SEM and R

Bill

2021-05-05

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).

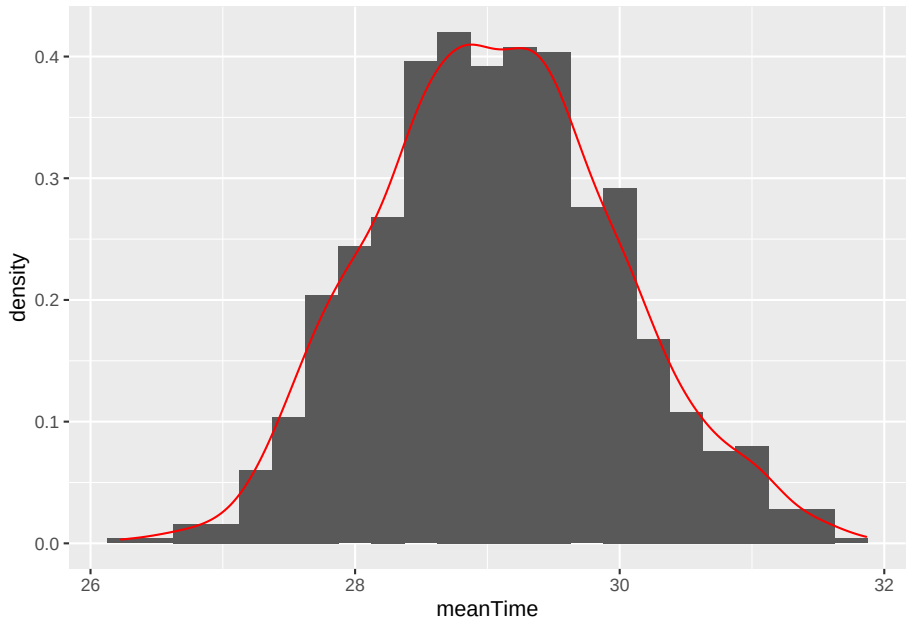
```
B = 1000
n = nrow(CommuteAtlanta)
boot.samples = matrix(sample(CommuteAtlanta$Time, size = B * n, replace = TRUE),
                      B, n)
```

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] 0.9213917
```

```
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.

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

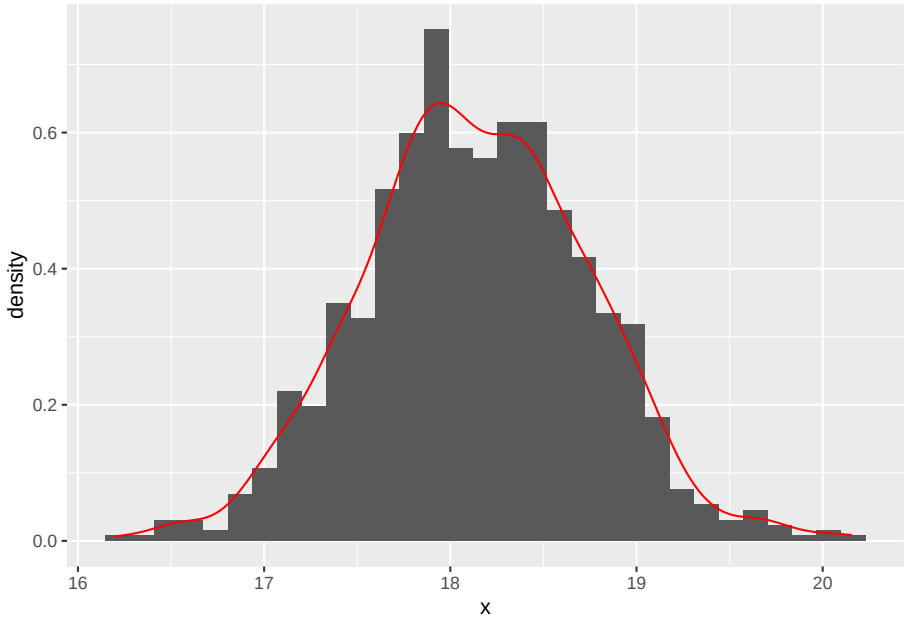
Note that, in the following, the author used 2 times SE to calculate the CI. The relationship between SD and SE:

