# Weird Statistics Questions

Hai Liang - hailiang@cuhk.edu.hk

11/26/2021

# Contents

# List of Figures



# List of Tables

# I. Descriptive Statistics

### 1. How to represent and summarize a variable? Why do we calculate mean and standard deviation for a variable? How about if the variable is skewed or discrete?

The most accurate way to represent a variable is to visualize it as a distribution histogram (distribution is close to the raw data). We usually use *mean* ($\mu$) to quantify the central tendency (typical value) of the distribution and *standard deviation* ($\sigma$) to characterize the degree of dispersion/spread. In addition, we use quantiles to measure locations of a distribution.

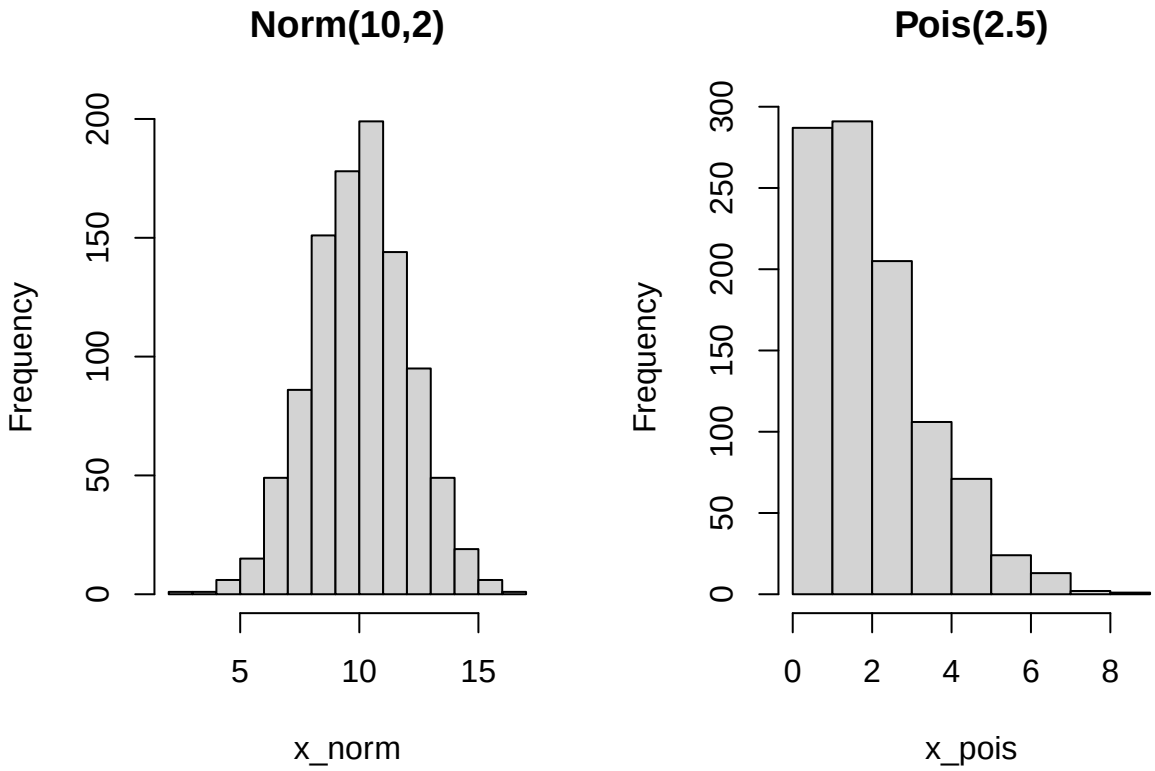
Notes:

- *mean* is just one of the measures of central tendency. Another two common measures are *mode* and *median*. Which measure is the most appropriate to use depends on the questions and also the shape of the distribution.
- *standard deviation* is defined according to the *mean*. If *mean* is not clearly defined, there is no *standard deviation* either. Given the association between *mean* and *standard deviation*, it is also meaningless to compare *standard deviations* across data sets. To compare dispersion across groups, we can use the *coefficient of variation*: $CV = \sigma/\mu$.
- if the distribution curve can be described mathematically using a distribution formula, the parameters of the formula will be the accurate measures.

