SEM and R

Bill

2021-04-30

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**
5.1 Warning 17
5.2 Introduction 17
5.3 Normal distribution, SD, SE 19
5.4 Sample function 20
5.5 Proportion 21
5.6 boot package 22
5.7 Concept of Percentile 23
5.8 Bootstrapping for correlation interval 26

6 Poisson Regression **31**
6.1 Use R for mediation 32

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 engogenous 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 obseved 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 endogeous).

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 exogeous variables to endogeous variables (can be observed or latent).

2.1.10 Regerssion path

Regression path: A path between exogeous and endogeous 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., $f =\sim NA * q$).

$a*$ **lables 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 calcuates 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.

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

```
m1a <- lm(read ~ motiv, data=dat)
(fit1a <-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
    ppsych ~~ 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
```

```
##   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

5.1 Warning

Warning:

This page is for my own personal study purpose. Distribution is prohibited.

5.2 Introduction

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 X 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).

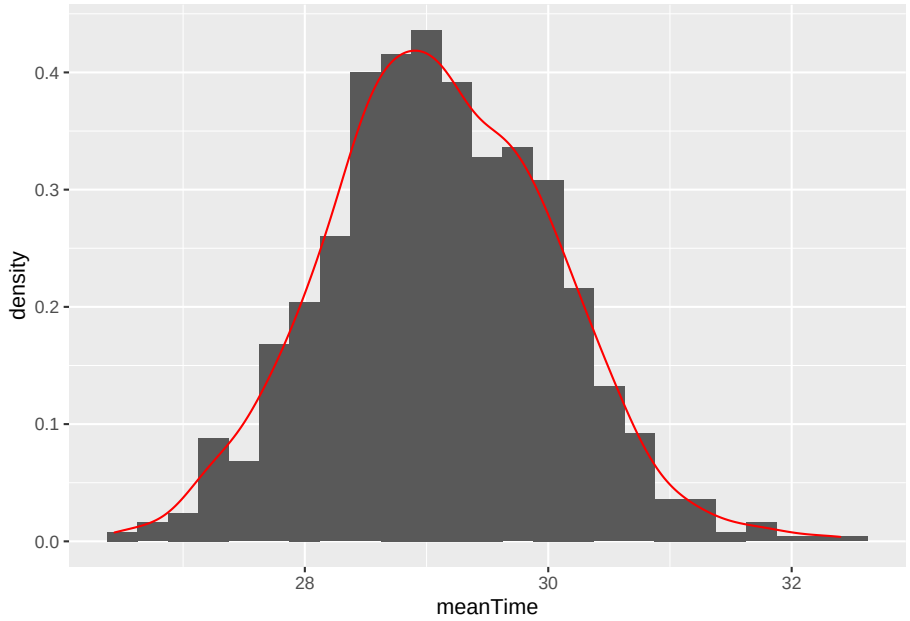
```
B = 1000
n = nrow(CommuteAtlanta)
boot.samples = matrix(sample(CommuteAtlanta$Time, size = B * n, replace = TRUE),
                      B, n)
```

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] 0.932219
```

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

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

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

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

5.3 Normal distribution, SD, SE

Note, if we do not use bootstraping, 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.

