SEM and R

Bill

2021-05-02

Contents

1 SEM and R **5**

2 Introduction **7**
- 2.1 Definitions (Basic Concepts) 7
- 2.2 The path diagram 8
- 2.3 Lavaan syntax 8
- 2.4 Regression and path analysis 9

3 Real data example (Simple linear regression) **11**
- 3.1 Read the data into the R Studio environment. 11

4 Real data example (Multiple linear regression) **15**

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

6 Poisson Regression **31**
- 6.1 Basic idea 31
- 6.2 Trying to understand 34

7 Use R for mediation **37**

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.

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

```r
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
```

```r
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).

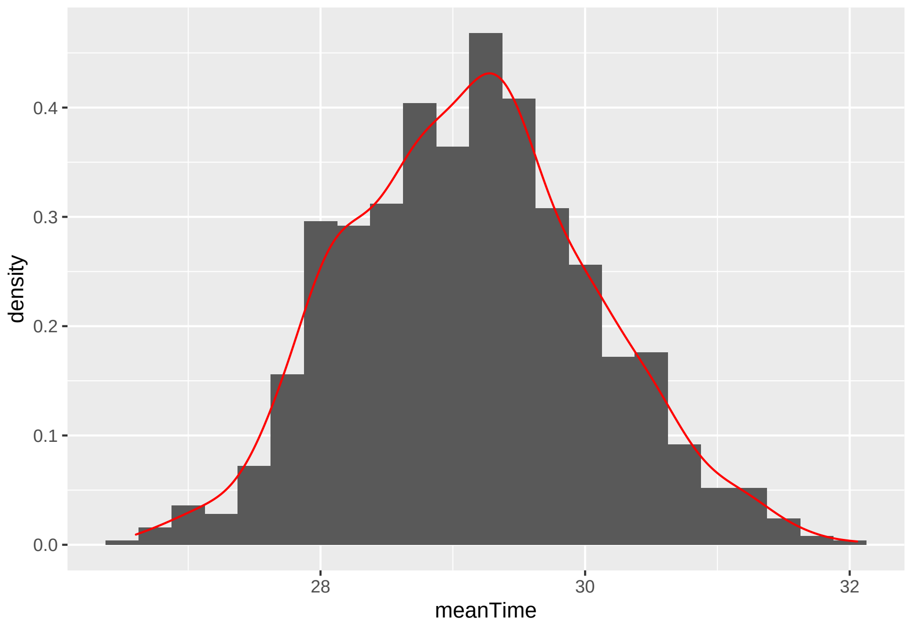
```r
B = 1000
n = nrow(CommuteAtlanta)
boot.samples = matrix(sample(CommuteAtlanta$Time, size = B * n, replace = TRUE),
                      B, n)
```

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.

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

We can then plot all the means.

```r
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] 0.9243346
```

```
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.

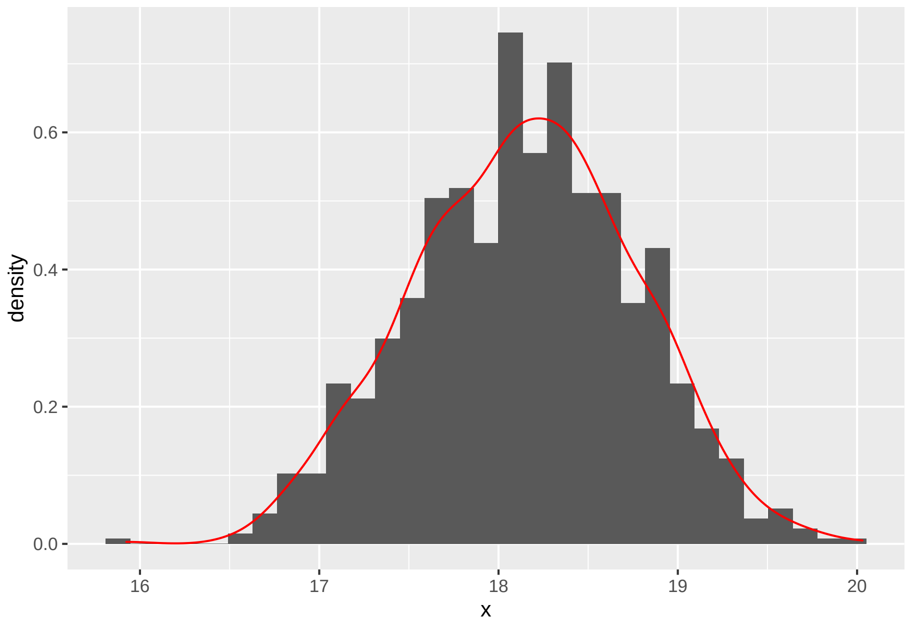
$$\overline{X} \pm Z\frac{S}{\sqrt{n}} = 29.11 \pm 1.96\frac{20.72}{\sqrt{500}} = 27.29, 30.93$$