For example, a Normal distribution (bell curve) could be described as $X \sim \mathsf{Norm}(\mu, \sigma)$, while a Poisson distribution (right skewed curve) could be described as $X \sim \mathsf{Pois}(\lambda)$. In a Normal distribution, *mean* and *standard deviation* are the two parameters that determine the distribution. Given the popularity of Normal distribution in the real world, the two parameters are commonly used measures to characterize data sets (represented as distribution histogram). In a Poisson distribution, $\lambda$ is the only parameter (rate parameter), which is the total number of events ($k$) divided by the number of units ($n$) in the data ($\lambda = k/n$).

In R, we use `rnorm` and `rpois` functions to generate Normal and Poisson distributions respectively, by setting the respective parameters.

```
x_norm = rnorm(1000, 10, 2) # mean = 10, sd = 2
x_pois = rpois(1000,2.5) # lambda = 2.5
par(mfrow=c(1,2))
hist(x_norm,main="Norm(10,2)")
hist(x_pois,main="Pois(2.5)")
```




As expected, the *mean*s of the generated data are close to the theoretical ones:

```
c(mean(x_norm),mean(x_pois))
```

```
## [1] 10.01864  2.44000
```

It is interesting to calculate the *standard deviation*s:

```
c(sd(x_norm),sd(x_pois))
```

```
## [1] 2.03030 1.56296
```

The *standard deviation* of the Poisson distribution is 1.56. The variance is the *standard deviation*$^2$ = 2.44, which is close to the *mean* $\lambda$. In fact, $E[X] = Var[X]$ for Poisson distribution. We will explain the formula soon below.

Question:

- As suggested in some text books, we should use *median* or *mode* to characterize the central tendency of skewed distributions. The median for x_pois is 2. Is this value informative, better than the *mean*?

*mean* and thus *standard deviation* are not well defined for categorical variables. The distribution could be described as $f(x = k) = p_k$. We use the `sample` function to generate two categorical variables by defining $p_k$s.

```r
x_cat1 = sample(8:12, size = 1000, replace = TRUE, prob = c(0.25,0.1,0.3,0.25,0.1))
x_cat2 = sample(8:12, size = 1000, replace = TRUE, prob = c(0.1,0.25,0.3,0.1,0.25))
c(mean(x_cat1),mean(x_cat2)) #means
```

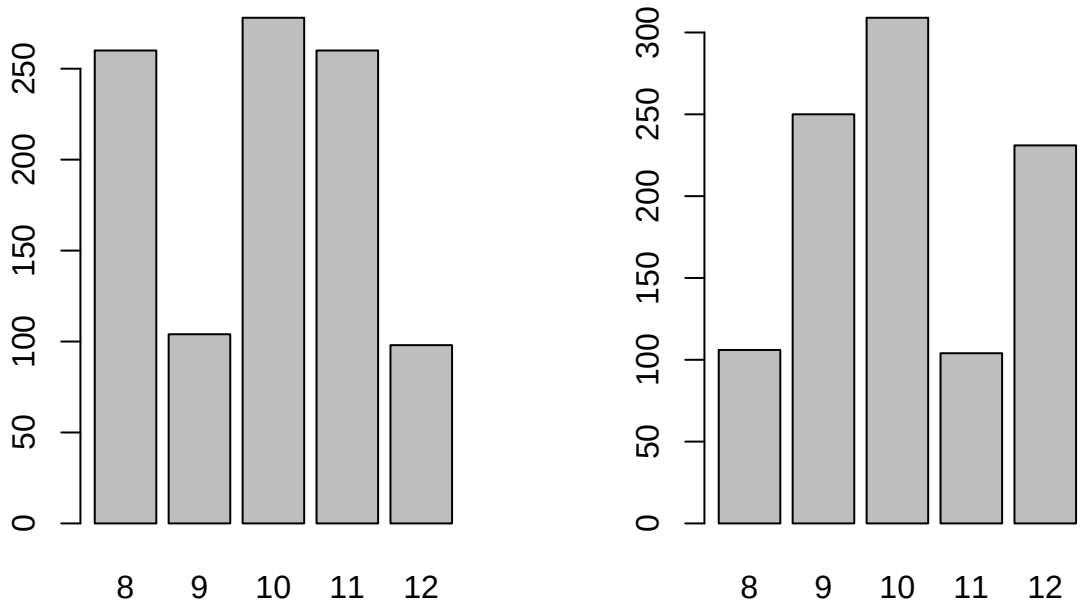
```
## [1]  9.832 10.104
```

```r
c(sd(x_cat1),sd(x_cat2)) #standard deviations
```

```
## [1] 1.330243 1.301106
```