"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] 16.9441 19.3679
```

5.4 Sample function

To understand the function of sample in R.

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

```
##  [1] 13 17 12  2 16  9  1  6 16  1 18  6 10  2  2 12  6  7  5 13
```

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)
```

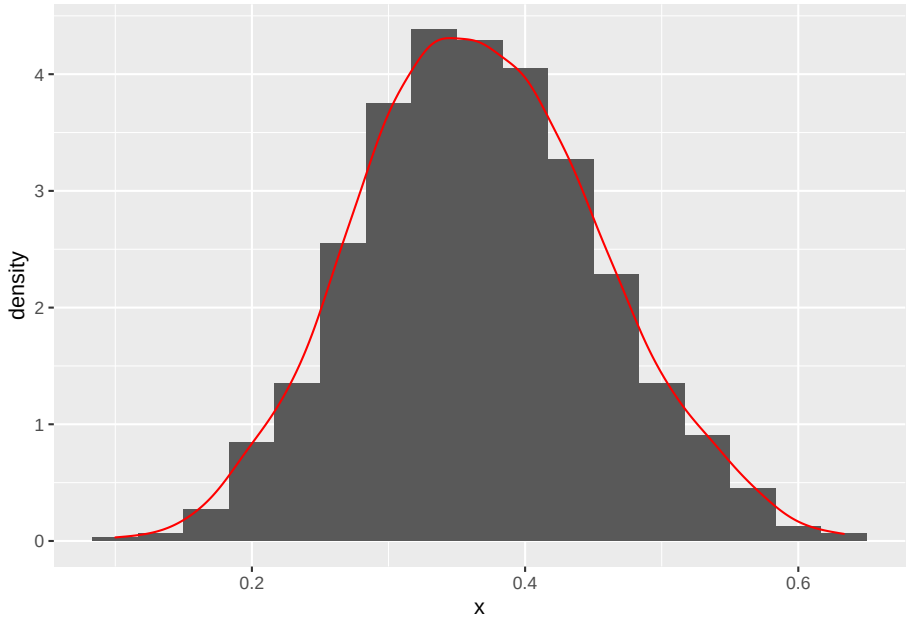
```
{
boot.sample = sample(n, replace = TRUE)
result[i] = mean(CommuteAtlanta$Distance[boot.sample])
}

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

```
## [1] 16.92454 19.38746
```

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)
```