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] 16.92365 19.38835
```

5.4 Sample function

To understand the function of sample in R.

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

```
##  [1] 16  4 11 19  9  7 15  7  2  2 10 10  6 17 13 17 15 18 17 11
```

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)
```

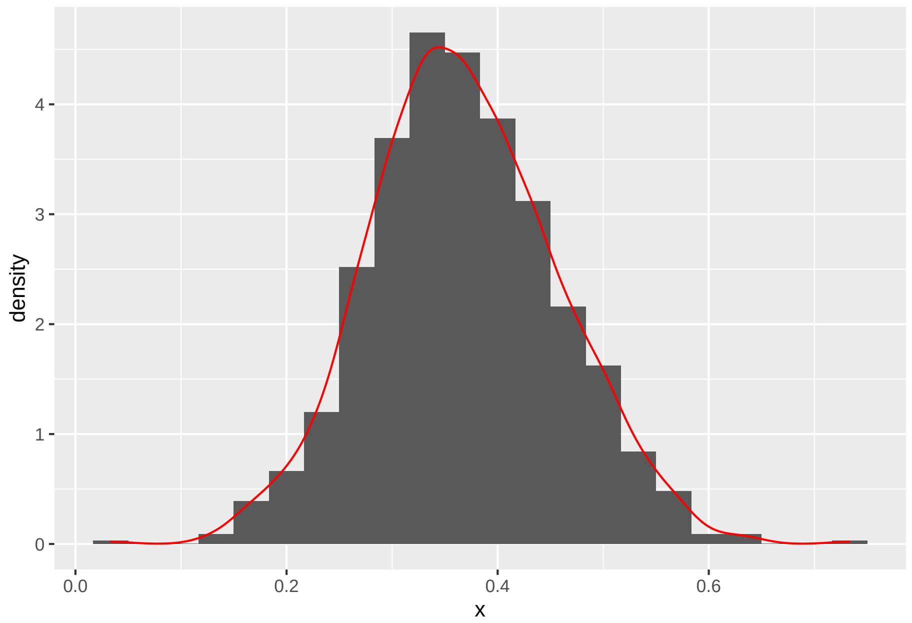
```
{
boot.sample = sample(n, replace = TRUE)
result[i] = mean(CommuteAtlanta$Distance[boot.sample])
}

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