Even though the *means* are hard to interpret (what does it mean if I say the average of race in Hong Kong is 3.1?), the above calculation indicates that the two variables have similar *means* and *standard deviations*. They are actually two different distributions, as presented below:

```r
tbl_cat1 = table(x_cat1)
tbl_cat2 = table(x_cat2)
par(mfrow=c(1,2))
barplot(tbl_cat1)
barplot(tbl_cat2)
```




The best way is to calculate the frequency tables:

```
tbl_cat1 = data.frame(tbl_cat1)
tbl_cat1$Prob_x1 = tbl_cat1$Freq/sum(tbl_cat1$Freq)
tbl_cat1$Category = paste0(tbl_cat1$x_cat1)
t1 = tbl_cat1[,c("Category","Prob_x1")]

tbl_cat2 = data.frame(tbl_cat2)
tbl_cat2$Prob = tbl_cat2$Freq/sum(tbl_cat2$Freq)
t1$Prob_x2 = tbl_cat2$Prob

t1
```

```
##   Category Prob_x1 Prob_x2
## 1        8   0.260   0.106
## 2        9   0.104   0.250
## 3       10   0.278   0.309
## 4       11   0.260   0.104
## 5       12   0.098   0.231
```

Question:

Are there any continuous distributions without *mean* or *variance*? See Pareto distribution.

If there are many distributions without "meaningful" *mean* or *variance*, why are they so popular measures in statistics? Let's introduce another concept – Expected Value.

- For discrete distributions: $E[X] = \sum_{k=1}^{n} x_k P(X = x_k)$.
- For continuous distributions: $E[X] = \int_{-\infty}^{\infty} x f(x) dx$.

```
x = c(8:12)
px_cat1 = tbl_cat1$Prob_x1 # estimated probs
px_cat2 = c(0.1,0.25,0.3,0.1,0.25) # true probs

EX_cat1 = sum(x*px_cat1) # expectation
EX_cat2 = sum(x*px_cat2) # expectation

EX_cat1 - mean(x_cat1) # the same
```

```
## [1] 0
```

```
EX_cat2 - mean(x_cat2) # slightly different
```

```
## [1] 0.046
```

Are there any cases that $E[X] \neq mean(X)$? Biased samples! The expectation, by definition, is true (of he population), while *mean* might not be true in biased samples.

For Normal distribution, $E[X] = \mu \approx mean$. For Poisson distribution, $E[X] = \lambda \approx mean$. If expectation is the only parameter for the distribution (e.g., $\lambda$ in Poisson), the *mean* suffices to characterize the distribution. That means if we know the parameter $\lambda$, we can recover the distribution using the formula: $\mathsf{Pois}(\lambda) = \frac{\lambda^k e^{-\lambda}}{k!}$. However, *mean* ($\mu$) is not sufficient to characterize a Normal distribution because it varies with *standard deviation* ($\sigma$) too. For categorical distribution, we need $k-1$ probabilities ($p_k$) to characterize the distribution.

Specifically, a binary variable with 0 and 1, we only need 1 probability (the proportion of 1) to describe the distribution ($E[X] = p$, $Var[X] = E[(X - E[X])^2] = p(1-p)$).

Therefore, to describe a (sampled) data precisely, we need to know the underlying distribution (that generated the observed data) first. And then, we could estimate the corresponding parameters based on the sample. But what should we do if we don't know the underlying distribution?

If the function is a probability distribution, then the first moment is the expected value (*mean*), the second central moment is the variance (*sd*$^2$: $Var[X] = E[(X - E[X])^2]$), the third standardized moment is the skewness (*skewness*), and the fourth standardized moment is the kurtosis (*kurtosis*)... Even if the $E(X)$ is not directly interpretable, it remains important when we consider the relationships between a sample and the population (see Q3).