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

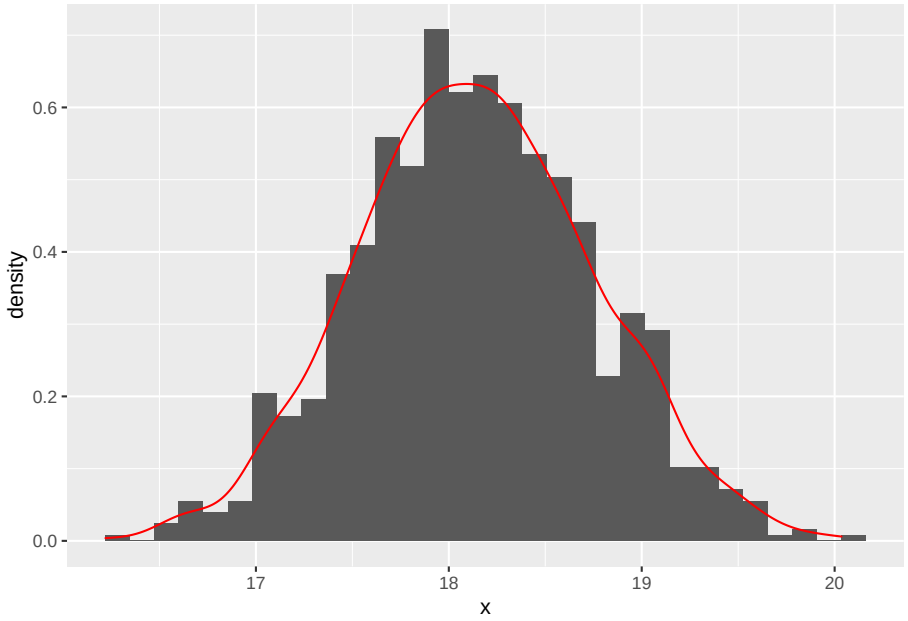
Note that, in the following, the author used 2 times SE to calculate the CI. The relationship between SD and SE:

"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] 16.93813 19.37387
```

5.4 Sample function

To understand the function of sample in R.

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

```
##  [1] 20  1  3 15  5  3  5 13 16 15 20  5  5  3  6 17  6  7  3  4
```

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)
```

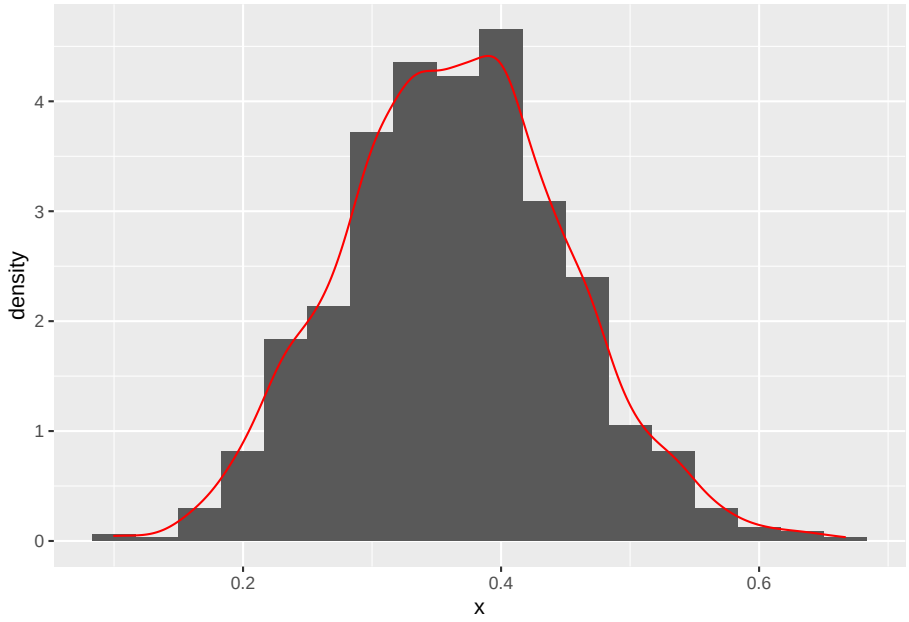
```
{
boot.sample = sample(n, replace = TRUE)
result[i] = mean(CommuteAtlanta$Distance[boot.sample])
}

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

```
## [1] 16.87146 19.44054
```

5.5 Proportion

So far, we have dealed with means. How about porpotions?Remember that, when calculating means, it starts with a single column of data to calculate the mean. Similarly, when calculating porpotions, 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)
```

