SEM and R

Bill

2021-04-22

Contents

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 Introduction

Warning:

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

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).

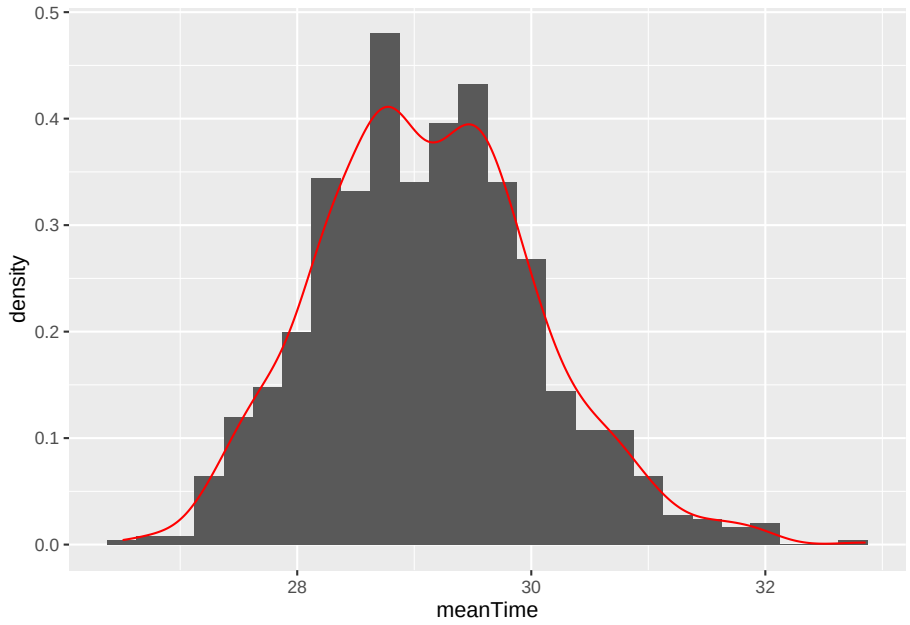
```
B = 1000
n = nrow(CommuteAtlanta)
boot.samples = matrix(sample(CommuteAtlanta$Time, size = B * n, replace = TRUE),
                      B, n)
```

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] 0.9389638
```

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

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

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

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

5.2 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.

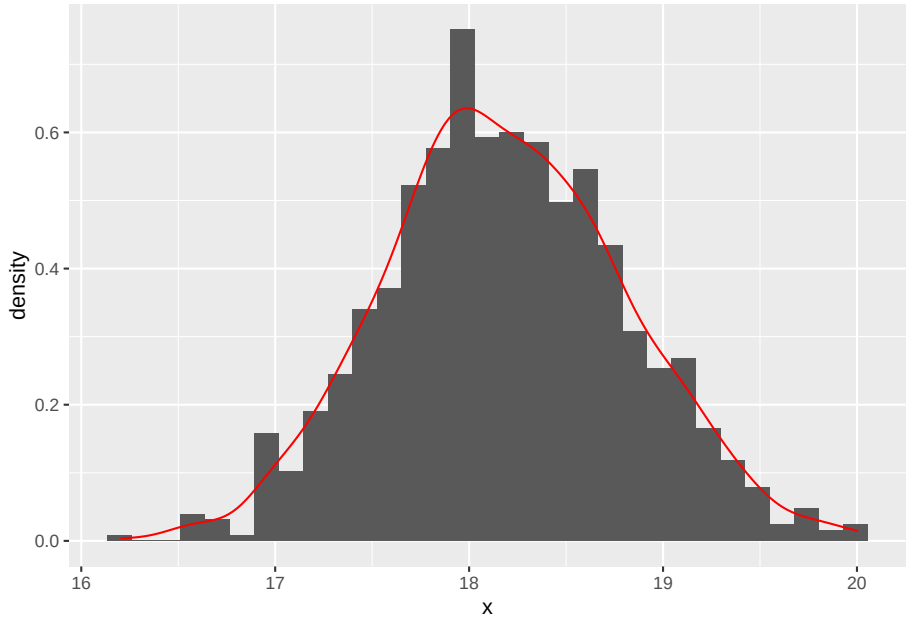
$$\overline{X} \pm Z\frac{S}{\sqrt{n}} = 29.11 \pm 1.96\frac{20.72}{\sqrt{500}} = 27.29, 30.93$$

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] 16.91637 19.39563
```

5.3 Sample function

To understand the function of sample in R.

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

```
##  [1] 18  5 16 20  3  9  2 17  1 20  4  5 13 18 14  1 13 19  9  9
```

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.

