SEM and R

Bill

2021-05-16

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 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).

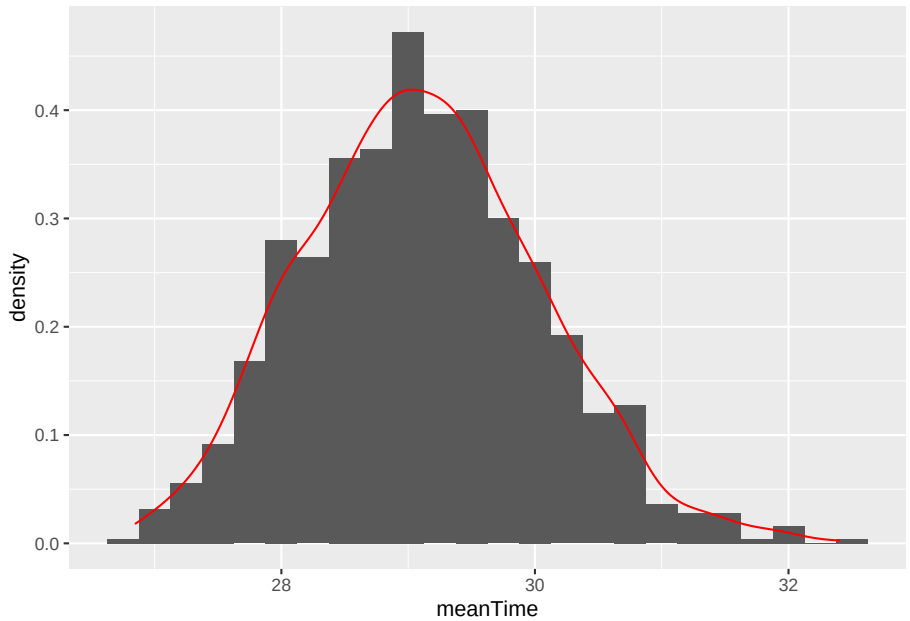
```
B = 1000
n = nrow(CommuteAtlanta)
boot.samples = matrix(sample(CommuteAtlanta$Time, size = B * n, replace = TRUE),
                      B, n)
```

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] 0.9306805
```

```
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.

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

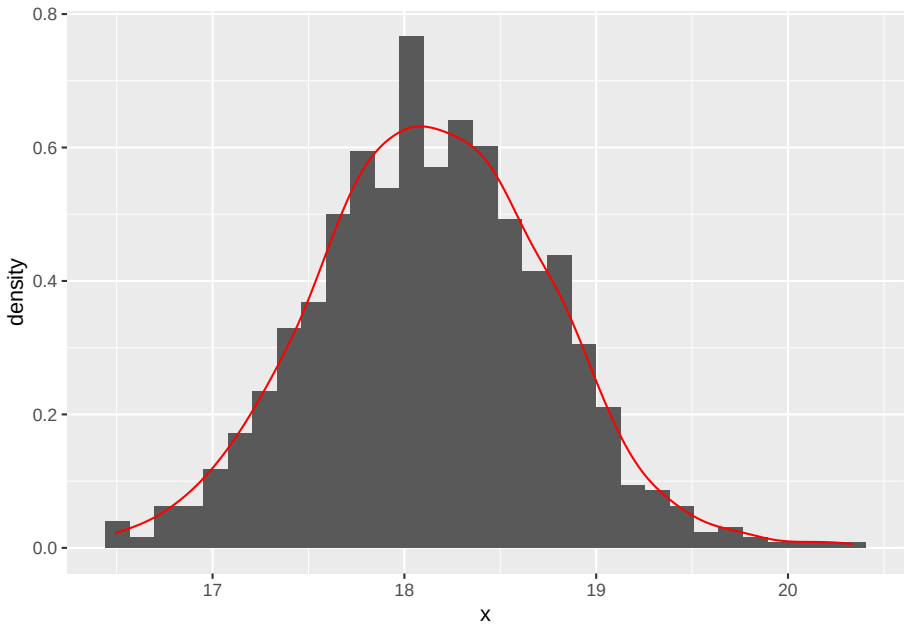
Note that, in the following, the author used 2 times SE to calculate the CI. The relationship between SD and SE:

"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] 16.94481 19.36719
```

5.4 Sample function

To understand the function of sample in R.

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

```
##  [1]  7  5 18  5 15  1  4 20  3 12  3 11 10 17  8 10 20 12  8 15
```

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)
```

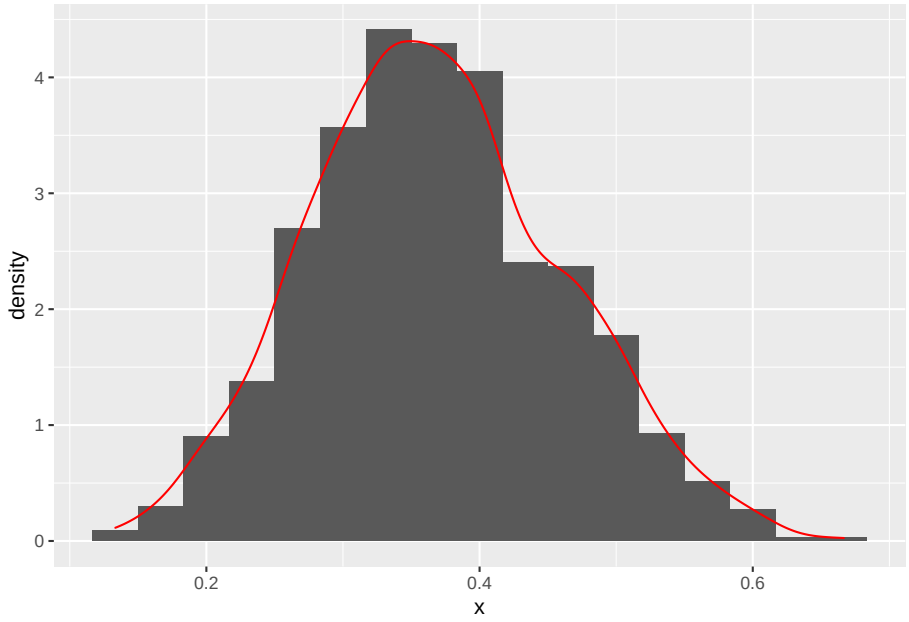
```
{
boot.sample = sample(n, replace = TRUE)
result[i] = mean(CommuteAtlanta$Distance[boot.sample])
}

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