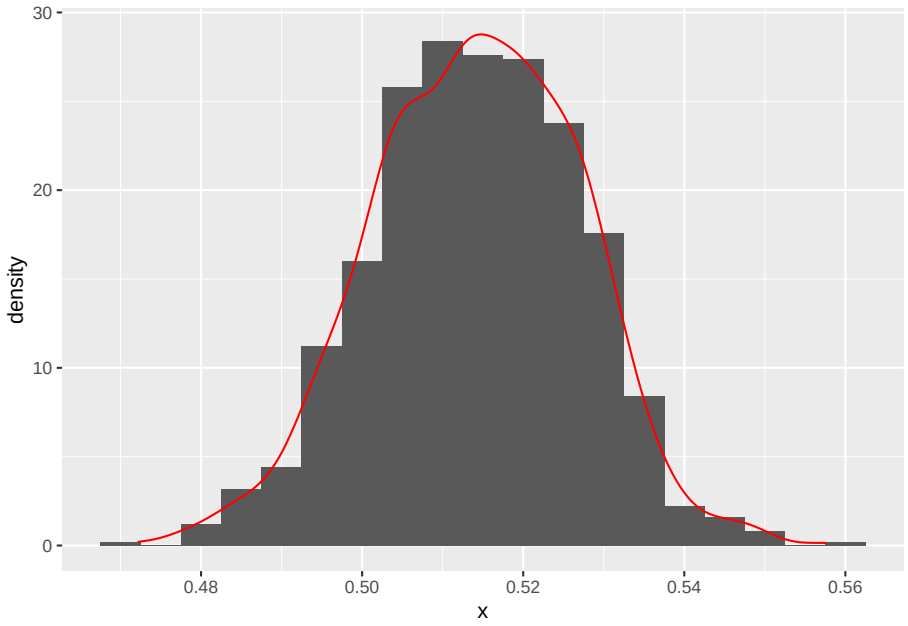



```
## [1] 0.1925558 0.5407775
```

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] 0.4889833 0.5401834
```

5.6 boot package

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

```
library(boot)

data(CommuteAtlanta)

my.mean = function(x, indices)
{
return( mean( x[indices] ) )
}

time.boot = boot(CommuteAtlanta$Time, my.mean, 10000)

boot.ci(time.boot)
```

```
## Warning in boot.ci(time.boot): bootstrap variances needed for studentized
```

```
## intervals
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = time.boot)
##
## Intervals :
## Level      Normal              Basic
## 95%   (27.29, 30.93 )   (27.24, 30.90 )
##
## Level     Percentile            BCa
## 95%   (27.32, 30.98 )   (27.43, 31.10 )
## Calculations and Intervals on Original Scale
```

5.7 Concept of Percentile

```
require(Lock5Data)
data(ImmuneTea)
tea = with(ImmuneTea, InterferonGamma[Drink=="Tea"])
coffee = with(ImmuneTea, InterferonGamma[Drink=="Coffee"])
tea.mean = mean(tea)
coffee.mean = mean(coffee)
tea.n = length(tea)
coffee.n = length(coffee)



B = 500
# create empty arrays for the means of each sample
tea.boot = numeric(B)
coffee.boot = numeric(B)
# Use a for loop to take the samples
for ( i in 1:B )
  {
tea.boot[i] = mean(sample(tea,size=tea.n,replace=TRUE))
coffee.boot[i] = mean(sample(coffee,size=coffee.n,replace=TRUE))
}

boot.stat = tea.boot - coffee.boot
boot.stat
```

```
##   [1]  4.6454545 10.3454545 19.7909091 29.0454545 12.0000000 17.4909091
##   [7] 16.0090909 24.6363636 18.0636364 24.2000000 12.8000000 10.1545455
```