Nevertheless, *mean* and *sd* could be interpreted in skewed distributions. Chebyshev's inequality says that at least $1 - 1/K^2$ of data from a sample must fall within $K$ standard deviations from the *mean*, where $K$ is any positive real number greater than one.

- For $K = 2$ we have $1 - 1/K^2 = 1$ - $1/4 = 3/4 = 75\%$. So Chebyshev's inequality says that at least 75% of the data values of any distribution must be within two *standard deviation*s of the *mean*.
- For $K = 3$ we have $1 - 1/K^2 = 1$ - $1/9 = 8/9 = 89\%$. So Chebyshev's inequality says that at least 89% of the data values of any distribution must be within three *standard deviations* of the *mean*.
- For $K = 4$ we have $1 - 1/K^2 = 1$ - $1/16 = 15/16 = 93.75\%$. So Chebyshev's inequality says that at least 93.75% of the data values of any distribution must be within four *standard deviation*s of the *mean*.

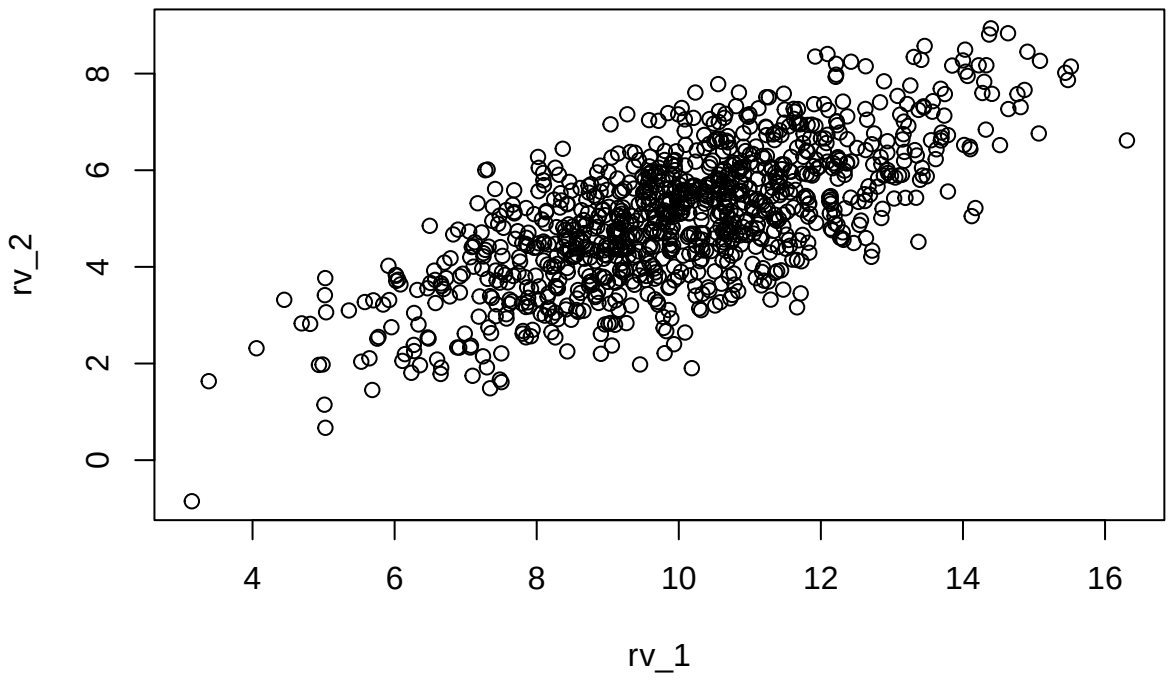
**2. We use correlation coefficients (why plural?) to quantify the (linear) relationships between two continuous variables, how can we show the relationships involving categorical/rank variables?**

```
rv_1 = rnorm(1000,10,2)
rv_2 = 0.5*rv_1 + rnorm(1000)

cor.test(rv_1,rv_2)
```

```
##
##  Pearson's product-moment correlation
##
## data:  rv_1 and rv_2
## t = 30.649, df = 998, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.6629429 0.7269324
## sample estimates:
##       cor
## 0.6963188
```