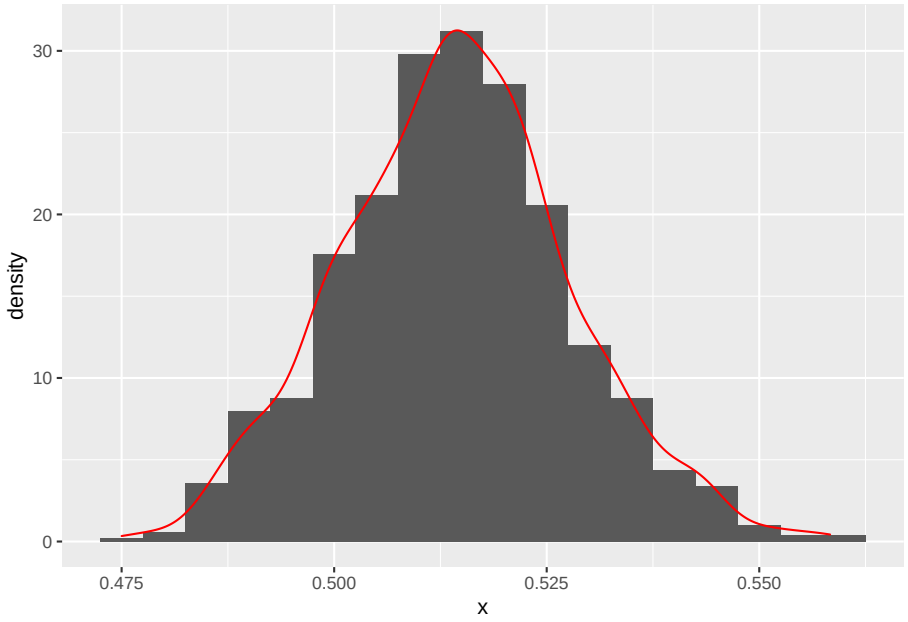



```
## [1] 0.1913839 0.5419494
```

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] 0.4876415 0.5415252
```

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%   (27.31, 30.90 )   (27.20, 30.83 )
##
## Level     Percentile            BCa
## 95%   (27.39, 31.02 )   (27.46, 31.09 )
## 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] 21.2090909 16.9727273 10.8363636  7.3727273  8.6000000 18.8636364
##   [7] 22.2454545 13.7363636 22.5181818 25.4000000 16.1363636  6.4272727
```

```
##  [13] 24.6636364  8.1454545 27.3636364 24.2727273 25.3000000 -0.6818182
##  [19] 13.6909091 24.8636364 12.0545455 15.9818182  4.4090909 23.2363636
##  [25] 22.5636364 18.6909091 18.2727273 16.2818182 24.7909091 10.4272727
##  [31] 21.2363636 -2.1000000 12.2909091 11.5909091  7.9272727  9.6272727
##  [37]  8.3090909  9.2909091 13.7636364 21.5727273 22.1272727 25.3454545
##  [43] 19.1636364  7.6272727 11.2272727 20.8090909 17.8727273 12.7090909
##  [49] 23.0181818 21.6545455 22.8545455  5.2636364 24.7545455 27.4363636
##  [55]  7.2272727 21.6909091 15.1363636 16.8000000 19.1454545 10.7272727
##  [61] 13.5272727 13.8727273 16.9727273 12.4454545 10.5909091 17.7818182
##  [67] 24.1818182 22.1818182  7.7909091 -1.3000000 11.4090909 15.6272727
##  [73] 13.8272727 15.7272727 26.1727273 13.3545455 17.3727273 20.4363636
##  [79]  1.7363636 23.2636364  8.6454545 17.9545455 19.0181818 18.7000000
##  [85] 18.5818182 18.7636364 13.3000000  9.3545455 16.4000000 20.2818182
##  [91] 18.3090909 14.8454545 14.0454545 17.7272727 21.7181818 11.7909091
##  [97] 20.4636364 25.4090909 13.7636364 18.4090909 11.7272727 12.1181818
## [103] 21.2727273 19.4090909 11.7545455  8.8818182 33.7454545 10.6727273
## [109] 14.1636364 19.9818182 17.9000000  2.4090909 17.1363636 35.9181818
## [115] 16.8545455 11.4454545 24.7818182  0.6272727 20.4818182  0.5000000
## [121] 17.1272727 19.4545455 17.8272727 36.3181818 14.8545455 20.4636364
## [127] 25.4545455 11.4909091 13.9545455 17.5727273  8.3636364 25.0909091
## [133] 16.7545455 20.8363636 26.6363636 21.9636364 12.4545455 26.0272727
## [139] 21.4454545 14.7454545 11.2090909 26.2000000  9.7545455 13.3727273
## [145] 25.9181818 20.2000000 25.2000000 17.1454545 19.6909091 23.2636364
## [151] 23.3000000 11.2272727 27.8909091 24.5454545 19.2909091 28.9545455
## [157] 11.4363636 16.4181818 22.8454545 24.4090909 26.8000000 21.4636364
## [163] 23.5545455 17.8272727  7.3727273 23.1636364  8.9272727  8.5000000
## [169] 22.6818182 12.0090909  5.0727273 17.2272727 24.5454545 11.7454545
## [175] 22.5909091 15.9545455  9.9545455 13.8363636 27.6818182 14.7181818
## [181] 25.7272727 19.5000000 13.3818182 19.7818182 11.3363636 28.4545455
## [187]  5.3090909 20.9636364 11.2727273 25.5363636 31.2363636 14.1090909
## [193] 17.4545455 17.5090909 17.1272727 30.4636364 14.4727273 12.3000000
## [199] 18.5818182 17.1909091 17.6272727 24.4272727 33.0818182 10.0727273
## [205] 32.4454545 18.0545455 15.0000000  6.9363636 21.6545455 27.0909091
## [211]  8.7181818 16.6636364  9.3272727  8.8090909 13.8909091 21.2181818
## [217] 20.5000000  7.9090909 19.3181818  2.2090909 22.0454545  9.2909091
## [223] 25.7545455  7.1454545 22.9545455 12.1545455 20.1363636  9.3818182
## [229] 24.0818182 13.2272727  7.9818182 19.0727273 27.7545455 10.7272727
## [235] 12.5181818 10.5454545  8.0000000 20.3363636 