```
## [1] 16.92112 19.39088
```

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)
```

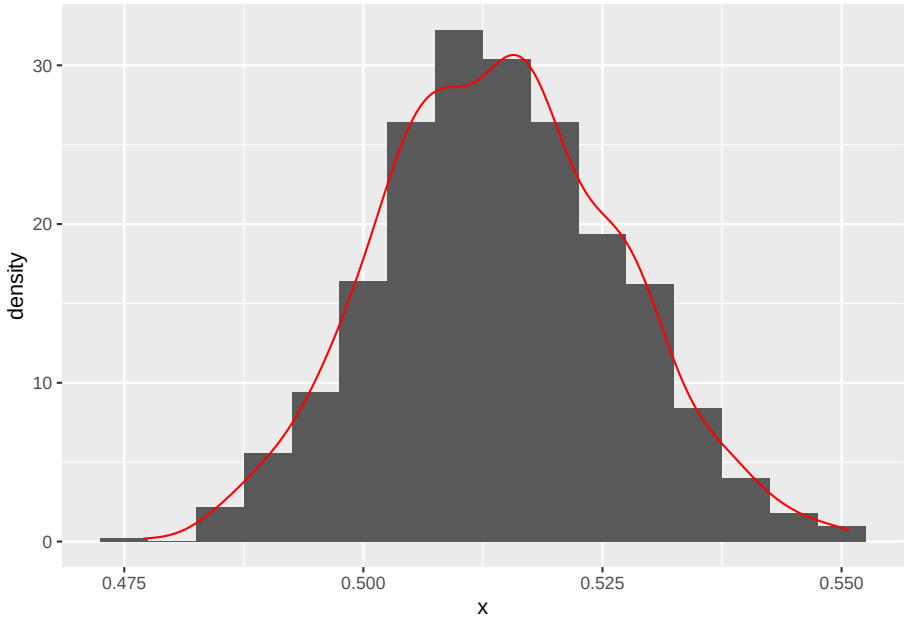



```
## [1] 0.1839065 0.5494269
```

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] 0.4895854 0.5395812
```

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%   (27.26, 30.94 )   (27.17, 30.90 )
##
## Level     Percentile            BCa
## 95%   (27.32, 31.05 )   (27.40, 31.13 )
## 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] 16.8727273 22.1909091 11.8454545 24.6727273  9.4181818 18.9181818
##   [7] 37.2454545 25.1545455 16.7727273 13.1454545 24.9545455 12.7818182
```

```
##  [13] 23.8000000 14.3454545 11.6818182 10.1090909 23.8545455 23.4000000
##  [19] 31.8636364  3.3727273 12.6272727 28.6636364 18.8000000 29.6090909
##  [25] 24.0181818 13.0363636 22.9545455 24.0727273 32.2636364 20.5000000
##  [31] 17.9090909 19.2727273 27.0727273 20.0909091 19.6000000 12.9818182
##  [37] 11.3818182  8.4272727 18.4454545 13.3636364 24.8000000 25.3636364
##  [43] 36.1818182 13.3818182 12.7818182 25.6363636 24.6363636  6.2363636
##  [49] 17.4090909  7.8090909 18.5636364 39.5181818 29.9090909 12.0363636
##  [55] 12.8181818 30.0363636 17.3000000 10.7181818 22.6909091 21.4636364
##  [61]  7.3000000 -8.6636364 25.1818182 11.4363636 11.6181818 24.2272727
##  [67] 12.0636364 13.1909091 25.1727273 10.1818182 19.9818182 27.1636364
##  [73] 10.8727273 11.7636364  2.4636364 21.0272727 12.3545455 10.9818182
##  [79] 13.0818182 20.9818182  7.2000000 23.9909091 16.7454545 13.3272727
##  [85] 32.6000000 17.4727273 22.0545455 18.2272727 24.3181818  6.5818182
##  [91] 15.5818182  9.1181818 15.1090909 22.5363636  4.5272727 11.3454545
##  [97]  7.5272727 21.4818182  4.5545455 21.7000000  6.2181818 24.3363636
## [103] 17.1545455 16.6818182 24.5090909 28.0363636  9.2000000 24.3090909
## [109]  9.2181818 22.0090909 13.2090909 16.6545455 20.2636364 18.3000000
## [115]  9.6545455 19.7636364 19.7272727 19.3272727 -1.3545455 13.2818182
## [121] 16.5454545 19.9818182  7.4181818 24.1090909 13.8545455 41.4727273
## [127]  3.3545455 22.8181818 32.5000000 27.8454545 18.3545455 27.4545455
## [133] 19.4363636 11.5818182 25.5545455 11.1000000 19.6181818 12.1727273
## [139] 13.7454545 30.8272727 23.5545455 15.1545455 23.6090909 17.3090909
## [145] 18.3636364 25.1090909 17.2000000 12.3636364 16.5090909 12.1090909
## [151]  2.3545455 15.7363636 20.0636364 10.9363636 24.2272727 13.1636364
## [157] 13.1272727 34.4000000 31.9727273 23.9272727  4.4545455 10.8454545
## [163] 10.1363636 15.0000000 29.5454545 16.0090909  9.1909091 22.2272727
## [169] 20.9363636 14.9727273 21.1000000 11.4545455 17.3818182  5.7363636
## [175] 23.1090909 16.1272727 12.8272727  1.1545455 -6.9818182 12.2181818
## [181] 27.8545455 14.1090909 16.0727273 17.1727273 28.3909091  2.0636364
## [187]  7.5636364 18.1363636  5.0636364 21.6545455 23.0636364 19.4000000
## [193] 18.1090909 25.2363636  8.9090909 15.8181818 29.7636364  9.7727273
## [199] 19.9909091 28.5181818 21.2818182  5.3818182 21.5363636 11.6272727
## [205] 18.5545455  8.6363636 16.4636364 24.2363636  1.7363636 18.4818182
## [211] 14.0818182 20.8000000 11.7818182 19.4545455 25.2545455 15.9727273
## [217] 20.5363636 35.0727273 12.3545455 21.2000000  3.9636364 19.0818182
## [223] 20.1090909  3.1272727 19.1454545 13.7636364 13.7090909 21.6818182
## [229] 12.4545455 12.5090909 15.0818182 20.6272727 26.0272727 28.0272727
## [235] 18.9636364 28.3454545 12.5818182 16.7181818 14.4272727  6.7636364
## [241] 15.7000000 23.9181818  9.6272727  5.2909091 29.3090909 24.9363636
## [247] 18.6090909 10.1090909 14.3454545 29.1727273 11.9909091 13.8636364
## [253] 11.2454545  2.4727273 13.0818182 22.2454545 24.6545455 22.5818182
## [259] 31.7454545 15.6818182 28.5545455 -1.4818182 28.0636364 19.0000000
## [265] 19.8545455 18.6909091 30.1181818 17.9545455  8.9636364 20.8727273
## [271] 18.0454545 26.5363636 17.4000000 28.1000000 32.4545455 22.6727273
## [277]  0.4181818 30.0272727 12.1090909 20.2181818 31.7545455 18.0636364
## [283] 19.2272727  6.2454545 21.8363636 14.1090909 34.6818182 11.6181818
```

```