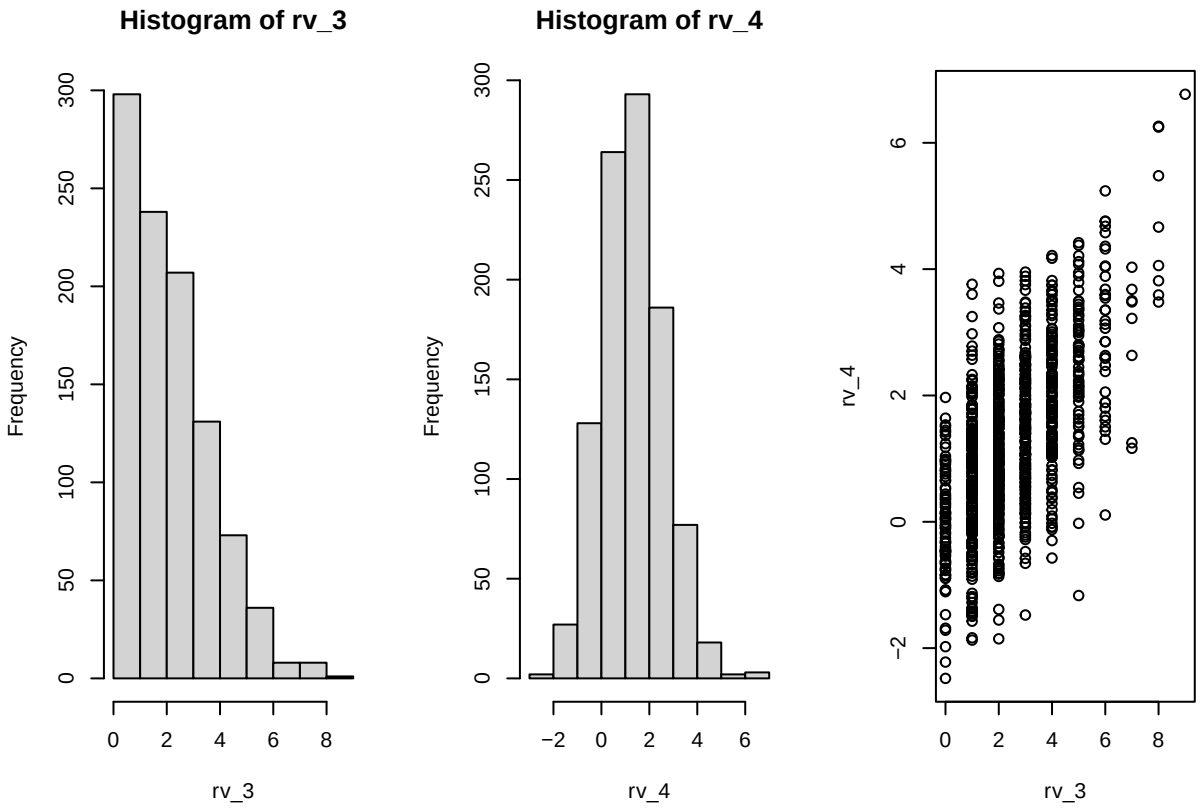
```
plot(rv_1,rv_2)
```




The Pearson coefficient $r = 0.7$ is larger than 0.5. Why? $r^2 = 0.48 \approx 0.5$. Now, we generate a non-normal variable rv_3.

```
rv_3 = rpois(1000,2.5)
rv_4 = 0.5*rv_3 + rnorm(1000)

par(mfrow=c(1,3))
hist(rv_3)
hist(rv_4)
plot(rv_3,rv_4)
```




We have different ways to calculate the correlation between two continuous variables (rank, interval, or ratio): Pearson's $r$, Spearman's $\rho$, and Kendall's $\tau$.Please check the method definitions here.

```
cor.test(rv_3,rv_4,method = "pearson")
```

```
##
##  Pearson's product-moment correlation
##
## data:  rv_3 and rv_4
## t = 24.356, df = 998, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.5701737 0.6480477
## sample estimates:
##       cor
## 0.6105846
```

```
cor.test(rv_3,rv_4,method = "spearman")
```

```
## Warning in cor.test.default(rv_3, rv_4, method = "spearman"): Cannot compute
## exact p-value with ties
```

```
##
##  Spearman's rank correlation rho
##
```

```
## data:  rv_3 and rv_4
## S = 71509120, p-value < 2.2e-16
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##       rho
## 0.5709449
```

```
cor.test(rv_3,rv_4,method = "kendall")
```

```
##
##  Kendall's rank correlation tau
##
## data:  rv_3 and rv_4
## z = 19.021, p-value < 2.2e-16
## alternative hypothesis: true tau is not equal to 0
## sample estimates:
##       tau
## 0.4345938
```

