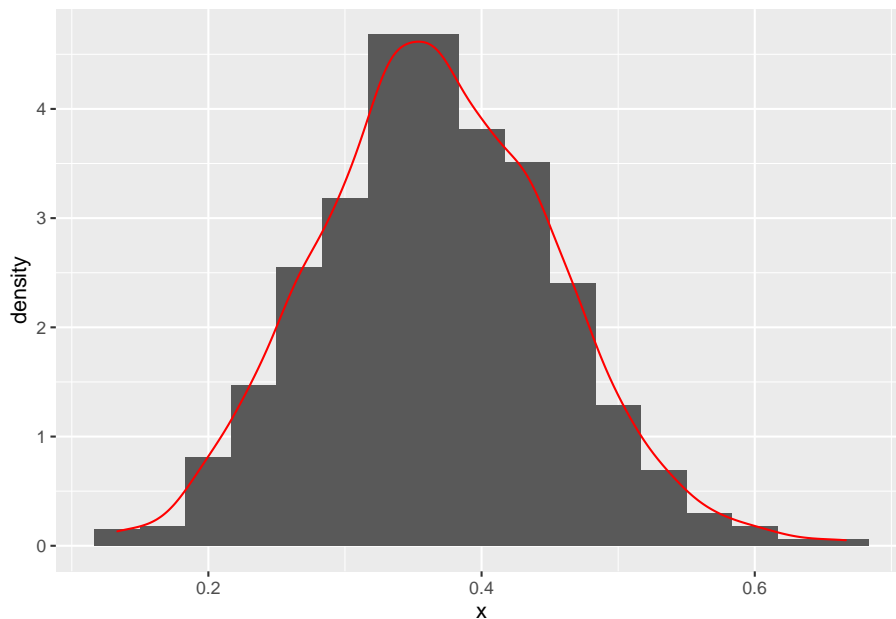
with(CommuteAtlanta, mean(Distance) + c(-1, 1) * 2 * sd(result))
```

```
## [1] 16.95361 19.35839
```

So far, we have dealt with means. How about proportions? Remember that, when calculating means, it starts with a single column of data to calculate the mean. Similarly, when calculating proportions, you can just use a single column

of data.

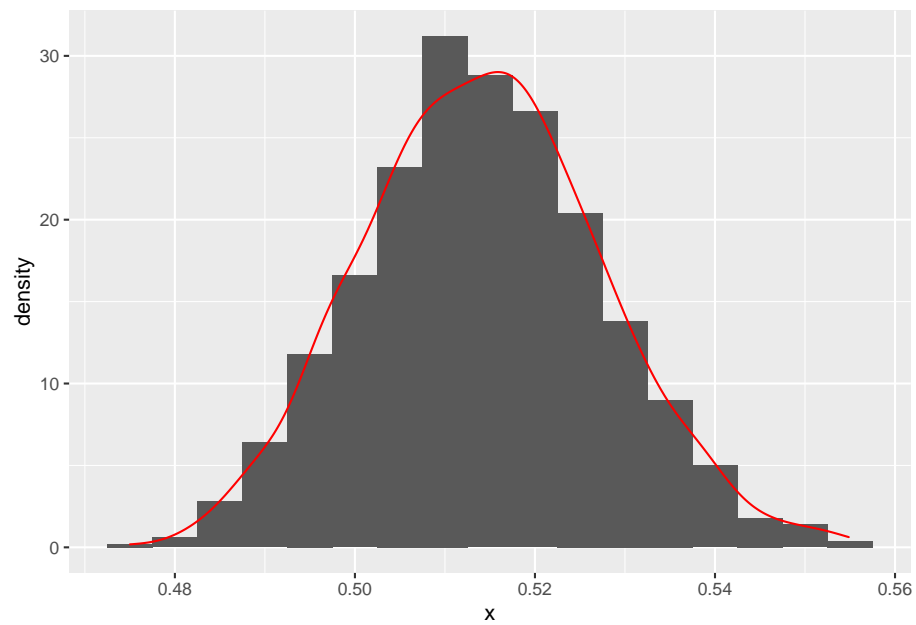
```
reeses = c(rep(1, 11), rep(0, 19))
reeses.boot = boot.mean(reeses, 1000, binwidth = 1/30)
```



```
## [1] 0.1929006 0.5404327
```

However, if we have 48 students (i.e., 48 observations) and thus we have a bigger sample. However, how can we do re-sampling? Based on the note, it is kind of simple. They group them together and then resample from it. Note that, when they re-sampling, the programming do not distinguish the difference between 48 observations. But just combined them as a single column (741+699=1440), and then generate a very long column (1440 times 1000) and then reshape it into a matrix (1440 time 1000). This is the basic logic of the boot.mean function.

```
reeses = c(rep(1, 741), rep(0, 699))
reeses.boot = boot.mean(reeses, 1000, binwidth = 0.005)
```



```
## [1] 0.4881597 0.5410070
```

After having a basic idea of bootstrapping, we can then use the package of boot.