## [289]  9.7727273  9.8727273 20.1545455 16.4909091 15.3727273 10.5272727
## [295] 17.8727273 12.7636364  2.2727273 33.2727273 19.9909091  9.1818182
## [301] 24.9545455  1.2545455 16.1181818 21.3545455  5.5272727 11.7909091
## [307] 26.9363636 20.2818182 17.8818182 33.4545455 10.5727273 18.0454545
## [313] 20.7272727 11.1727273 22.0272727 30.2090909 21.7636364 27.3545455
## [319] 13.8454545 14.4272727 19.1000000 21.2727273 13.3818182 11.7818182
## [325]  8.1636364  0.5454545  4.8727273 20.5363636  6.7818182 14.7090909
## [331] 14.2363636 19.2818182 20.2000000 28.3454545 20.8000000 13.6090909
## [337] 15.0727273 16.9181818 28.6454545 22.5909091 30.8454545 15.2272727
## [343] 29.2636364 11.4181818 21.1818182  9.8090909  4.3363636 21.5454545
## [349] 17.4454545 14.1636364 21.4818182 22.0818182  6.6090909 25.2090909
## [355] 29.5090909 31.4818182 15.1000000 14.7000000 26.3636364 13.5000000
## [361] 16.8636364 20.9909091 18.8000000 17.1000000 11.8090909 25.0363636
## [367] 24.7000000 16.5545455  9.9909091 16.5363636 14.7545455 15.7000000
## [373] 23.5272727 11.5454545 15.5090909 19.6000000 10.6454545 14.2181818
## [379] 18.6545455  8.2454545 13.0272727 15.3272727 17.2090909  8.2636364
## [385] 24.1090909 12.3090909 20.7909091 31.4363636 21.0272727 20.9545455
## [391] 19.7000000 25.1454545 14.8727273 26.9454545 23.6909091 12.7727273
## [397]  4.4636364 19.2545455 29.9272727 25.7363636  7.0818182 23.1272727
## [403] 16.7545455 21.8181818 10.9272727 22.0181818 14.4181818  4.3818182
## [409] 10.0181818 18.7363636 15.3090909  4.6000000  8.6727273 17.7090909
## [415] 25.4363636 28.0181818 24.6909091  8.6545455 12.6909091 18.7090909
## [421] -6.0000000 16.2545455 11.0818182  3.4909091 21.9636364  6.6818182
## [427] 13.6000000 23.3181818 28.5909091 21.7181818 17.7363636 23.8545455
## [433] 33.7272727 19.4272727 11.3000000 19.1000000 13.5363636  0.1727273
## [439] 21.5181818 37.7090909 10.4090909 28.5454545 31.4909091  9.6727273
## [445] 14.7818182 31.2000000  9.7545455 17.9181818 25.8272727 12.3181818
## [451] 18.5909091 22.3000000 18.4272727 10.4727273 14.3454545 16.7636364
## [457] 16.1818182  5.0000000 29.0454545 27.3636364 20.2909091  8.6090909
## [463]  7.5909091 15.1909091 17.8909091 23.9181818 12.3909091 11.8272727
## [469] 13.2727273 15.3181818 13.4545455 22.9454545 16.8000000 17.9727273
## [475] 17.0727273 10.7909091  6.5090909 25.5000000 26.5727273 16.3909091
## [481] 20.5272727  2.8272727  7.5000000 20.8909091 11.2818182 21.0090909
## [487]  4.6727273 19.4818182 17.6272727  9.5000000 12.8363636  8.0363636
## [493] 16.8909091 18.7636364 17.1272727 26.1545455 17.4909091 29.2090909
## [499] 25.4454545 23.0000000
```

```
# 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%
##  4.463182 30.240000
```

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

```
##      2.5%     97.5%
##  2.311591 32.478409
```

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

```
##     0.5%    99.5%
## -3.76350 37.47959
```

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
```

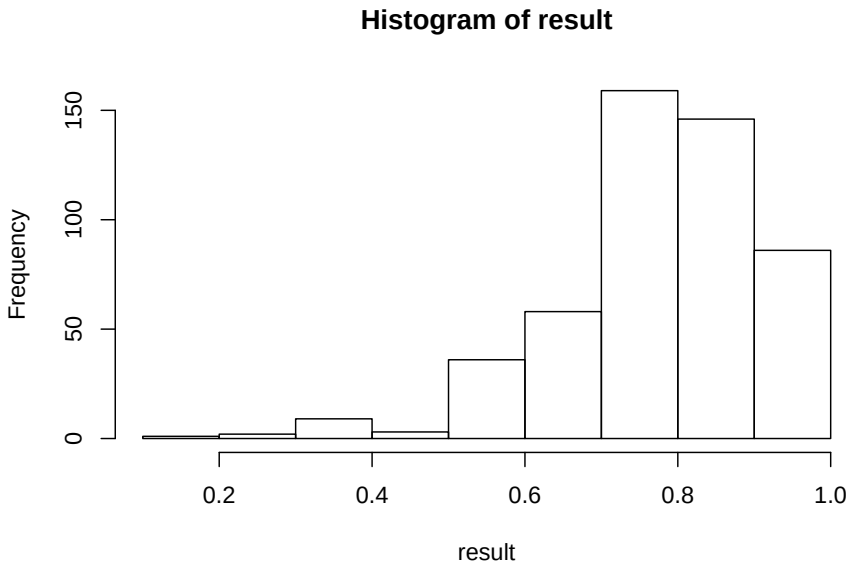
In the following, I will write my own code to execute the bootstrapping. I set the bootstrapping number only 500, for illustrative purposes. As we can see, the distribution is not symmetrical.

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

6.1 Basic idea

The following is based on the lecture note of https://bookdown.org/roback/bookdown-BeyondMLR/ch-poissonreg.html

There is also some R code related to this.

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

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

head(data_HH)
```

```
##   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 = data_HH)
summary(result_1)
```

```
##
## Call:
## glm(formula = total ~ age, family = poisson, data = data_HH)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.9079  -0.9637  -0.2155   0.6092   4.9561
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.5499422  0.0502754  30.829  < 2e-16 ***
## age         -0.0047059  0.0009363  -5.026 5.01e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 2362.5  on 1499  degrees of freedom
## Residual deviance: 2337.1  on 1498  degrees of freedom
## AIC: 6714
##
## Number of Fisher Scoring iterations: 5
```