```
##  [13] -2.4181818  6.2090909 11.7090909 14.8545455 21.9000000 13.4454545
##  [19]  2.8818182  7.1454545 18.1818182 15.1000000 23.0363636 16.5363636
##  [25] 25.4272727 15.5000000 11.4090909 29.3818182 20.6636364 14.1818182
##  [31]  4.7454545 27.3818182 15.4272727 22.6545455 12.2727273  0.6636364
##  [37] 13.8818182 -1.2727273 14.3181818  8.0363636 22.3545455 26.1545455
##  [43] 29.6090909 24.4636364 26.3818182 26.2818182 14.8454545  4.9000000
##  [49] 17.5818182 22.9545455 15.5181818  9.0727273  3.1363636 14.6636364
##  [55] 22.5181818 23.0545455 17.7000000 25.9818182 13.2272727 14.2818182
##  [61] 27.2181818  6.8181818 22.1272727 25.7454545 15.0727273 33.8636364
##  [67]  7.7363636 18.8000000 22.5636364  8.6272727  4.7727273 19.7818182
##  [73] 13.1000000  9.2818182 18.7272727 12.4000000 13.0818182 26.6272727
##  [79] 22.8090909 22.3545455 18.0090909 23.7090909 10.8818182 11.0272727
##  [85] 14.3545455  6.7818182 32.4545455 19.3454545  9.8090909 19.7272727
##  [91] 11.4000000 14.7909091 19.2636364 16.6363636 18.6181818 17.3545455
##  [97] 19.4090909 36.1181818 29.5090909 20.9636364 29.6454545  5.6090909
## [103] 34.9272727 15.1272727 19.6727273 15.9363636 18.3636364 -7.3454545
## [109] 17.7727273 24.1000000 11.5545455 25.3363636 13.5181818 20.9727273
## [115] 18.2818182 22.0818182 36.8636364 31.7090909 23.6181818 18.1727273
## [121] 29.1454545 -5.8363636 12.3000000 36.4454545 19.1181818  8.3454545
## [127] 19.1181818 16.9909091 27.1636364 15.4454545 -0.4363636 16.0454545
## [133] 28.6636364  3.4818182  7.1909091 10.8181818 15.3909091 14.4636364
## [139] 13.8818182 12.0363636 17.2636364 31.0000000 21.5545455 23.3272727
## [145]  5.6363636  9.7000000 22.4454545 12.3818182 21.5363636 19.2363636
## [151] 18.2909091 13.3090909 20.6272727  0.3818182 16.4545455 14.9909091
## [157] 14.3454545 27.3090909 12.0454545 18.8727273 18.1000000  6.3181818
## [163]  9.8000000 18.9090909 19.2454545 23.3818182  6.7909091  9.4454545
## [169] 13.8545455 25.1818182 13.3454545 12.2454545  8.9909091 22.1454545
## [175] 25.0909091 16.9090909 22.7818182  8.8727273 30.0454545 33.3818182
## [181] 14.8454545 13.4727273 30.3454545 25.7909091 13.7727273 27.1636364
## [187]  3.3090909 14.0363636 16.4636364 12.3363636 26.7909091 21.9090909
## [193]  5.6090909 16.3272727 13.9181818 16.4818182 14.4818182 12.2272727
## [199] 28.1363636 19.0090909 15.0000000  5.6181818 15.4090909 14.8272727
## [205] 23.0727273 11.8272727 26.5909091 27.8454545 19.2000000 17.4818182
## [211] 33.2090909 12.4636364 18.4818182  8.0090909  9.9181818 18.5909091
## [217]  6.1636364 14.9090909 12.8818182 22.2181818 17.7727273 18.6181818
## [223] 11.3272727 18.1636364 23.1545455 23.2909091 15.4181818 12.0090909
## [229] 23.4090909  6.6818182 19.2363636 19.0181818 17.2636364 24.9636364
## [235]  7.1727273 24.9363636  9.5454545 29.5818182 38.2272727 12.4545455
## [241] 16.0363636 10.9090909 17.4727273 24.3909091 23.2363636 14.2363636
## [247] 15.1181818 23.5454545 24.4636364 22.7454545 13.5090909 20.5909091
## [253] 17.7454545 23.6545455 15.2090909 16.9636364 30.2454545 21.3909091
## [259] 18.3727273 19.3454545 27.9818182  9.0090909 34.0545455 27.2909091
## [265] 17.9363636 34.1545455 16.0454545 16.5000000 13.0454545 14.5818182
## [271] 20.1363636 11.5727273 13.5727273 -8.2818182 35.0000000 13.5727273
## [277] 24.2272727 26.4363636 25.1272727 24.6090909 14.4727273  9.1272727
## [283]  8.1545455 -0.2545455 16.9272727 23.0272727 21.9818182 12.6363636
```