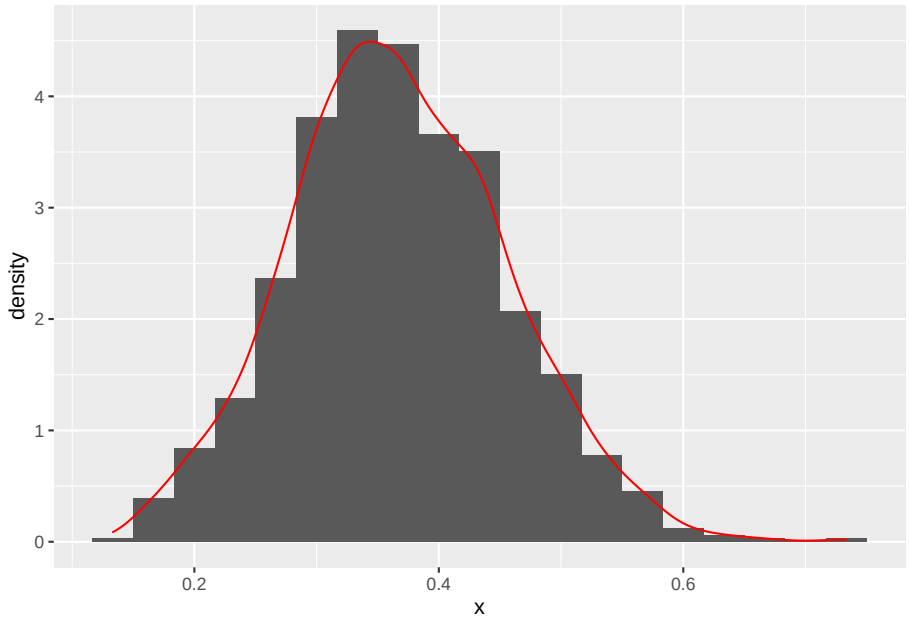
```
n = length(CommuteAtlanta$Distance)
B = 1000
result = rep(NA, B)
for (i in 1:B)
{
boot.sample = sample(n, replace = TRUE)
result[i] = mean(CommuteAtlanta$Distance[boot.sample])
}

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

```
## [1] 16.8511 19.4609
```

5.4 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.

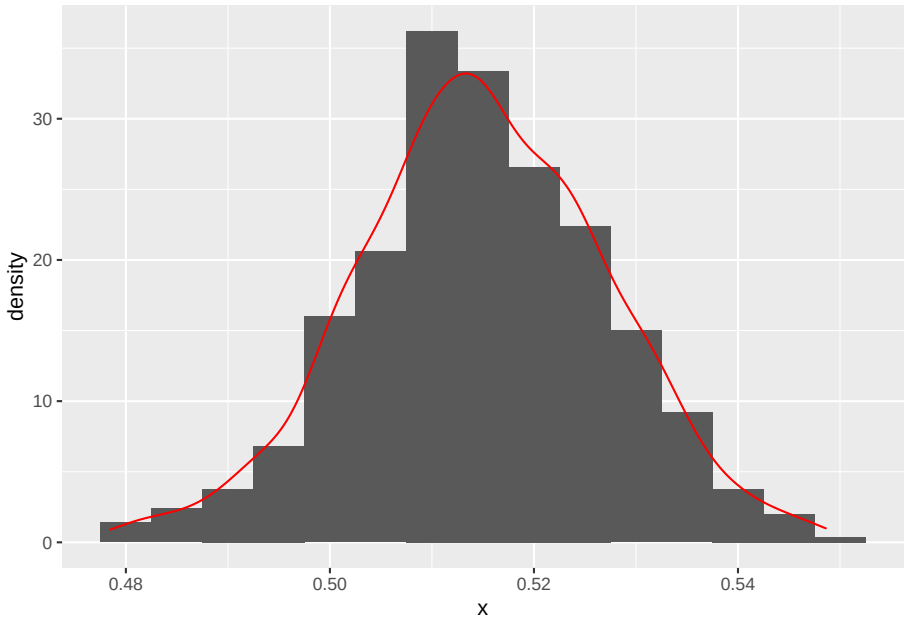
```
reeses = c(rep(1, 11), rep(0, 19))
reeses.boot = boot.mean(reeses, 1000, binwidth = 1/30)
```




```
## [1] 0.1901450 0.5431883
```

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] 0.4900437 0.5391230
```