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

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

That is, the difference in the household size mean by changing 1 unit of age (i.e., $\lambda_{Age+1} - \lambda_{Age}$) is a decrease of 5% of λ_{Age}.

We can then calculate the confidence interval.

$$(\hat{\beta}_1 - Z * SE(\hat{\beta}_1), \hat{\beta}_1 + Z * SE(\hat{\beta}_1))$$

$$(-0.0047 - 1.96 * 0.00094, -0.0047 + 1.96 * 0.00094) = (0.0065, 0.0029)$$

We can then plug them back to the exponential.

```
exp(-0.0065)
```

```
## [1] 0.9935211
```

```
exp(-0.0029)
```

```
## [1] 0.9971042
```

$$(e^{0.0065}, e^{0.0029}) = (0.9935, 0.9971)$$

You can also get the confidence interval directly use R code

```
confint(result_1)
```

```
## Waiting for profiling to be done...
```

```
##                    2.5 %       97.5 %
## (Intercept)  1.451170100  1.648249185
## age         -0.006543163 -0.002872717
```

```
exp(confint(result_1))
```

```
## Waiting for profiling to be done...
```

```
##                 2.5 %    97.5 %
## (Intercept) 4.2681057 5.1978713
## age         0.9934782 0.9971314
```

Note that, we use original beta to construct a confidence interval and then exponentiate the endpoints is due to the fact that the oringal one is more close to normal distribution.

6.2 Trying to understand

With $\hat{\beta_0} = 1.55$ and $\hat{\beta_1} = -0.005$, we can write down the following. I also simulated the data and showed the relationship between X and Y. As we can see the figure, the relationship is pretty linear. Thus, something to keep in mind, the poisson distribution we typically see is the histogram of Y, rather than the relationship between X and Y.

$$log(\hat{\lambda}) = 1.55 - 0.005Age$$

$$\hat{\lambda} = e^{1.55-0.005Age}$$

```
data_age<-seq(10,100,0.5)
f_age<-function(x){exp(1.55-(0.005*x))}
cbind(data_age,f_age(data_age))
```

```
##        data_age
##   [1,]     10.0 4.481689
##   [2,]     10.5 4.470499
##   [3,]     11.0 4.459337
##   [4,]     11.5 4.448202
##   [5,]     12.0 4.437096
##   [6,]     12.5 4.426017
##   [7,]     13.0 4.414965
##   [8,]     13.5 4.403942
##   [9,]     14.0 4.392946
##  [10,]     14.5 4.381977
##  [11,]     15.0 4.371036
##  [12,]     15.5 4.360122
##  [13,]     16.0 4.349235
##  [14,]     16.5 4.338376
##  [15,]     17.0 4.327543
##  [16,]     17.5 4.316738
##  [17,]     18.0 4.305960
##  [18,]     18.5 4.295208
##  [19,]     19.0 4.284483
##  [20,]     19.5 4.273786
##  [21,]     20.0 4.263115
##  [22,]     20.5 4.252470
##  [23,]     21.0 4.241852
##  [24,]     21.5 4.231261
##  [25,]     22.0 4.220696
##  [26,]     22.5 4.210157
##  [27,]     23.0 4.199645
##  [28,]     23.5 4.189159
##  [29,]     24.0 4.178699
```