21.5636364  6.9000000
## [241] 27.5363636  2.6363636 20.9272727 29.8636364 11.9181818  3.3545455
## [247] 17.3363636 22.0272727 24.5636364 23.0818182 15.3090909 21.2000000
## [253] 26.7818182 19.5454545 30.1727273 35.3272727 17.0090909 19.6727273
## [259] 23.6181818  4.8727273 14.8818182 16.0090909 17.7363636 17.3454545
## [265] 24.0272727 15.0818182  8.0909091 16.5727273 25.5818182 18.0545455
## [271] 19.1363636 21.8272727 25.4818182 38.3545455 22.1000000 14.3454545
## [277] 13.9181818 24.8909091 16.5454545 27.7181818 26.6363636 18.1454545
## [283] 20.5363636 12.6363636 18.2545455 23.3909091 13.8818182  8.4818182
```

```
## [289] 10.3000000 20.6272727  9.5363636 24.1909091 12.6090909 14.9363636
## [295] 16.3636364 -8.9909091 28.4727273 11.7090909 13.3909091 12.1909091
## [301] 13.2090909 25.1363636  9.6000000 25.3454545 28.1909091 26.5818182
## [307] -5.4909091 25.2454545  7.0090909 14.8272727 23.0454545  7.3818182
## [313] 18.4818182  8.7090909 17.6363636 15.8818182 13.9090909 35.0636364
## [319] 10.8545455 13.0636364 16.7818182 20.0909091 22.9272727 14.6272727
## [325] 21.1909091 32.4181818 19.7636364 11.9090909 39.2818182 14.8090909
## [331] 14.0818182 15.6090909 11.7909091 23.4181818  6.7000000 17.5000000
## [337] 25.5636364 24.2909091 20.6545455  6.9272727 10.7636364 17.9818182
## [343] 18.1181818 10.0636364 32.5454545 31.8000000 31.4545455 19.3636364
## [349] 11.9181818 28.5090909 20.8181818 12.2636364 19.6272727 14.5636364
## [355] 19.7454545 17.6636364 13.1000000  5.1909091 33.3363636 10.5090909
## [361]  9.5454545 18.9636364 26.2909091 17.6909091 29.7818182  5.6000000
## [367] 28.5818182 11.6000000  4.7000000 28.2545455 12.1909091 16.5181818
## [373] 16.2272727  7.3909091 10.5181818  6.8090909 18.6909091 20.1636364
## [379] 19.0454545  7.8000000 29.8818182 21.6000000 15.6090909  8.9181818
## [385] 22.7454545 25.0545455 11.3636364 11.7636364 15.9545455 19.5727273
## [391] 16.8272727  9.7000000 32.6090909 19.2363636 22.8727273 18.9454545
## [397] 20.3636364 24.4181818 16.3818182 10.9272727 25.6181818  7.7909091
## [403] 16.4909091 24.7363636 10.7454545 27.0363636 19.9909091 29.4000000
## [409] 17.2000000  5.1272727 13.6727273 11.4909091 17.3818182 25.4818182
## [415]  4.0000000 12.2363636 23.8818182  2.1909091 20.1545455 24.1727273
## [421] 18.2727273 10.9818182 22.4000000  8.5090909 23.1909091 22.0909091
## [427] 21.2636364  9.8818182 16.1909091 15.2636364  7.2090909 14.0181818
## [433] 16.0454545 15.0727273 13.6818182 14.9454545 25.8636364 29.8363636
## [439] 17.3000000 28.8909091  9.4000000 14.1181818 13.9000000  6.4909091
## [445]  5.6545455 21.4000000 19.8272727 19.8909091 12.6909091 10.9090909
## [451]  6.0000000 21.7181818 13.0727273 18.3818182  3.8181818  7.8454545
## [457]  9.2363636  9.3545455 20.1090909 15.5181818 18.8090909 23.7000000
## [463]  9.9000000 20.1000000 15.9909091 11.0454545 22.1636364 30.3636364
## [469] 18.9272727 21.7818182 29.3727273 16.7727273 11.0272727 11.1818182
## [475] 10.7363636 26.4000000 26.8545455 16.4454545 25.3000000 23.2454545
## [481]  7.1090909 17.0000000 33.2727273 27.4363636 18.7545455 13.5909091
## [487]  7.5000000 26.8363636 30.3363636 14.1545455 13.5181818 21.9363636
## [493]  3.0272727 28.3454545  8.0363636 22.7363636 12.4727273 17.4090909
## [499]  5.0727273 20.2090909
```

```
# 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%
##  5.585455 29.419091
```