we can see that Pearson and Spearman correlations are roughly the same, but Kendall is very much different. That's because Kendall is a test of strength of **dependence** (i.e. one could be written as a linear function of the other), whereas Pearson and Spearman are nearly equivalent in the way they correlate normally distributed data. All of these correlations are correct in their result, it's just that Pearson and Spearman are looking at the data in one way, and Kendall in another.

In fact, the Pearson' $r$ of the ranks of the original variables is equivalent to Spearman's $\rho$:

```
cor(rv_1,rv_2,method = "pearson") # pearson
```

```
## [1] 0.6963188
```

```
cor(rank(rv_1),rank(rv_2),method = "pearson") # ranks -> pearson
```

```
## [1] 0.6557537
```

```
cor(rv_1,rv_2,method = "spearman") #spearman
```

```
## [1] 0.6557537
```

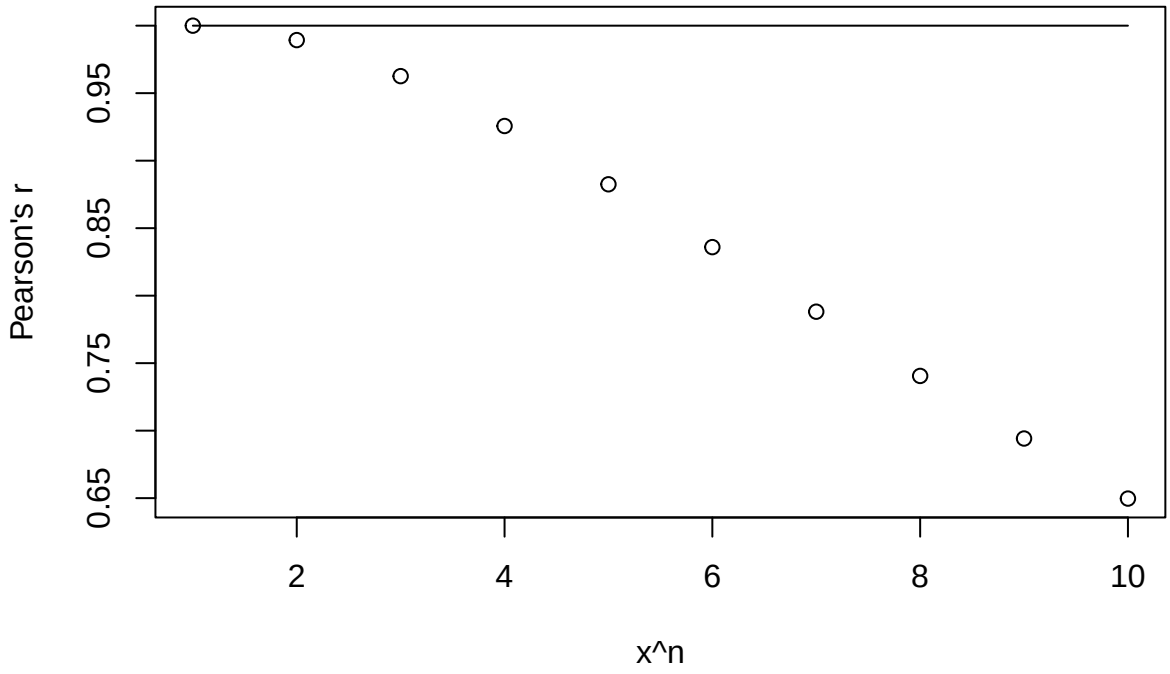
Pearson's correlation ($r$) is a measure of the **linear** relationship between two continuous random variables. It does not assume normality although it does assume finite variances and finite covariance ($Cov(X,Y) = E[(X-E(X))(Y-E(Y))] = E[XY]-E[X]E[Y]$). When the variables are Bivariate Normal (the sampling distribution of $r\sqrt{\frac{n-2}{1-r^2}}$ will follow a Student's t-distribution), Pearson's correlation provides a complete description of the association. In order words, violation of Bivariate Normal may influence the significance tests (when sample size is small).