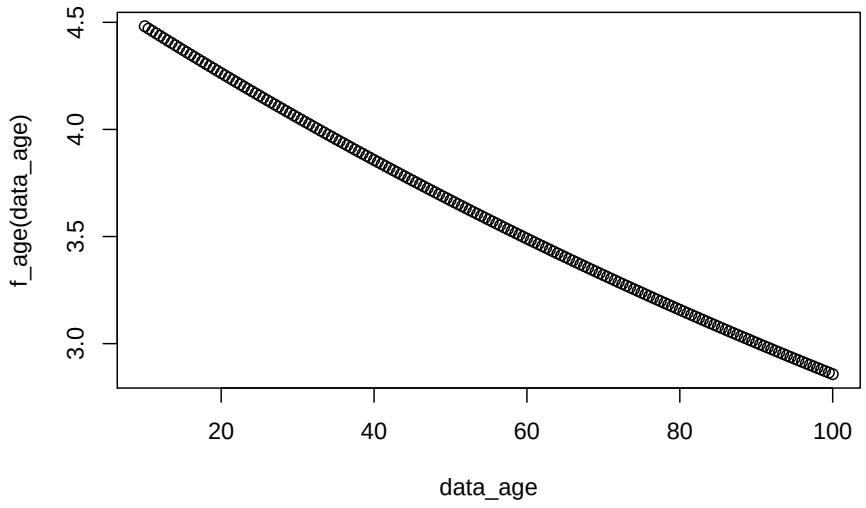
```
##  [30,]     24.5 4.168265
##  [31,]     25.0 4.157858
##  [32,]     25.5 4.147476
##  [33,]     26.0 4.137120
##  [34,]     26.5 4.126791
##  [35,]     27.0 4.116486
##  [36,]     27.5 4.106208
##  [37,]     28.0 4.095955
##  [38,]     28.5 4.085728
##  [39,]     29.0 4.075527
##  [40,]     29.5 4.065351
##  [41,]     30.0 4.055200
##  [42,]     30.5 4.045075
##  [43,]     31.0 4.034975
##  [44,]     31.5 4.024900
##  [45,]     32.0 4.014850
##  [46,]     32.5 4.004825
##  [47,]     33.0 3.994826
##  [48,]     33.5 3.984851
##  [49,]     34.0 3.974902
##  [50,]     34.5 3.964977
##  [51,]     35.0 3.955077
##  [52,]     35.5 3.945201
##  [53,]     36.0 3.935351
##  [54,]     36.5 3.925525
##  [55,]     37.0 3.915723
##  [56,]     37.5 3.905946
##  [57,]     38.0 3.896193
##  [58,]     38.5 3.886465
##  [59,]     39.0 3.876761
##  [60,]     39.5 3.867081
##  [61,]     40.0 3.857426
##  [62,]     40.5 3.847794
##  [63,]     41.0 3.838187
##  [64,]     41.5 3.828603
##  [65,]     42.0 3.819044
##  [66,]     42.5 3.809508
##  [67,]     43.0 3.799996
##  [68,]     43.5 3.790508
##  [69,]     44.0 3.781043
##  [70,]     44.5 3.771603
##  [71,]     45.0 3.762185
##  [72,]     45.5 3.752792
##  [73,]     46.0 3.743421
##  [74,]     46.5 3.734075
##  [75,]     47.0 3.724751
```

```
##  [76,]     47.5 3.715451
##  [77,]     48.0 3.706174
##  [78,]     48.5 3.696920
##  [79,]     49.0 3.687689
##  [80,]     49.5 3.678481
##  [81,]     50.0 3.669297
##  [82,]     50.5 3.660135
##  [83,]     51.0 3.650996
##  [84,]     51.5 3.641880
##  [85,]     52.0 3.632787
##  [86,]     52.5 3.623716
##  [87,]     53.0 3.614668
##  [88,]     53.5 3.605643
##  [89,]     54.0 3.596640
##  [90,]     54.5 3.587659
##  [91,]     55.0 3.578701
##  [92,]     55.5 3.569766
##  [93,]     56.0 3.560853
##  [94,]     56.5 3.551962
##  [95,]     57.0 3.543093
##  [96,]     57.5 3.534246
##  [97,]     58.0 3.525421
##  [98,]     58.5 3.516619
##  [99,]     59.0 3.507838
## [100,]     59.5 3.499080
## [101,]     60.0 3.490343
## [102,]     60.5 3.481628
## [103,]     61.0 3.472935
## [104,]     61.5 3.464263
## [105,]     62.0 3.455613
## [106,]     62.5 3.446985
## [107,]     63.0 3.438379
## [108,]     63.5 3.429793
## [109,]     64.0 3.421230
## [110,]     64.5 3.412687
## [111,]     65.0 3.404166
## [112,]     65.5 3.395666
## [113,]     66.0 3.387188
## [114,]     66.5 3.378730
## [115,]     67.0 3.370294
## [116,]     67.5 3.361879
## [117,]     68.0 3.353485
## [118,]     68.5 3.345111
## [119,]     69.0 3.336759
## [120,]     69.5 3.328428
## [121,]     70.0 3.320117
```

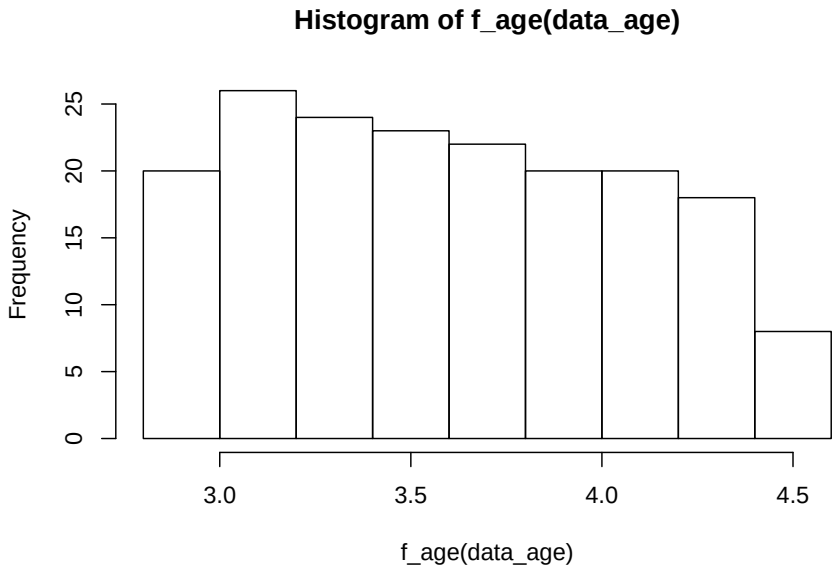
```
## [122,]     70.5 3.311827
## [123,]     71.0 3.303558
## [124,]     71.5 3.295309
## [125,]     72.0 3.287081
## [126,]     72.5 3.278874
## [127,]     73.0 3.270687
## [128,]     73.5 3.262520
## [129,]     74.0 3.254374
## [130,]     74.5 3.246248
## [131,]     75.0 3.238143
## [132,]     75.5 3.230058
## [133,]     76.0 3.221993
## [134,]     76.5 3.213948
## [135,]     77.0 3.205923
## [136,]     77.5 3.197918
## [137,]     78.0 3.189933
## [138,]     78.5 3.181968
## [139,]     79.0 3.174023
## [140,]     79.5 3.166098
## [141,]     80.0 3.158193
## [142,]     80.5 3.150307
## [143,]     81.0 3.142441
## [144,]     81.5 3.134595
## [145,]     82.0 3.126768
## [146,]     82.5 3.118961
## [147,]     83.0 3.111174
## [148,]     83.5 3.103405
## [149,]     84.0 3.095657
## [150,]     84.5 3.087927
## [151,]     85.0 3.080217
## [152,]     85.5 3.072526
## [153,]     86.0 3.064854
## [154,]     86.5 3.057202
## [155,]     87.0 3.049568
## [156,]     87.5 3.041954
## [157,]     88.0 3.034358
## [158,]     88.5 3.026782
## [159,]     89.0 3.019224
## [160,]     89.5 3.011686
## [161,]     90.0 3.004166
## [162,]     90.5 2.996665
## [163,]     91.0 2.989183
## [164,]     91.5 2.981719
## [165,]     92.0 2.974274
## [166,]     92.5 2.966848
## [167,]     93.0 2.959440
```

```
## [168,]     93.5 2.952050
## [169,]     94.0 2.944680
## [170,]     94.5 2.937327
## [171,]     95.0 2.929993
## [172,]     95.5 2.922677
## [173,]     96.0 2.915379
## [174,]     96.5 2.908100
## [175,]     97.0 2.900839
## [176,]     97.5 2.893596
## [177,]     98.0 2.886371
## [178,]     98.5 2.879164
## [179,]     99.0 2.871975
## [180,]     99.5 2.864804
## [181,]    100.0 2.857651
```

```
plot(data_age,f_age(data_age))
```




```
hist(f_age(data_age))
```