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

```
##      2.5%     97.5%
##  3.182727 32.432500
```

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

```
##     0.5%    99.5%
## -1.70400 36.12018
```

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
```

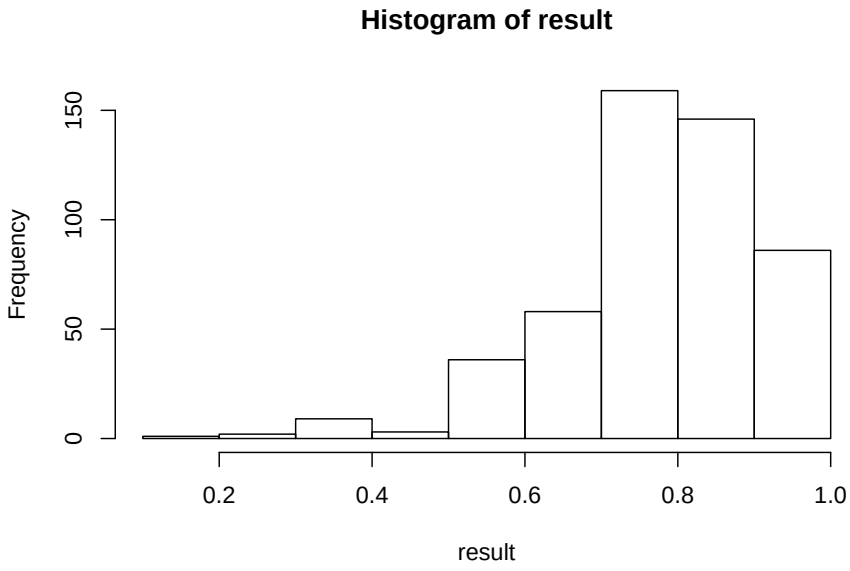
In the following, I will write my own code to execute the bootstrapping. I set the bootstrapping number only 500, for illustrative purposes. As we can see, the distribution is not symmetrical.

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
```

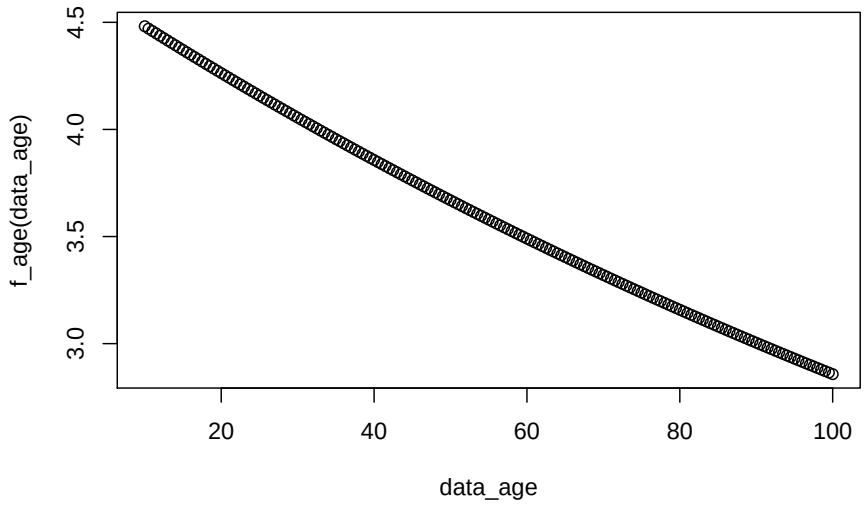
```
##  [30,]     24.5 4.168265
##  [31,]     25.0 4.157858
##  [32,]     25.5 4.147476
##  [33,]     26.0 4.137120
##  [34,]     26.5 4.126791
##  [35,]     27.0 4.116486
##  [36,]     27.5 4.106208
##  [37,]     28.0 4.095955
##  [38,]     28.5 4.085728
##  [39,]     29.0 4.075527
##  [40,]     29.5 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
```