```
## [289] 28.4272727 32.5000000 25.0363636 11.1636364 12.4545455  1.9909091
## [295]  8.1363636  0.9000000 16.8909091 20.0000000 27.8363636 23.3909091
## [301] 14.5727273 16.3181818 21.4090909  2.9272727 18.5363636 26.8363636
## [307] 11.0727273 17.3727273 14.2909091 18.1818182 21.7909091  5.3272727
## [313] 30.6181818 28.6454545 18.6727273 25.7272727 25.8181818 17.0454545
## [319] 21.5363636 10.7363636 -1.8545455  6.8181818  6.6000000 20.9454545
## [325] 30.3727273 12.1727273 27.3090909 18.3363636 16.0636364 25.0000000
## [331] 16.1454545 12.1636364 25.8454545 10.3454545  5.0909091 18.8090909
## [337] 16.4545455 12.2727273 15.2000000 14.0727273 13.0636364 18.3454545
## [343]  8.2000000 18.4909091 21.5181818 13.0272727 15.7363636 20.9090909
## [349] 26.7454545 11.6363636 23.3363636 14.2636364 32.6545455 27.9727273
## [355] 21.7545455  6.7090909 18.5909091 12.7636364  7.0545455 28.2909091
## [361] 21.0363636 23.7909091 26.3090909 20.7818182  8.9545455 10.5181818
## [367] 18.8636364 25.3636364 14.5454545 14.8727273 14.4454545 24.6636364
## [373] 19.7454545 17.7090909 25.9727273 23.5727273 29.0000000 23.4545455
## [379]  0.1909091 18.7090909 22.7727273 16.7818182 21.6272727  9.5636364
## [385] 24.3363636 15.5909091 28.2090909 11.6636364 18.8545455  1.5818182
## [391] 17.7909091  5.4636364 15.2818182 18.2181818 16.9909091 23.0454545
## [397] 24.1363636 19.1363636 24.2181818 22.3181818 29.2090909 20.9727273
## [403] 22.6727273 14.4727273 10.9636364 20.5000000  9.2636364  8.8000000
## [409] 22.0636364 28.6545455 19.4363636  8.9454545 15.7636364 31.1272727
## [415]  8.4727273 10.7727273 29.4272727 17.9818182  7.4363636 16.6363636
## [421] 12.2363636 30.2909091  9.3454545 17.8272727 13.4363636 18.5454545
## [427] 12.9636364  7.1090909 19.8363636 18.0090909 25.0363636 18.1454545
## [433]  7.2090909 -4.4363636 32.6818182 14.4000000 26.0181818 13.9090909
## [439] 23.0000000 15.5000000 27.7818182 26.2727273 15.1636364 20.3454545
## [445]  8.4000000 23.2636364 23.1454545 18.6727273  5.5454545 23.0363636
## [451] 26.4000000 18.0636364 14.5454545 33.4545455 15.1818182 18.4272727
## [457] 20.7909091 17.6818182 23.4000000 18.0909091 15.8636364  6.2727273
## [463] 31.0818182 19.8818182 24.2000000 23.9181818  8.4000000 23.5545455
## [469]  7.8818182  8.2000000 20.6636364 16.5454545  2.3636364 22.0454545
## [475] 17.4000000 14.3727273 26.3909091 15.9909091 10.0363636 22.4545455
## [481] 24.4545455 20.7727273 26.8545455 29.0909091 20.4727273 23.7363636
## [487]  1.1636364 15.2909091 14.6363636 29.8636364 20.7272727 22.6727273
## [493] 32.8545455 23.2818182 21.7363636 20.8363636 18.4818182  1.7454545
## [499] 12.3454545  0.7181818
```

```
# Find endpoints for 90%, 95%, and 99% bootstrap confidence intervals using percentiles.

# 90%:  5% 95%
quantile(boot.stat,c(0.05,0.95))
```

```
##        5%       95%
##  4.740455 30.247727
```

```
# 95%: 2.5% 97.5%
quantile(boot.stat,c(0.025,0.975))
```

```
##       2.5%      97.5%
##  0.8045455 32.7725000
```

```
# 99%:  0.5% 99.5%
quantile(boot.stat,c(0.005,0.995))
```

```
##      0.5%     99.5%
## -5.143364 36.283455
```

5.8 Bootstrapping for correlation interval

Some data and code are from: https://blog.methodsconsultants.com/posts/understanding-bootstrap-confidence-interval-output-from-the-r-boot-package/