6.3 Deviance

```
basic_model <- glm(total ~ 1, family = poisson, data = data_HH)
deviance_1 <- anova(basic_model, result_1, test = "Chisq")
deviance_1
```

```
## Analysis of Deviance Table
##
## Model 1: total ~ 1
## Model 2: total ~ age
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1      1499     2362.5
## 2      1498     2337.1  1   25.399 4.661e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

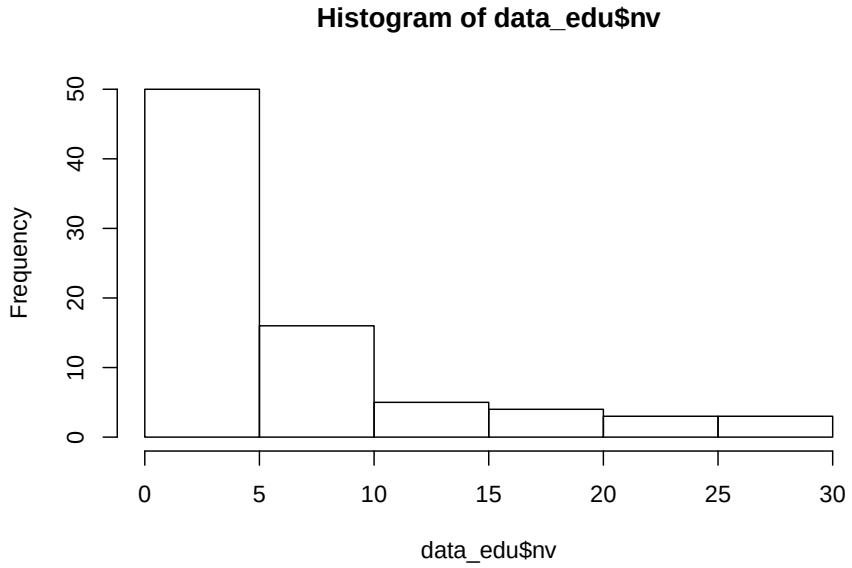
6.4 Overdispersion (using another example)

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

```
##   Enrollment type nv     nvrate enroll1000 region
## 1       5590    U 30 5.36672630      5.590     SE
```

```
## 2        540    C  0 0.00000000      0.540     SE
## 3      35747    U 23 0.64341064     35.747      W
## 4      28176    C  1 0.03549120     28.176      W
## 5      10568    U  1 0.09462528     10.568     SW
## 6       3127    U  0 0.00000000      3.127     SW
```

```
hist(data_edu$nv)
```




```
results_3<- glm(nv ~ type + region, family = poisson,
                offset = log(enroll1000), data = data_edu)
summary(results_3)
```

```
##
## Call:
## glm(formula = nv ~ type + region, family = poisson, data = data_edu,
##     offset = log(enroll1000))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.5697  -1.9079  -0.7233   0.8738   8.4564
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.60161    0.17120  -9.355  < 2e-16 ***
## typeU        0.34011    0.13234   2.570  0.01017 *
## regionMW     0.09942    0.17752   0.560  0.57547
```

```
## regionNE     0.78109    0.15305   5.103 3.33e-07 ***
## regionSE     0.87668    0.15314   5.725 1.04e-08 ***
## regionSW     0.50251    0.18508   2.715  0.00663 **
## regionW      0.27324    0.18741   1.458  0.14484
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 491.00  on 80  degrees of freedom
## Residual deviance: 426.01  on 74  degrees of freedom
## AIC: 657.89
##
## Number of Fisher Scoring iterations: 6
```

```
results_4 <- glm(nv ~ type + region, family = quasipoisson,
                 offset = log(enroll1000), data = data_edu)
summary(results_4)
```

```
##
## Call:
## glm(formula = nv ~ type + region, family = quasipoisson, data = data_edu,
##     offset = log(enroll1000))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.5697  -1.9079  -0.7233   0.8738   8.4564
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.60161    0.48312  -3.315  0.00142 **
## typeU        0.34011    0.37347   0.911  0.36542
## regionMW     0.09942    0.50097   0.198  0.84324
## regionNE     0.78109    0.43191   1.808  0.07460 .
## regionSE     0.87668    0.43216   2.029  0.04609 *
## regionSW     0.50251    0.52230   0.962  0.33913
## regionW      0.27324    0.52887   0.517  0.60694
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasipoisson family taken to be 7.963687)
##
##     Null deviance: 491.00  on 80  degrees of freedom
## Residual deviance: 426.01  on 74  degrees of freedom
## AIC: NA
##
```

```
## Number of Fisher Scoring iterations: 6
```

```
difference_dev <- anova(results_4, results_3, test = "F")
difference_dev
```

```
## Analysis of Deviance Table
##
## Model 1: nv ~ type + region
## Model 2: nv ~ type + region
##   Resid. Df Resid. Dev Df Deviance F Pr(>F)
## 1        74     426.01
## 2        74     426.01  0        0
```

Chapter 7

Use R for mediation

References:

https://bookdown.org/roback/bookdown-BeyondMLR/ch-poissonreg.html

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

7.1 Normal Distribution Case

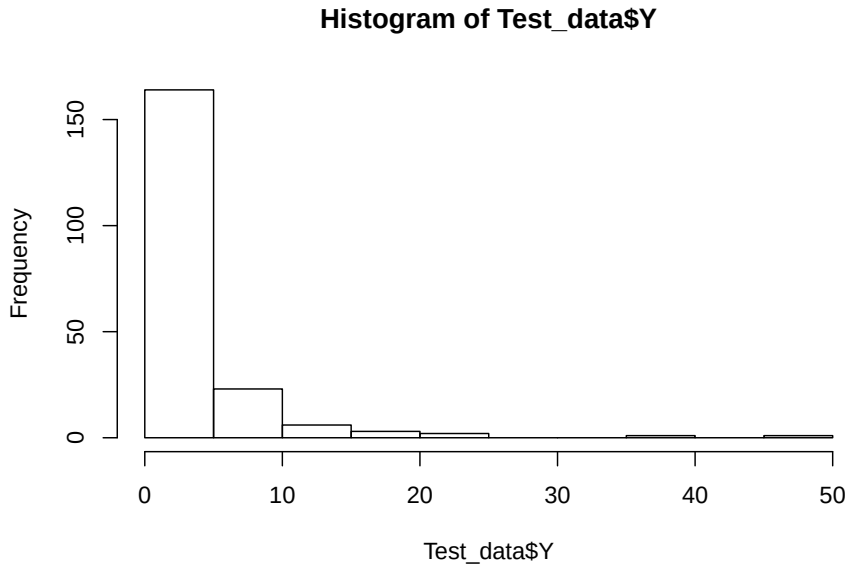
The following code generates the data to be used in the mediation model. Based on the histogram, we can see that it follows Poisson distribution.

