SEM and R

Bill

2021-04-22

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 Use Boot for correlation . . . 26
5.9 Use R for mediation . . . 27

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.

```r
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
```

```r
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)

```r
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).

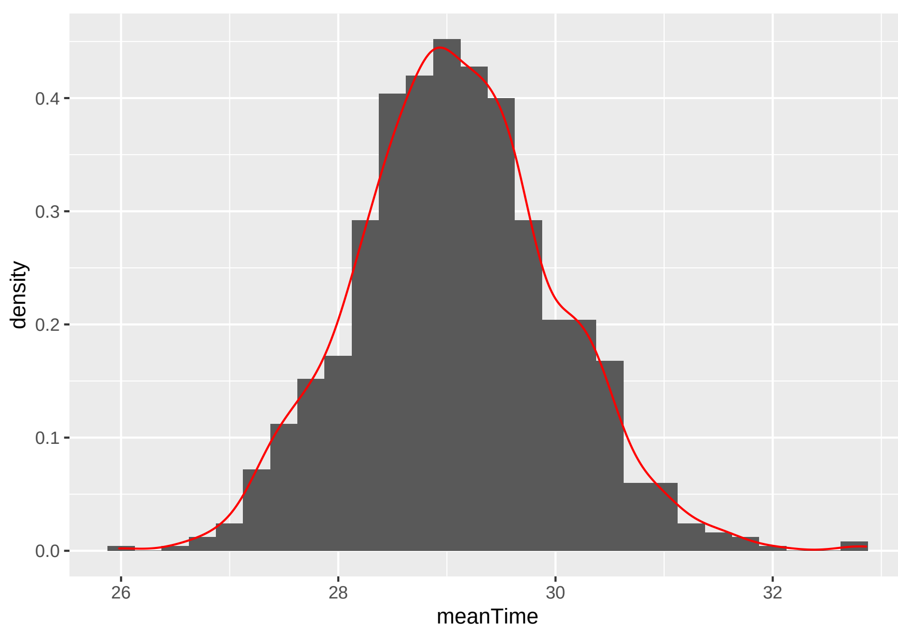
```
B = 1000
n = nrow(CommuteAtlanta)
boot.samples = matrix(sample(CommuteAtlanta$Time, size = B * n, replace = TRUE),
                      B, n)
```

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] 0.926212
```

```
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.

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

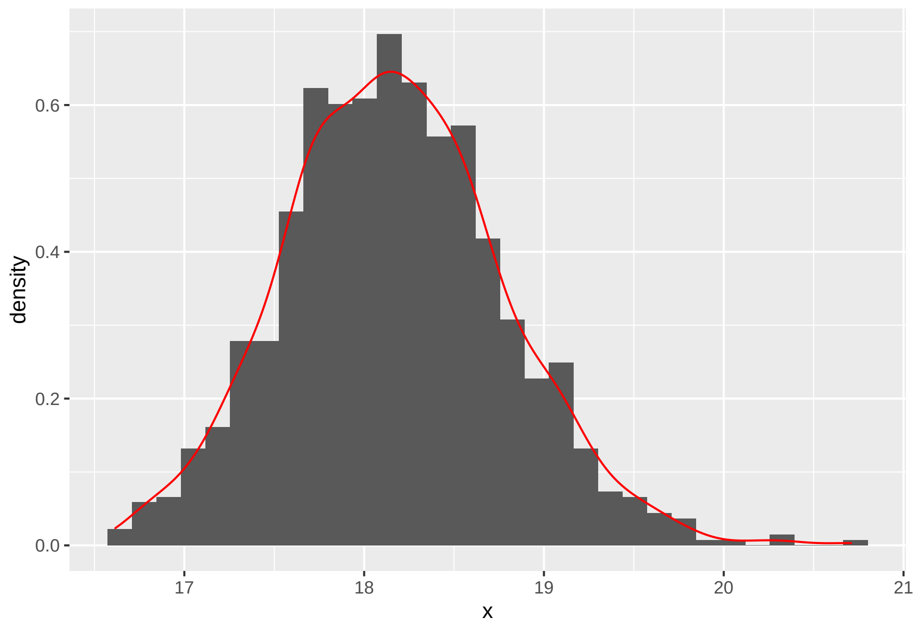
Note that, in the following, the author used 2 times SE to calculate the CI. The relationship between SD and SE:

"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] 16.94029 19.37171
```

5.4 Sample function

To understand the function of sample in R.

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

```
##  [1]  8  5  5  1  2  4 12 10 18 17  4 16 16  8 13  4  3  3 19  8
```

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)
```