```
data_correlation<-read.csv("data_correlation.csv",fileEncoding="UTF-8-BOM")

data_correlation
```

```
##    Student LSAT  GPA
## 1        1  576 3.39
## 2        2  635 3.30
## 3        3  558 2.81
## 4        4  578 3.03
## 5        5  666 3.44
## 6        6  580 3.07
## 7        7  555 3.00
## 8        8  661 3.43
## 9        9  651 3.36
## 10      10  605 3.13
## 11      11  653 3.12
## 12      12  575 2.74
## 13      13  545 2.76
## 14      14  572 2.88
## 15      15  594 2.96
```

```
cor.test(data_correlation$LSAT,data_correlation$GPA)
```

```
##
##  Pearson's product-moment correlation
##
## data:  data_correlation$LSAT and data_correlation$GPA
## t = 4.4413, df = 13, p-value = 0.0006651
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.4385108 0.9219648
## sample estimates:
```

```
##       cor
## 0.7763745
```

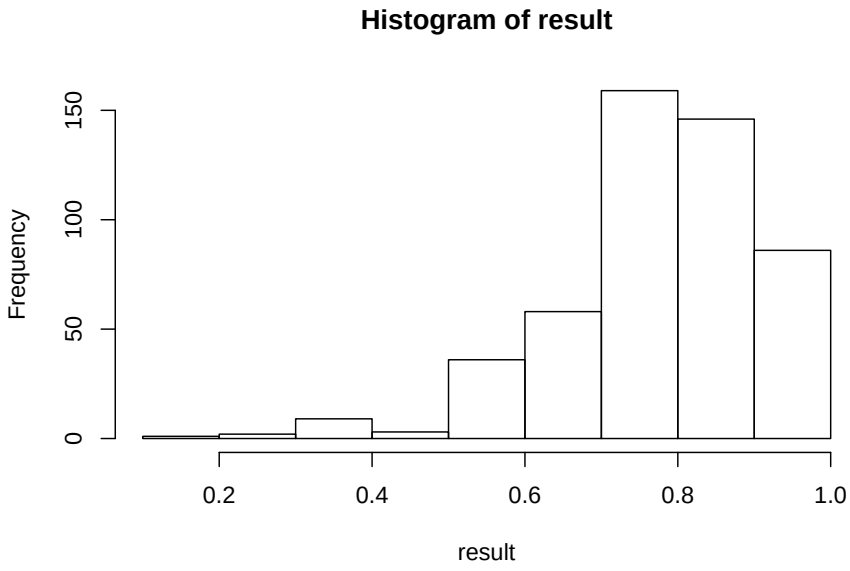
In the following, I will write my own code to execute the bootstrapping. I set the bootstrapping number only 500, for illustrative purposes. As we can see, the distribution is not symmetrical.

As we can see, the quantile result and c(-1, 1) X 2 are not the same, as the latter assumes symmetrical distribution. However, based on the histogram, we know it is not the case. Thus, quantile would be more appropriate. You can compare the result with that from the boot function.

```
n_row = nrow(data_correlation)
n_row
```

```
## [1] 15
```

```
set.seed(12345)

B = 500
result = rep(NA, B)
for (i in 1:B)
{
boot.sample = sample(n_row, replace = TRUE)
result_temp = cor.test(data_correlation[boot.sample,]$LSAT,data_correlation[boot.sample,]$GPA)
result[i]=result_temp$estimate
}
hist(result)
```




```
# 95%: 2.5% 97.5%
quantile(result,c(0.025,0.975))
```

```
##      2.5%     97.5%
## 0.4369293 0.9556859
```

```
sd(result)
```

```
## [1] 0.1342631
```

```
mean(result) + c(-1, 1) * 1.96 * sd(result)
```

```
## [1] 0.5107704 1.0370816
```

```
cor(data_correlation$LSAT,data_correlation$GPA)
```

```
## [1] 0.7763745
```

```
cor(data_correlation$LSAT,data_correlation$GPA)+ c(-1, 1) * 1.96 * sd(result)
```

```
## [1] 0.5132189 1.0395301
```

```
# why add 0.005? Not sure. The following is from the webpage. Later note: please refer