Spearman's correlation ($\rho$) applies to ranks and so provides a measure of a **monotonic** (could be non-linear) relationship between two continuous random variables. It is also useful with ordinal data and is robust to outliers (unlike Pearson's correlation). When correlating skewed variables, particularly highly skewed variables, a log or some other transformation often makes the underlying relationship between the two variables clearer. In such settings it may be that the raw metric is not the most meaningful metric

anyway. Spearman's $\rho$ has a similar effect to transformation by converting both variables to ranks. From this perspective, Spearman's $\rho$ can be seen as a quick-and-dirty approach (or more positively, it is less subjective) whereby you don't have to think about optimal transformations.

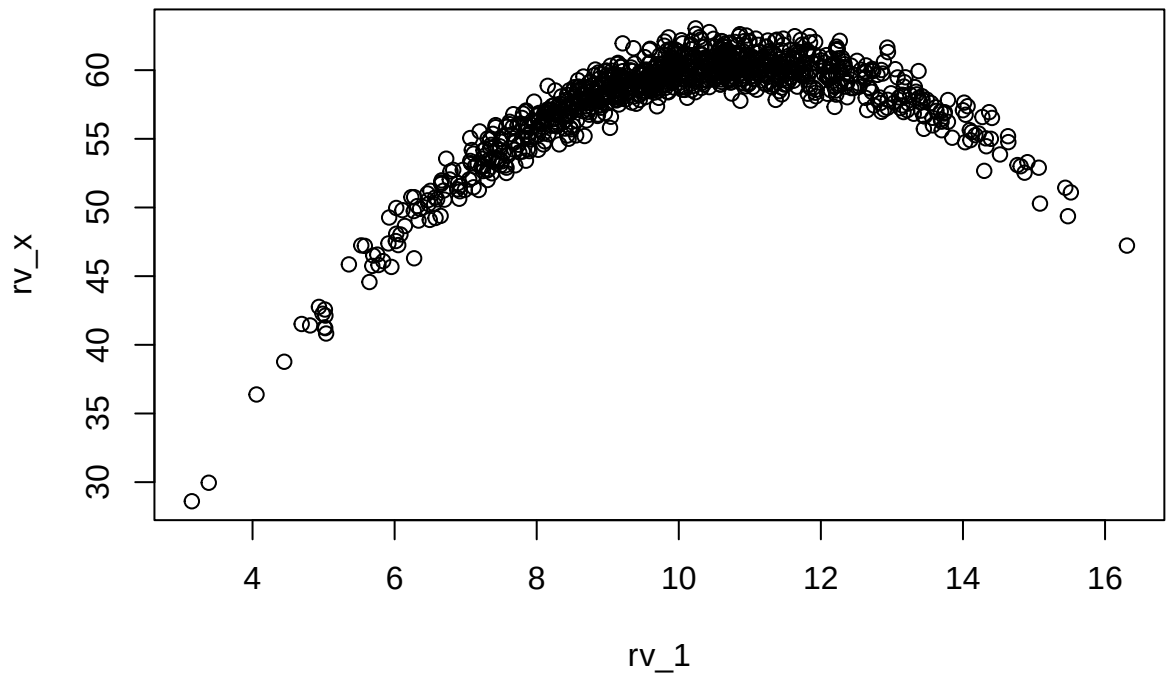
The two coefficients will be very different, if the relationship between the variable is non-linear. The following R code shows how the correlations between $x$ and $x^n$ differ over $n$.

```
rs = sapply(1:10,function(i)cor(rv_1,rv_1^i,method = "pearson"))
rhos = sapply(1:10,function(i)cor(rv_1,rv_1^i,method = "spearman"))
plot(rs,ylab = "Pearson's r",xlab="x^n")
lines(rhos)
```