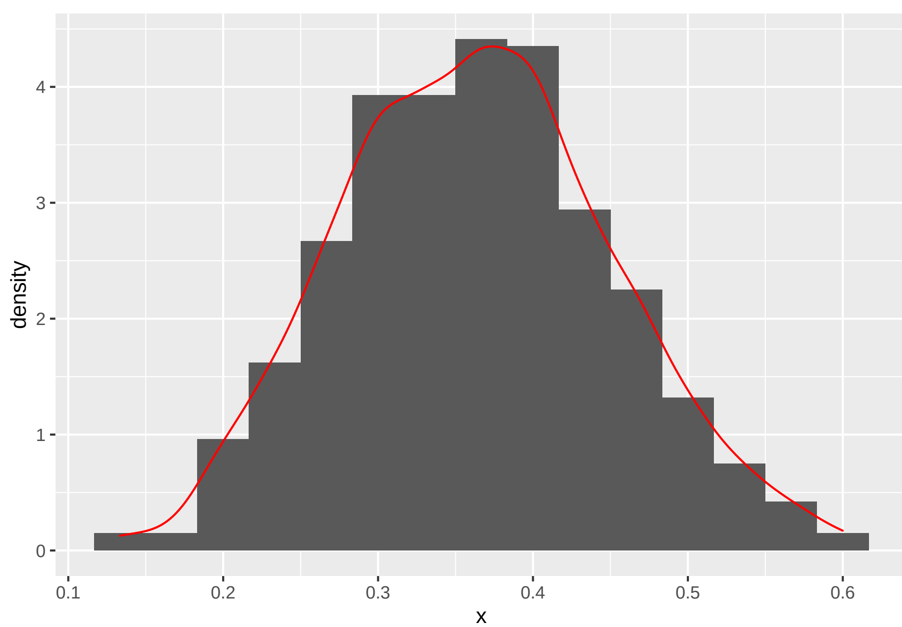
```
{
boot.sample = sample(n, replace = TRUE)
result[i] = mean(CommuteAtlanta$Distance[boot.sample])
}

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

```
## [1] 16.9046 19.4074
```

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.

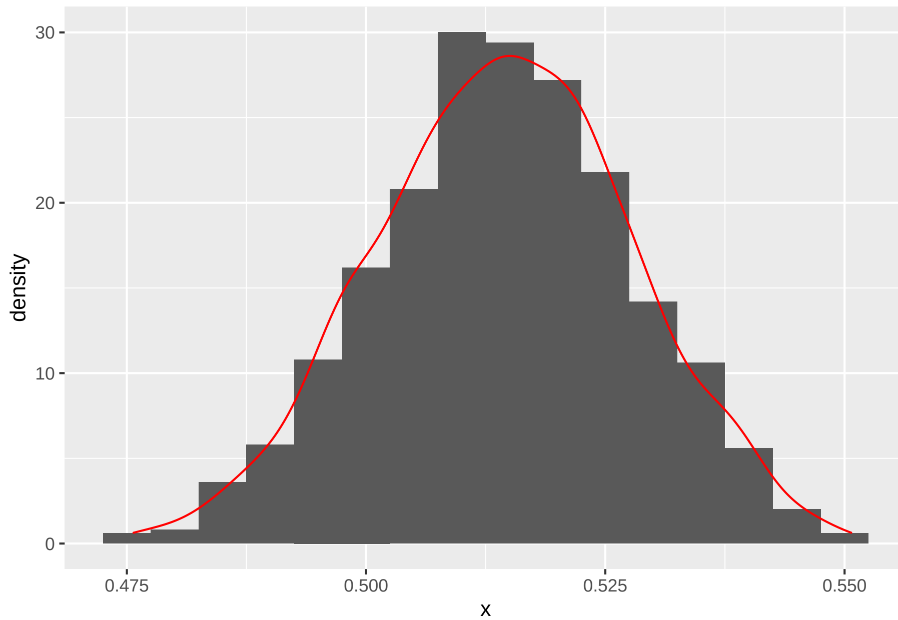
```
reeses = c(rep(1, 11), rep(0, 19))
reeses.boot = boot.mean(reeses, 1000, binwidth = 1/30)
```




```
## [1] 0.1924560 0.5408773
```

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] 0.4879056 0.5412611
```

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%   (27.30, 30.92 )   (27.29, 30.87 )
##
## Level     Percentile            BCa
## 95%   (27.35, 30.93 )   (27.43, 31.03 )
## 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]  16.0727273  32.3727273  26.9818182  26.3181818  30.3909091
##   [6]  17.4818182  17.0818182  22.9545455  26.3545455  25.8545455
```