5.5 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%   (27.27, 30.94 )   (27.20, 30.86 )
##
## Level     Percentile            BCa
## 95%   (27.36, 31.02 )   (27.45, 31.12 )
## Calculations and Intervals on Original Scale
```

5.6 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] 30.5818182 30.9363636 20.5909091 19.2636364 15.9181818 14.6090909
##   [7] 27.3818182 18.1090909 22.1818182 14.5454545 14.5090909 11.6272727
##  [13]  2.8545455 13.9181818 21.0636364  9.8636364 16.0272727 22.6272727
##  [19] 12.2000000  1.8363636 21.8272727 20.8181818 23.1727273 16.5090909
##  [25] 10.5909091  0.4181818 25.6272727 15.9636364 20.5090909  5.3818182
##  [31] 12.5545455 22.6545455 22.5181818 30.7181818  8.0000000 21.2272727
##  [37] 22.0363636 23.6454545 14.9636364  9.5727273 15.6181818  2.3181818
##  [43]  7.9818182  6.9909091 28.6818182  9.2181818 10.0000000 25.2363636
```

```
##  [49]  5.8727273  9.2727273 23.9181818 13.1000000  3.8454545 28.4818182
##  [55]  8.6636364 17.7818182  3.0363636 20.9363636 26.5545455  3.1363636
##  [61]  0.7000000 13.8909091  7.9000000 26.6545455 17.3090909  7.9909091
##  [67] 28.4363636 11.1454545 19.1818182 20.7454545 14.1000000 21.1181818
##  [73] 26.0363636 15.1636364 25.8272727 18.8181818 24.0727273 11.2545455
##  [79] 21.2181818 18.4363636 27.9272727 28.4818182 23.0090909 19.1636364
##  [85]  2.4909091 16.7818182  9.2727273 23.3363636 24.8636364 12.8090909
##  [91] 14.2727273 10.7363636 20.6818182 24.8909091 14.6272727 23.7727273
##  [97]  2.7454545 17.4454545  7.1000000 13.9545455 15.8636364  9.7272727
## [103] 26.3000000  2.0818182 11.0454545  6.1454545 21.2818182 16.0727273
## [109]  8.9090909  9.3454545 19.5272727  5.2454545 17.9636364 13.1818182
## [115] 29.9090909 21.4454545 30.7454545 16.2545455 23.9727273  9.4454545
## [121] 20.8545455 15.7545455  1.6090909  5.3000000 31.1181818 16.9363636
## [127] 25.1636364  8.5181818 11.9454545 32.3636364 23.7818182 19.7272727
## [133] 14.0090909 20.7090909 29.7636364 11.6000000 30.5363636 19.5272727
## [139] 30.6272727 20.0545455 17.9545455  8.9727273 13.0818182 13.8000000
## [145] 13.3909091 17.6454545 12.1090909 10.9454545 22.3727273 18.8454545
## [151]  4.1090909 18.1181818 18.8181818  9.4363636 12.9363636  8.3454545
## [157] 34.5454545  3.9000000 19.5090909 19.1090909 20.3727273 16.0000000
## [163] 30.5363636 27.3545455 13.9636364 19.7363636 11.3272727 24.2363636
## [169] 25.4727273 19.1000000  7.9818182 24.5636364  5.3727273 12.8545455
## [175]  8.4363636 11.8545455 28.6454545 30.4454545 14.3818182  1.4090909
## [181] 17.7363636 18.2000000 25.9909091  9.6727273 21.5272727 31.3454545
## [187]  6.9181818 12.9909091 28.2636364 26.2454545  5.9727273 13.4909091
## [193] 23.3727273 14.2727273 10.3727273 11.9363636  5.1272727 16.8363636
## [199] 18.2727273 33.4181818 20.3000000 15.1363636 10.0272727 16.4545455
## [205]  0.1818182 16.1181818 19.7181818 11.4181818  9.3363636 24.2818182
## [211]  6.3363636 16.3909091 13.8363636 17.3181818 23.0363636 12.9636364
## [217] 20.8090909 18.4090909  5.1636364 12.9181818 -6.8090909 26.2727273
## [223]  6.4272727 22.5090909 -3.8272727  8.6272727 38.7727273 27.8545455
## [229] 18.8272727 14.6090909 36.5636364  0.7818182 21.9363636 26.5090909
## [235]  8.2272727 26.3818182 16.8909091 13.6818182 23.5909091  5.0818182
## [241] 12.8454545 10.4000000 16.4272727 10.5909091 22.1454545 15.6000000
## [247] 12.1272727  2.9272727 21.2545455 23.2000000 15.0727273 25.9727273
## [253] 13.9454545  9.9818182  8.6090909 10.9545455 10.4363636 15.9181818
## [259] 17.1636364  5.2818182  9.7272727  3.4545455 14.4000000 26.8363636
## [265] 21.8818182  3.1818182 