0.776+0.005+c(-1, 1) * 1.96 * 0.131
```

```
## [1] 0.52424 1.03776
```

In the blog mentioned above, the author used the boot function in R. For the logic of basic interval, please refer to: https://blog.methodsconsultants.com/

posts/understanding-bootstrap-confidence-interval-output-from-the-r-boot-package/

```
library(boot)

get_r <- function(data, indices, x, y) {
  d <- data[indices, ]
  r <- round(as.numeric(cor(d[x], d[y])), 3)
  r}

set.seed(12345)

boot_out <- boot(
  data_correlation,
  x = "LSAT",
  y = "GPA",
  R = 500,
  statistic = get_r
)

boot.ci(boot_out)
```

```
## Warning in boot.ci(boot_out): bootstrap variances needed for studentized
## intervals
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 500 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot_out)
##
## Intervals :
## Level      Normal              Basic
## 95%   ( 0.5247,  1.0368 )   ( 0.5900,  1.0911 )
##
## Level     Percentile            BCa
## 95%   ( 0.4609,  0.9620 )   ( 0.3948,  0.9443 )
## Calculations and Intervals on Original Scale
## Some BCa intervals may be unstable
```

Chapter 6

Poisson Regression

There are some other sources of website

https://rdrr.io/github/sta303-bolton/sta303w8/f/inst/rmarkdown/templates/philippines/skeleton/skeleton.Rmd

```
fHH1 <- read.csv("https://raw.githubusercontent.com/proback/BeyondMLR/master/data/fHH1.csv")

head(fHH1)
```

```
##   X     location age total numLT5                          roof
## 1 1 CentralLuzon  65     0      0 Predominantly Strong Material
## 2 2  MetroManila  75     3      0 Predominantly Strong Material
## 3 3  DavaoRegion  54     4      0 Predominantly Strong Material
## 4 4      Visayas  49     3      0 Predominantly Strong Material
## 5 5  MetroManila  74     3      0 Predominantly Strong Material
## 6 6      Visayas  59     6      0 Predominantly Strong Material
```

$$log(\lambda_X) = \beta_0 + \beta_1 X$$

$$log(\lambda_{X+1}) = \beta_0 + \beta_1 (X+1)$$

Thus,

$$log(\lambda_{X+1}) - log(\lambda_X) = (\beta_0 + \beta_1 (X+1)) - (\beta_0 + \beta_1 X)$$

Thus,

$$log(\frac{\lambda_{X+1}}{\lambda_X}) = \beta_1$$

Thus,

$$\frac{\lambda_{X+1}}{\lambda_X} = e^{\beta_1}$$

Note that, λ here is the mean. It is poisson regression, and the parameter is the mean. Thus, $\frac{\lambda_{X+1}}{\lambda_X} = e^{\beta_1}$ suggests the ratio change in the DV as the IV change in one unit.

$$log(\hat{\lambda}) = b_0 + b_1 Age$$

```
result_1 = glm(total ~ age, family = poisson, data = fHH1)
result_1
```

```
##
## Call:  glm(formula = total ~ age, family = poisson, data = fHH1)
##
## Coefficients:
## (Intercept)          age
##    1.549942    -0.004706
##
## Degrees of Freedom: 1499 Total (i.e. Null);  1498 Residual
## Null Deviance:       2362
## Residual Deviance: 2337  AIC: 6714
```

$$\frac{\lambda_{Age+1}}{\lambda_{Age}} = e^{\beta_1} = e^{-0.0047} = 0.995$$

But, what does it mean? It is a bit tricky. But, we can make some modification to help us understand.

$$\lambda_{Age+1} = 0.995\lambda_{Age}$$

$$\lambda_{Age+1} - \lambda_{Age} = 0.995\lambda_{Age} - \lambda_{Age} = -0.005\lambda_{Age}$$

Thus, we can understand that, the mean of household size difference by changing 1 unit of age (i.e., $\lambda_{Age+1} - \lambda_{Age}$) is $-0.005\lambda_{Age}$.

6.1 Use R for mediation

https://advstats.psychstat.org/book/mediation/index.php