```
##  [11]  14.6272727  17.6363636   5.0545455  21.6181818  21.2636364
##  [16]   4.1272727  17.3000000  20.8181818   8.7181818  14.3090909
##  [21]   9.5000000  17.9454545  20.9909091  16.0090909  10.1363636
##  [26]   5.5272727  30.5727273  15.0636364  -2.7636364   2.5000000
##  [31]  23.9636364  15.6636364  27.8272727  27.7545455   6.6727273
##  [36]  17.6181818  25.0636364  19.4818182  13.1818182  21.0090909
##  [41]   5.3545455  10.7090909  17.9545455  14.8454545  14.7727273
##  [46]  33.1181818  17.8363636  26.5181818  12.0727273  14.1272727
##  [51]  20.0727273  16.8181818  10.3727273  26.2818182  26.5636364
##  [56]  15.5636364  16.1818182  27.7454545   9.5272727  14.5909091
##  [61]   4.4454545   4.4454545  18.3909091   0.3818182  16.2545455
##  [66]  15.9454545  12.0909091   5.1090909  18.9727273  21.2909091
##  [71]  37.9727273  22.2818182  25.9090909  20.5636364  27.5727273
##  [76]  23.3727273  22.5909091  16.8090909  26.3181818   2.5090909
##  [81]  18.0727273  14.3727273  15.8272727  18.1454545  28.1818182
##  [86]  16.4545455  16.3818182  11.0727273  24.9818182   3.3909091
##  [91]   5.0818182  23.2000000  26.9545455  13.2636364  13.3727273
##  [96]  18.4181818  28.9181818  20.9000000  16.4181818  21.2909091
## [101]  10.6090909  17.6727273   8.2272727  14.3727273  35.0727273
## [106]   7.2636364  17.0272727  22.0272727  24.9090909  18.4727273
## [111]  20.7727273  13.7727273  24.5909091  16.2272727  21.1454545
## [116]  21.1363636   7.0454545  22.2363636   4.5727273  17.2272727
## [121]  30.2727273  18.1909091  -5.3000000  11.3000000   7.6818182
## [126]  13.5181818  19.4000000  16.9909091  26.1545455  12.4636364
## [131]  16.3545455  15.7727273   2.4545455  18.3636364   9.2363636
## [136]  12.8909091   3.9363636  10.2545455  18.2545455  23.1181818
## [141]  17.0272727  10.5636364  20.2636364  11.8181818  15.2090909
## [146]  17.9454545  22.2000000  12.7272727  14.1818182  17.8000000
## [151]  14.7909091  24.6090909  22.3090909  20.7909091  13.6000000
## [156]   6.9090909  25.3272727  31.3363636   1.5000000  21.9545455
## [161]  12.8000000  11.0181818  17.9818182  32.8909091  39.9363636
## [166]  29.0000000  12.3181818  30.3454545  12.4909091  11.2545455
## [171]  24.6818182  17.9727273  24.5090909   8.5818182  17.9090909
## [176]  25.8454545  32.6727273  27.7000000  19.9454545   9.8272727
## [181]  28.7090909  29.5545455  12.9818182  20.9454545  24.0636364
## [186]  22.9727273  24.6727273  10.4818182  16.5818182  12.0727273
## [191]   2.5454545  14.5090909  33.4818182  35.1090909  13.4818182
## [196]   9.0454545  15.6454545  24.0181818  18.7181818  19.6090909
## [201]  23.2818182  13.7090909  22.7272727  19.8000000   9.1090909
## [206]  14.4727273  -2.3363636  24.8181818  28.5363636  13.3363636
## [211]   8.0909091   0.4727273  12.8454545  32.1181818  18.8909091
## [216]  18.9818182   8.6818182  20.8727273  16.9454545   0.2181818
## [221]  23.9181818  15.5636364  22.1272727  16.3636364  28.9272727
## [226]  13.8727273  21.3272727   9.4454545  19.7818182  18.2909091
## [231]  16.4636364  18.8363636  18.4545455  20.9363636  24.8545455
## [236]  17.0272727  15.7181818  18.1000000  28.0909091  41.5272727
```