```
## [1] 16.92337 19.38863
```

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.

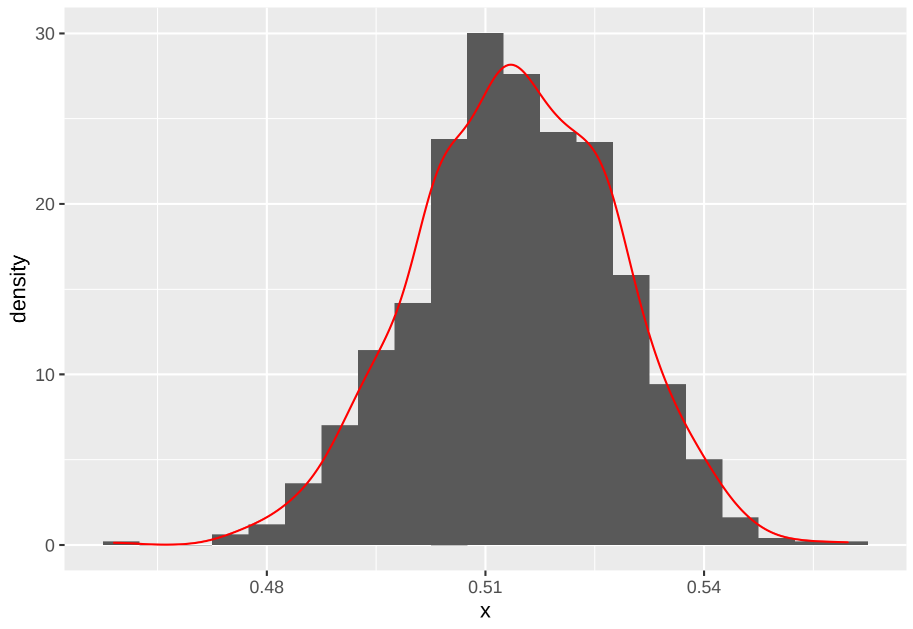
```
reeses = c(rep(1, 11), rep(0, 19))
reeses.boot = boot.mean(reeses, 1000, binwidth = 1/30)
```



```
## [1] 0.1889370 0.5443964
```

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] 0.4873028 0.5418639
```

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, 30.93 )   (27.23, 30.84 )
##
## Level     Percentile            BCa
## 95%   (27.38, 30.99 )   (27.44, 31.10 )
## Calculations and Intervals on Original Scale
```

5.7 Concept of Percentile

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



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

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