11.3454545 11.0454545 20.6000000 25.9272727
## [271]  4.6545455 16.7818182 12.4181818 19.2363636 15.7090909  4.9636364
## [277] 18.4363636  6.8727273 22.9363636 15.5909091 16.8363636 26.1181818
## [283]  8.0818182 11.7545455 17.8090909 27.7545455 13.5818182 22.5454545
## [289] 18.9000000 28.4545455 15.9090909 19.7000000 14.9454545  3.1272727
## [295] 17.7181818  3.7181818 28.0363636 13.4636364 15.6909091 12.5909091
## [301] 20.8454545  4.5090909 20.0000000 12.3454545 10.2363636  6.2545455
## [307]  9.0818182 23.7454545 11.1727273 12.9727273  5.1181818 12.1818182
## [313] 28.0000000 24.8909091 16.5363636 12.3363636 17.8818182 11.3727273
## [319] 16.9636364 12.3363636 18.9090909 13.1818182 17.5727273  5.1090909
```

```
## [325] 34.0181818 29.5090909 28.3090909 12.7090909 13.4363636  9.9454545
## [331] 18.9000000  3.1636364 11.0818182 22.6272727 11.9181818  8.2545455
## [337] -9.1000000 15.3181818  9.3454545 18.1727273 21.5454545 16.6000000
## [343] 12.8727273 16.0363636 23.7272727 11.8272727 15.7909091 12.1454545
## [349] 21.0545455 27.6909091 18.8818182 16.4181818 22.6090909  3.5272727
## [355]  6.8181818 13.8000000 11.4545455 14.2545455 -1.2181818 12.5909091
## [361] 24.2545455 -2.6727273 10.1909091 23.9545455 21.2818182 19.4000000
## [367] 21.5090909  9.6272727 14.6545455 14.0363636 23.3272727 15.7181818
## [373] 13.7272727 12.6454545 26.8454545  5.1000000 15.2181818  6.1363636
## [379] 19.0090909  3.8272727 19.9727273 -8.6727273 20.6727273 14.2545455
## [385] 24.3272727 26.1000000 17.2454545 27.3090909  7.3363636 19.5727273
## [391] 22.7818182 13.9000000 20.8818182 19.6000000 23.0272727  6.1636364
## [397] 18.7090909 18.8818182 15.0545455 11.7545455 21.4181818 18.5818182
## [403] 20.4818182  6.6636364 24.8636364 33.8818182 25.2090909 13.8636364
## [409] 19.2090909 25.3727273 21.9272727 17.2090909 24.6272727 33.8909091
## [415]  9.1090909 26.5909091 22.2090909 26.0818182 22.4818182 17.7636364
## [421] 28.5727273 19.4363636  8.2090909  6.8909091 11.2727273 19.3909091
## [427] 11.5636364 33.0909091 37.3090909  9.2181818 17.1000000 11.3909091
## [433] 21.9727273 11.2909091 19.2272727 14.0636364 15.0818182 14.2818182
## [439] 37.6818182 15.9454545 20.6363636 24.7909091 20.9090909 25.0181818
## [445] 10.7909091 16.3818182  7.6454545 28.2636364 20.5181818 27.7636364
## [451]  3.8727273 10.5090909 10.1090909 19.0454545 21.9909091 18.9363636
## [457]  1.1818182 19.3363636 13.9000000 15.5636364 16.5727273 16.3090909
## [463] 25.5545455 18.0454545 13.4181818 23.6090909 18.8545455  8.9363636
## [469] 18.3363636 17.4181818  5.8090909 26.8727273  1.6454545  7.9181818
## [475]  9.8818182 -1.1727273 22.6363636 18.3181818  7.3636364 12.6181818
## [481] 23.3000000 14.6090909 12.6000000 12.2181818 19.7636364 12.1909091
## [487] 18.0545455 34.5545455 23.0818182 23.4727273  3.5090909 17.7545455
## [493] 14.7363636  1.6727273 28.3363636 10.6454545  1.1181818 15.5818182
## [499] 15.9818182  7.1000000
```

```
# 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%
##  3.122727 28.723182
```

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

```
##      2.5%     97.5%
##  1.289773 31.237500
```

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

```
##      0.5%     99.5%
## -5.333091 36.940091
```

5.7 Use Boot for correlation

The following code is from: https://blog.methodsconsultants.com/posts/understanding-bootstrap-confidence-interval-output-from-the-r-boot-package/

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

```
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
```

5.8 Use R for mediation

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