```
## [241]  20.4727273  31.7545455  16.5000000 -12.4454545  21.1363636
## [246]  17.4545455  10.3636364  14.6636364  13.8727273  17.0090909
## [251]  19.6363636  17.2272727  16.8545455  15.4636364  19.2545455
## [256]  14.9454545   2.6818182  28.6818182  20.8272727   8.7181818
## [261]  39.4272727  13.4272727  21.5090909  21.7818182  32.0272727
## [266]  25.7727273   6.7454545  25.8727273  20.4454545   4.8909091
## [271]  -5.2545455  19.3727273  15.2363636  17.6454545   8.4181818
## [276]  13.4181818  19.0272727  27.0909091  21.8000000  13.2090909
## [281]  15.4363636  11.7909091  12.8000000  23.1363636  15.4272727
## [286]  13.3363636  13.0909091   1.4454545  23.5636364  23.7727273
## [291]  -2.6272727  25.0727273  22.7909091  -2.1818182  24.4090909
## [296]  14.8454545  14.7909091  17.7363636  30.4545455   9.7272727
## [301]  15.4636364  18.9363636  12.9363636  13.2818182  12.8454545
## [306]  15.0272727  13.7272727  17.9818182   7.5818182  13.5272727
## [311]  12.6181818  20.4454545   9.1818182  30.9181818  25.2454545
## [316]  30.4727273  32.9181818  20.5363636  26.6000000  10.0909091
## [321]  26.4272727  19.3636364  12.4909091  21.9454545   4.5363636
## [326]  29.2727273  24.9181818  10.9272727  20.5000000  14.0090909
## [331]  12.3818182  12.0909091  31.3000000   8.0545455  17.4272727
## [336]  19.6454545   7.1272727  12.8000000  10.3000000  16.3363636
## [341]  23.0545455   8.9636364  18.3272727  10.4272727   9.3090909
## [346]  24.0454545  14.5636364  24.0090909  22.7636364  13.4636364
## [351]  13.6090909  26.2545455  16.2454545   9.4363636  15.9090909
## [356]  12.3454545   5.7000000  18.7454545  33.4545455   9.0818182
## [361]  10.9636364  11.7636364  24.8909091   6.8545455  29.4818182
## [366]  11.7363636  36.0909091  14.8454545  32.5181818  15.0363636
## [371]   5.2818182  17.7909091   2.7636364  17.9818182  11.1818182
## [376]  13.5000000  23.5727273  24.4454545  20.6727273  14.4545455
## [381]  27.1636364  13.5000000  15.0000000  19.2545455  22.6090909
## [386]  14.5818182  16.7181818   3.0636364   2.6363636  21.2909091
## [391]  14.1545455  25.1636364  13.4545455  19.2363636  20.0636364
## [396]  32.7636364  12.0545455   7.9818182   7.7363636   7.2636364
## [401]  24.5818182  23.8181818  25.6818182   6.9727273  15.0727273
## [406]  16.2454545   8.4363636   8.9090909  18.5818182  20.2727273
## [411]  18.4000000  17.1454545  27.3909091  27.9000000  16.6363636
## [416]  10.9363636  15.4454545  28.1545455  17.3181818   1.0636364
## [421]  18.5272727  21.8272727  19.5727273  12.2545455  21.2454545
## [426]  24.3181818   6.5000000  24.7272727  30.9181818  17.0272727
## [431]  10.9000000   0.2181818  22.1454545  29.9090909  15.7545455
## [436]  13.5272727  17.6090909  26.6545455  -3.6363636  12.8272727
## [441]  19.8727273  16.4454545  21.0545455   7.1272727  19.0545455
## [446]   7.5090909  14.5272727  19.1363636  22.5545455  10.5272727
## [451]  22.7818182  10.2818182  19.2000000  17.7181818  26.2818182
## [456]  15.0727273  16.5454545  28.9363636  15.0636364  20.4727273
## [461]  20.6272727  13.8636364  17.9545455  10.8727273   6.5818182
## [466]  25.0636364  23.2272727  15.6818182  21.5090909  15.8909091
```

```
## [471]  27.6181818  16.2909091  -1.1545455  18.7363636  22.1636364
## [476]  26.3727273  -1.0636364  12.5454545  13.1000000  25.0636364
## [481]  32.4181818  17.0454545  32.7181818  16.7090909  14.7000000
## [486]   6.2818182   6.2545455  11.2454545  24.7090909  26.8909091
## [491]  14.2181818  15.9909091  16.3454545  15.3454545   9.7272727
## [496]   1.9545455   4.4000000  16.8454545  11.2909091  13.4181818
```

```
# 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%
##  3.048636 30.590000
```

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

```
##     2.5%    97.5%
##  0.42500 32.74205
```

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

```
##      0.5%     99.5%
## -4.453545 38.707273
```

5.8 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
```

5.9 Use R for mediation

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