```
##   [1] 22.0545455 26.2818182 25.9000000 32.1636364 20.1636364  7.4454545
##   [7] 23.4181818 19.9727273  6.0454545 14.6090909  5.0454545 37.4545455
##  [13]  5.6818182  5.1636364 22.0181818 11.0727273 15.6727273  9.8818182
##  [19] 14.7272727  2.9272727 17.5545455  8.7636364 16.5727273 15.7909091
##  [25] 14.0818182 22.8727273 20.3545455 17.3363636 20.5181818 11.8545455
##  [31] 16.9727273  5.4818182 26.0090909 21.7454545 23.4545455  5.7454545
##  [37] 11.0818182  3.8636364 24.0909091 32.5727273  3.2545455  2.1636364
##  [43] 20.7454545 22.0272727 19.1272727 14.7727273  3.7636364 10.1818182
##  [49] 20.1545455 12.1090909 -0.5727273 11.1363636 12.0636364 21.9818182
##  [55] 11.3818182 11.3818182  5.8272727 27.3727273 15.8363636  8.7727273
##  [61] 10.4181818 16.9636364 30.8909091 27.6090909 21.9272727 13.5090909
##  [67] 13.4363636  3.3818182 26.5272727 11.2181818 21.1000000 33.5454545
##  [73] 21.2727273 20.0363636 25.0090909 13.4727273 12.7636364 18.3090909
##  [79] 16.8636364 10.8454545 17.4909091  3.4272727 11.9636364 13.8272727
##  [85]  9.8454545  4.3545455 26.8363636  9.9909091 32.6000000  6.4272727
##  [91] 12.7181818 24.5181818 14.1272727 25.9818182 26.9727273 14.8181818
##  [97] 20.5909091 12.5545455 24.5545455 23.3727273  9.6090909 18.5272727
## [103] 21.3000000 11.5090909 24.3363636 21.8636364 25.1545455 20.3727273
## [109] 14.0818182  8.5000000 11.8272727 25.0272727  5.6818182 15.4272727
## [115]  8.6909091 21.2181818 22.9636364 21.5181818 12.8454545 17.8090909
## [121] 17.9090909  6.0363636 17.8363636 10.7545455 18.0545455 18.9636364
## [127] 13.4272727  8.3272727 13.9090909  7.0363636  6.0000000 19.9272727
## [133] 24.2909091 21.6727273 18.6363636 22.3909091 24.9000000  9.4272727
## [139] 25.2272727  9.8272727 15.8181818 19.3181818 27.9000000  8.8181818
## [145]  9.6090909 30.4000000 13.9818182 10.6727273 27.0363636 10.7545455
## [151] 21.8000000 20.7818182 10.4363636 23.3000000 24.6090909 15.8636364
## [157] 17.8545455 12.1181818 17.7545455  7.3727273  4.7818182 22.7363636
## [163] 16.1181818 13.6727273 12.5090909 17.4545455 22.0090909 13.5090909
## [169] 11.6181818 10.4000000 -2.4272727 18.4545455  6.8272727 14.4272727
## [175] 20.5454545  7.0090909 17.4363636 13.6818182 16.2545455 20.5636364
## [181] -0.1818182  8.9272727 21.3272727 12.1818182  6.4272727 21.3818182
## [187] 11.9000000 20.7545455 14.3909091  1.6454545 22.6272727 17.1454545
## [193] 19.0090909 -3.9363636 22.3545455 14.7363636  4.9545455 19.5909091
## [199] 15.0727273  0.3090909 13.6727273 11.4727273 17.1727273 16.2000000
## [205] 18.7000000 15.1272727 32.6727273 20.3727273 21.8909091 24.8727273
## [211] 18.0000000 12.5454545 14.4181818 14.2545455 25.6727273  1.7181818
## [217] 18.5818182 16.9727273  6.7363636 33.9181818 27.7636364 14.9727273
## [223] 12.0181818 20.5272727 26.7090909  9.7909091 17.1909091 12.8090909
## [229] 18.4272727 12.8454545 27.3181818 26.3000000 16.7636364 13.3090909
## [235] 15.0181818  7.8272727 14.8363636 18.9181818 17.6636364 10.5363636
## [241]  4.6727273 16.8000000 28.5090909 25.2090909 27.1818182 25.2181818
## [247]  5.3727273 20.1363636 18.3181818 13.5181818 28.4909091 10.6090909
## [253] 12.2272727 15.1818182 25.9272727  9.3909091 15.2727273  8.1000000
## [259]  9.1272727 14.3727273 -1.1909091 25.1727273  5.5272727 26.3545455
## [265]  7.0000000 19.8272727 37.1636364 14.6181818 12.9000000 19.7181818
## [271] 23.1363636 17.3636364 27.4454545 34.1181818 39.5636364 24.9000000
## [277] 25.2363636 32.8363636 10.9363636 20.4818182 17.9272727 23.3636364
## [283] 18.3545455 12.4363636 10.0000000  5.0636364 19.6181818 28.7727273
## [289]  9.6727273 26.9909091  3.7636364  8.6090909 13.6909091  3.4272727
## [295] 20.9000000 22.4000000 13.3363636 27.0545455 17.4090909 13.0727273
## [301] 19.4272727 14.5909091 16.5909091 19.7818182 18.2636364  6.6818182
## [307] 22.0363636 17.7272727 18.5454545 20.4545455 26.5181818 17.1909091
## [313] 11.1000000 -1.3818182 11.8181818  4.1636364 26.2818182 24.5363636
## [319] 12.2090909  7.5909091 28.6727273 19.7090909 20.6181818  5.0727273
## [325] 18.3000000 19.3454545  7.9272727 18.5636364 10.7909091 11.4272727
## [331] 20.6181818  9.4454545 31.2727273 23.4454545 23.7000000 18.5272727
## [337] 12.1818182 11.4272727 11.3818182 18.4000000 11.6000000 17.9636364
## [343] 17.2454545 23.5545455 17.7454545 12.6181818 13.3818182 12.9454545
## [349] 21.7363636 21.9000000 17.1272727  5.1545455 17.0181818 20.5545455
## [355] 29.9454545 18.4454545 17.1000000 19.5454545 17.4818182 21.8909091
## [361]  2.1454545 25.0454545  7.4454545 25.0909091 14.9454545 31.8000000
## [367] 25.5272727 15.6909091 10.4909091 25.5909091 11.3909091 12.6272727
## [373]  7.3636364 26.1000000 13.8000000 32.3090909 15.3090909  3.5727273
## [379] 22.9545455  3.5636364 32.2454545 14.1636364 16.4272727 29.7454545
## [385] 13.1272727  9.5545455 -2.3636364 27.7727273 12.3000000 11.9727273
## [391] 17.8090909 13.9727273 17.8090909 22.7363636 21.8000000 10.6545455
## [397] 19.5636364 20.8727273 25.2818182 14.0545455 20.1545455 23.0454545
## [403] 11.9454545  6.5454545 10.8909091 13.6454545 19.3636364 26.0454545
## [409] 25.2000000 17.5272727 12.3272727 25.8181818 26.3909091  2.8636364
## [415] 16.7636364 18.7727273 27.1454545 20.3181818 13.8727273 19.8090909
## [421] 14.9363636 21.1000000  9.3090909 15.1818182 23.1545455 24.7454545
## [427] 15.1363636 11.6727273 17.0727273  2.4818182 10.5636364 10.4727273
## [433] 15.2818182 24.0818182  9.3818182 23.7272727 31.6000000  6.4000000
## [439] 18.1636364  0.2545455 20.3454545 19.8727273 -1.1727273 11.9000000
## [445] 32.9090909 15.7000000 16.3363636 25.7454545 21.2090909  9.3090909
## [451] 13.9545455 23.8454545 10.0363636 32.3727273 20.6727273 25.2272727
## [457] 27.2545455 17.1545455 19.5454545 -0.1000000  4.2545455 13.7636364
## [463] 34.7272727  6.7818182 19.0090909 10.0636364  8.7909091 16.2727273
## [469] 24.2090909 12.2363636 36.9090909  7.0818182 31.4545455 25.5000000
## [475] 11.7454545 24.0909091 11.0272727 11.4636364 12.0818182  4.8818182
## [481]  6.7181818 16.3363636 14.2090909 16.0818182 24.6545455 12.5090909
## [487] 16.3090909 19.5818182  0.1909091  3.3454545 18.0363636 29.9000000
## [493] 25.6727273 18.7090909 15.0090909 11.2272727 15.8090909  7.7363636
## [499] 27.8545455 22.9181818
```