```
# Generate data for mediation analysis
# https://ademos.people.uic.edu/Chapter14.html

# Generate Possion data
# https://stats.stackexchange.com/questions/27443/generate-data-samples-from-poisson-regression

set.seed(123)
N <- 200
X <- rnorm(N, 1, 1)
M <- 0.6*X + rnorm(N, 0, 1)
mu <- exp(0.2*X+0.8*M )
Y <- rpois(n=N, lambda=mu)

Test_data <- data.frame(X, M, Y)

head(Test_data)
```

```
##           X         M Y
## 1 0.4395244 2.4625250 7
## 2 0.7698225 1.7743065 4
```

```
## 3 2.5587083 1.2700799 2
## 4 1.0705084 1.1854991 1
## 5 1.1292877 0.2632327 1
## 6 2.7150650 1.1527921 9
```

```
hist(Test_data$Y)
```

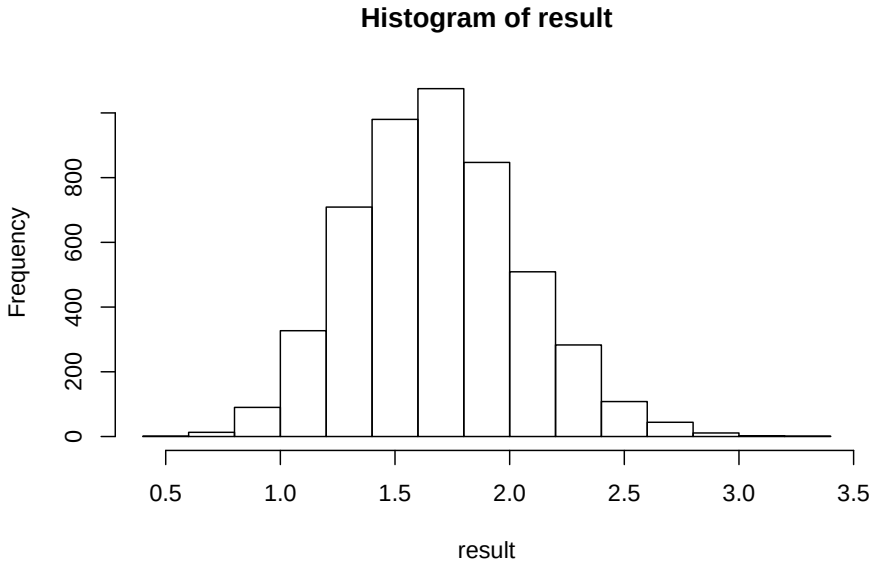



```
# write.csv(Test_data,"Test_data.csv")
```

Next, while in reality it follows Poisson distribution, the following assumes normal distribution. You will find the results are consistent with PROCESS.

```
Normal_Mediation<-function(X, M, Y, data_used, resampling_size=5000)

  {

  result = rep(NA, resampling_size)
  n_row = nrow(data_used)

  for (i in 1:resampling_size)
  {
  boot.sample = sample(n_row, replace = TRUE)
  data_temp<-data_used[boot.sample,]

  # a path
  result_a_temp<-lm(M~X, data = data_temp)$coefficients
```

```
  names(result_a_temp) <- NULL
  a_0_temp<-result_a_temp[1]
  a_1_temp<-result_a_temp[2]

  # b path
  result_b_temp<-lm(Y~M+X, data = data_temp)$coefficients
  names(result_b_temp) <- NULL
  b_0_temp<-result_b_temp[1]
  b_1_temp<-result_b_temp[2]
  c_1_apostrophe_temp<-result_b_temp[3]

  #calculating the indirect effect
  indirect_temp<-a_1_temp*b_1_temp
  result[i]=indirect_temp
  }

  hist(result)
  sd(result)

  print(mean(result) + c(-1, 1) * 2 * sd(result))
   print(quantile(result,c(0.025,0.975)))