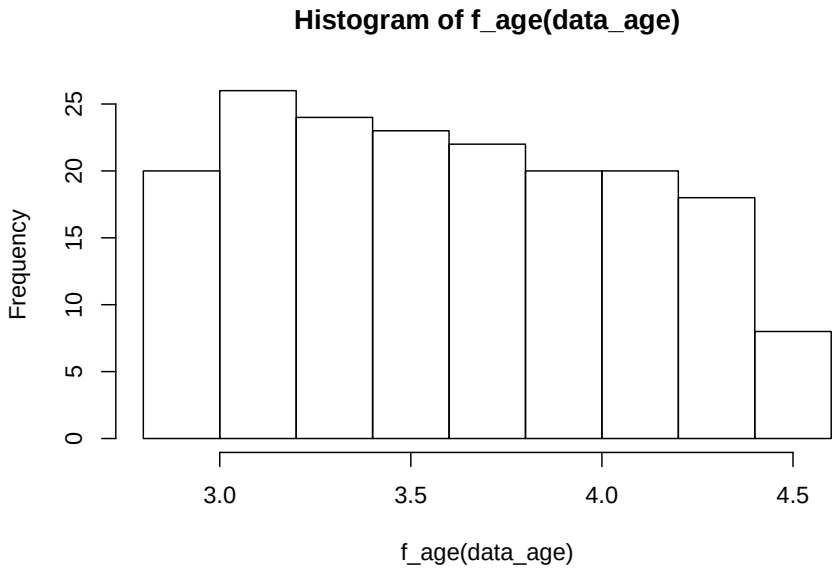
```
## [122,]     70.5 3.311827
## [123,]     71.0 3.303558
## [124,]     71.5 3.295309
## [125,]     72.0 3.287081
## [126,]     72.5 3.278874
## [127,]     73.0 3.270687
## [128,]     73.5 3.262520
## [129,]     74.0 3.254374
## [130,]     74.5 3.246248
## [131,]     75.0 3.238143
## [132,]     75.5 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
```

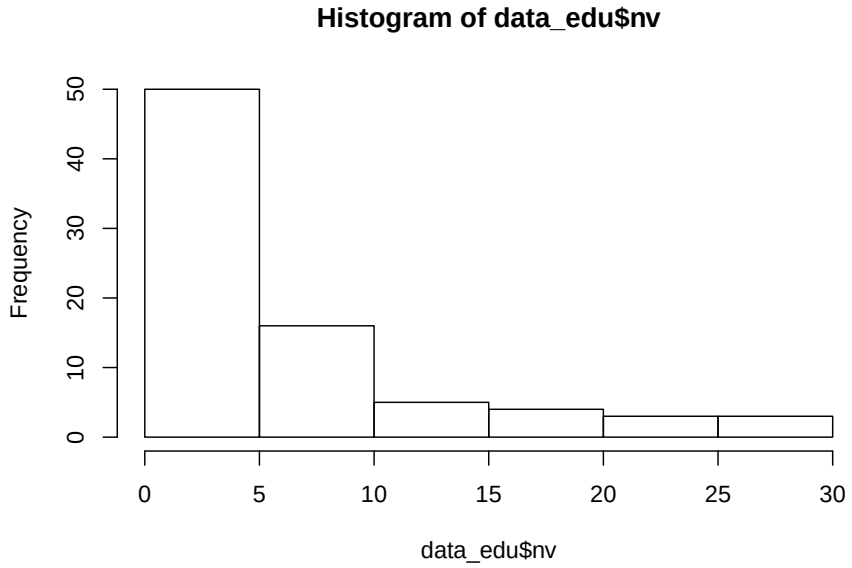
6.4 Overdispersion (using another example)

```
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)
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

https://advstats.psychstat.org/book/mediation/index.php https://bookdown.org/roback/bookdown-BeyondMLR/ch-poissonreg.html