```r
# 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%
##  3.572273 29.753182
```

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

```
##     2.5%    97.5%
##  1.68000 32.47773
```

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

```
##      0.5%     99.5%
## -1.877636 37.037636
```

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
```

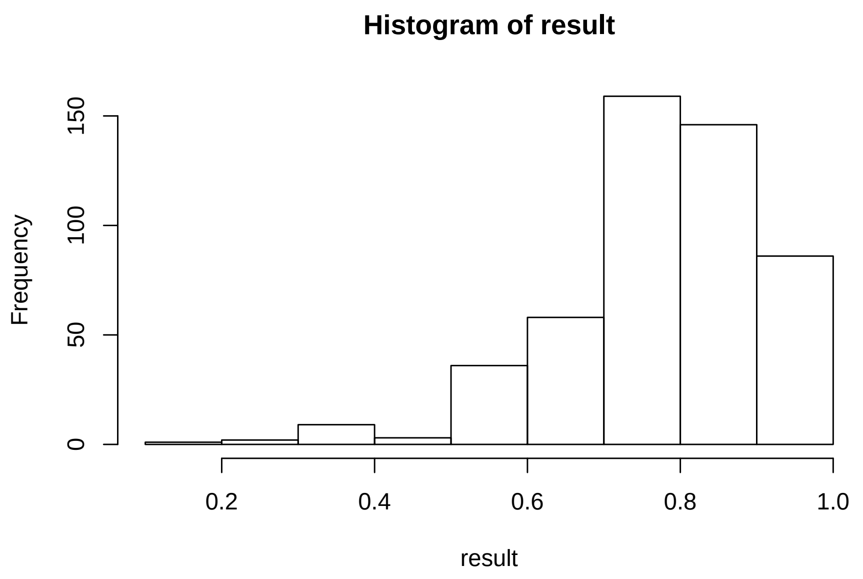
In the following, I will write my own code to execute the bootstrapping. I set the bootstrapping number only 500, for illustrative purposes. As we can see, the distribution is not symmetrical.