}

Normal_Mediation(X=X, M=M,Y=Y, data_used = Test_data,resampling_size=5000)
```




```
## [1] 0.9502475 2.4221161
##     2.5%    97.5%
## 1.021319 2.453762
```

7.2 Poisson Distribution Case

However, the problem is that the DV is count data. So, it is better to take that into consideration. The following is based on the paper of Geldhof 2017, Accommodating binary and count variables in mediation, A case for conditional indirect effects.

In particular,

Poisson regression uses the log link. For the b path function, it is as follows.

$$log(Y) = e^{b_0 + b_1 M + c^{'} X}$$

Thus, its first partial derivative again M is as follows.

$$b_1 e^{b_0 + b_1 M + c^{'} X}$$

Where,

$$M = a_0 + a_1 X$$

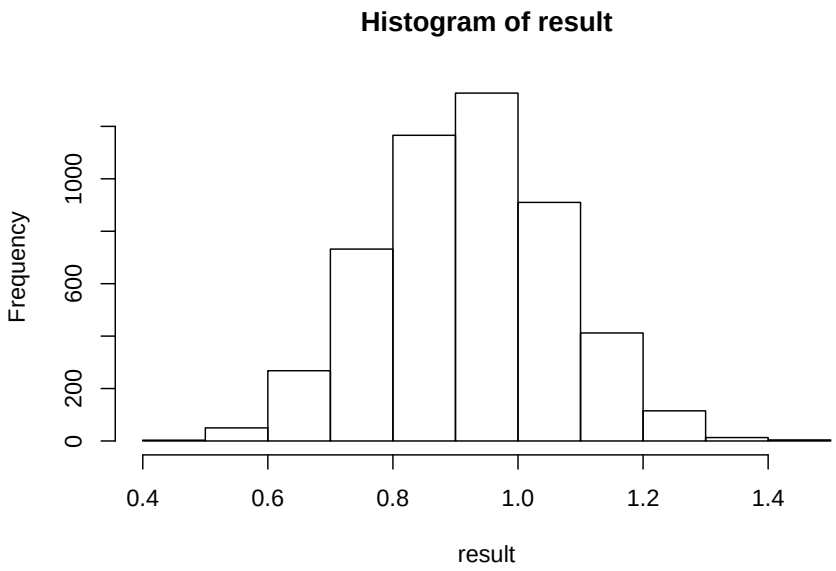
Thus, the indirect effect is as follows.

$$IndirectEffect = a_1 b_1 e^{b_0 + b_1 M + c^{'} X} = a_1 b_1 e^{b_0 + b_1 (a_0 + a_1 X) + c^{'} X}$$

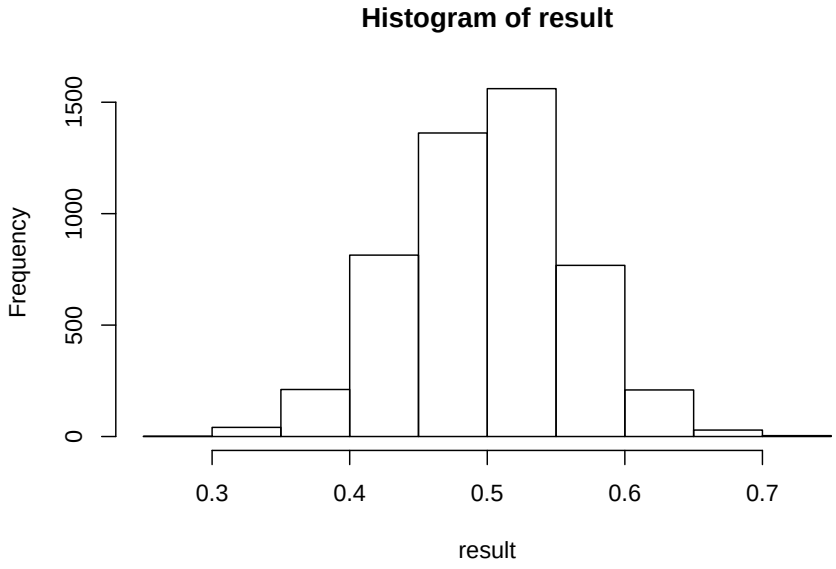
As we can see the indirect effect is not a constant, as it depends on X. Different X values will lead to different indirect effects. Thus, you can see the following R code takes this into consideration.

```
# x_predetermined = 0 : X = Mean
# x_predetermined = 1 : X = Mean + SD
# x_predetermined = -1 : X = Mean - SD

Poisson_Mediation<-function(X, M, Y, data_used, x_predetermined=0, resampling_size=5000)
{

    result = rep(NA, resampling_size)
    n_row = nrow(data_used)


    if(x_predetermined==0){x_predetermined=mean(data_used$X)}
    else if (x_predetermined==-1){x_predetermined=mean(data_used$X)-sd(data_used$X)}
    else(x_predetermined=mean(data_used$X)+sd(data_used$X))

for (i in 1:resampling_size)
{
  boot.sample = sample(n_row, replace = TRUE)
  data_temp<-data_used[boot.sample,]

  # a path
  result_a_temp<-lm(M~X, data = data_temp)$coefficients
  names(result_a_temp) <- NULL
  a_0_temp<-result_a_temp[1]
  a_1_temp<-result_a_temp[2]

  # b path
  result_b_temp<-glm(Y~M+X, data = data_temp, family = quasipoisson)$coefficients
  names(result_b_temp) <- NULL
  b_0_temp<-result_b_temp[1]
  b_1_temp<-result_b_temp[2]
  c_1_apostrophe_temp<-result_b_temp[3]

  #calculating the indirect effect

  M_estimated_temp=a_0_temp+a_1_temp*x_predetermined
  indirect_temp<-a_1_temp*b_1_temp*exp(b_0_temp+b_1_temp*M_estimated_temp+c_1_apostrophe_temp*x_p
  result[i]=indirect_temp
```

```
}
hist(result)
quantile(result,c(0.025,0.975))
}

# X = Mean
Poisson_Mediation(X=X, M=M,Y=Y, data_used = Test_data,x_predetermined=0,resampling_size
```

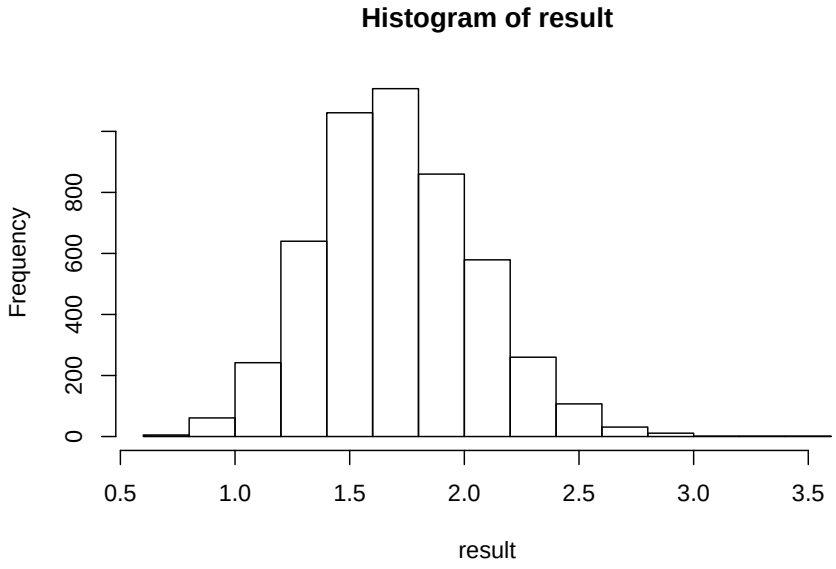



```
##      2.5%     97.5%
## 0.6432712 1.2018952
# X = Mean - 1 SD
Poisson_Mediation(X=X, M=M,Y=Y, data_used = Test_data,x_predetermined=-1,resampling_siz
```




```
##      2.5%     97.5%
## 0.3773349 0.6186202
```

```
# X = Mean + 1 SD
Poisson_Mediation(X=X, M=M,Y=Y, data_used = Test_data,x_predetermined=1,resampling_size=5000)
```



```
##     2.5%    97.5%
## 1.073396 2.435226
```