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/

```r
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

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

head(fHH1)
```

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

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

Thus,

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

Thus,

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

Thus,

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

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

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

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

```
##
## Call:
## glm(formula = total ~ age, family = poisson, data = fHH1)
##
## 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
```

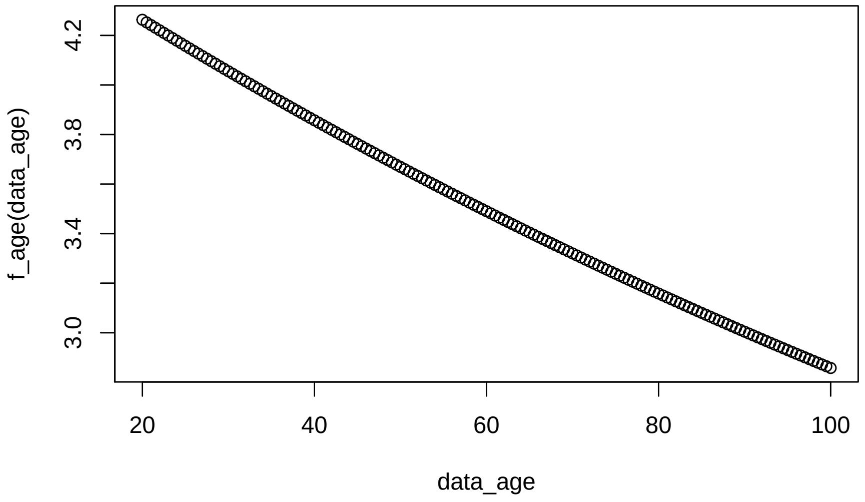
Note that, we use original beta to construct a confidence interval and then exponentiate the endpoints is due to the fact that the oringal one is more close to normal distribution.

6.2 Trying to understand

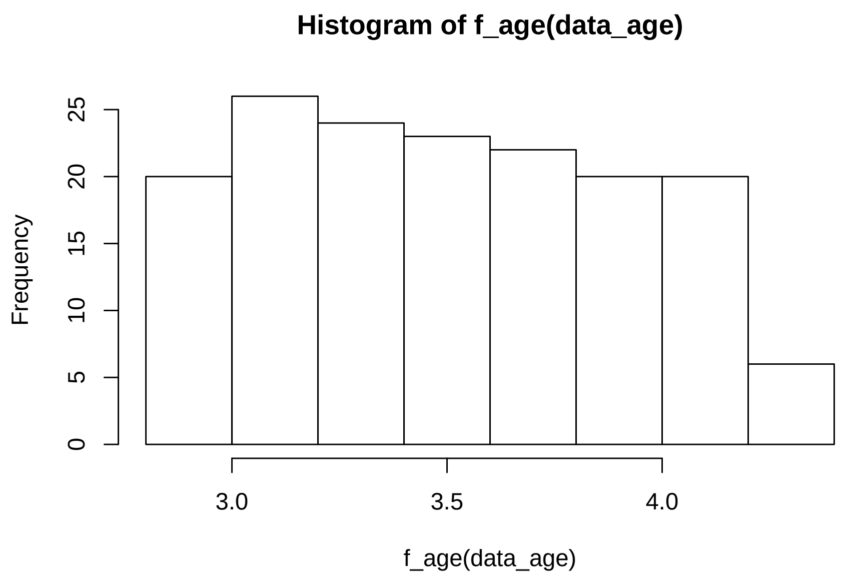
With $\hat{\beta}_0 = 1.55$ and $\hat{\beta}_1 = -0.005$, we can write down the following. I also simulated the data and showed the relationship between X and Y. As we can see the figure, the relationship is pretty linear. Thus, something to keep in mind, the poisson distribution we typically see is the histogram of Y, rather than the relationship between X and Y.

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

```r
data_age<-seq(20,100,0.5)
f_age<-function(x){exp(1.55-(0.005*x))}
# cbind(data_age,f_age(data_age))
plot(data_age,f_age(data_age))
```



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


Histogram of f_age(data_age)
Frequency
0
5
10
15
20
25
3.0
3.5
4.0
f_age(data_age)

Chapter 7

Use R for mediation

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