Spearman's $\rho$ vs. Kendall's $\tau$ (tau). These two are so much computationally different that you cannot directly compare their magnitudes. Spearman is usually higher by 1/4 to 1/3 and this makes one incorrectly conclude that Spearman is "better" for a particular dataset. The difference between $\rho$ and $\tau$ is in their ideology, proportion-of-variance for $\rho$ and probability for $\tau$. $\rho$ is a usual Pearson's $r$ applied for ranked data. $\tau$ is more "general" than $\rho$, for $\rho$ is warranted only when you believe the underlying (model, or functional in population) relationship between the variables is strictly **monotonic**. While $\tau$ allows for **non-monotonic** underlying curve and measures which monotonic "trend", positive or negative, prevails there overall. $\rho$ is comparable with $r$ in magnitude; $\tau$ is not.

```
rv_x = -0.5*rv_1^2+11*rv_1+rnorm(1000) # it is non-linear non-monotonic
plot(rv_1,rv_x)
```




```
cor.test(rv_1,rv_x, method = "pearson")
```

```
##
##  Pearson's product-moment correlation
##
## data:  rv_1 and rv_x
## t = 21.271, df = 998, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.5143361 0.5997478
## sample estimates:
##       cor
## 0.5585206
```

```
cor.test(rv_1,rv_x, method = "spearman")
```

```
##
##  Spearman's rank correlation rho
##
## data:  rv_1 and rv_x
## S = 80978148, p-value < 2.2e-16
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##       rho
## 0.5141306
```

```
cor.test(rv_1,rv_x, method = "kendall")
```

```
##
##  Kendall's rank correlation tau
##
## data:  rv_1 and rv_x
## z = 18.829, p-value < 2.2e-16
## alternative hypothesis: true tau is not equal to 0
## sample estimates:
##       tau
## 0.3976496
```

Cosine similarity (a kind of dependence!) between two vectors A and B is defined as

$$CosSim = \frac{\sum(A_i B_i)}{(\sqrt{\sum A_i^2}\sqrt{\sum B_i^2})}$$

Given any two vectors, the similarity score could be calculated in the following way:

```
a = sum(rv_1*rv_x)
b = sqrt(sum(rv_1*rv_1))*sqrt(sum(rv_x*rv_x))
a/b
```

```
## [1] 0.9855639
```

Finally, how to quantify the "correlation" (dependence is a more appropriate term here) between two categorical variables:

```
tbl = matrix(data=c(65, 45, 20, 40), nrow=2, ncol=2, byrow=T)
dimnames(tbl) = list(City=c('B', 'T'), Gender=c('M', 'F'))

tbl
```

```
##     Gender
## City  M  F
##    B 65 45
##    T 20 40
```

```
chi2 = chisq.test(tbl, correct=T)
c(chi2$statistic, chi2$p.value)
```

```
##   X-squared
## 9.298484848 0.002293435
```

$\chi^2$ is NOT normalized. In a $k \times l$ contingency table, the theoretical maximal value of $\chi^2$ is $N(min(k,l)-1)$, So, we define **Cramer's** $V$: $V = \sqrt{\frac{\chi^2}{N(min(k,l)-1)}}$. $V$ ranges from 0 to 1. Therefore,

```
sqrt(chi2$statistic/sum(tbl))
```

```
## X-squared
## 0.2338738
```

Alternatively, you can use the `cramersv` function in `lsr` package.

```
library(lsr)
cramersV(tbl)
```

```
## [1] 0.2338738
```

The **uncertainty coefficient** (also called entropy coefficient or Thiel's U) is a measure of nominal association. It is based on the concept of information entropy. the uncertainty coefficient ranges in [0,1]. And please read other methods.

```
library(DescTools)
UncertCoef(tbl, direction = "column")
```

```
## [1] 0.04435282
```

```
UncertCoef(tbl, direction = "row")
```

```
## [1] 0.04735172
```

```
UncertCoef(tbl, direction = "symmetric")
```

```
## [1] 0.04580323
```

**A final question**: is it possible that the association between categorical variables is negative?

## II. Inferential Statistics (Tests)

### 3. Why can we use a sample to infer the population, under what conditions? What is a sampling distribution? Is it observable? Why does large sample size work?

A simple random sample from a population could be used to estimate the population characteristics.

```
# generate a random variable according to normal distribution with mean = 15 and sd = 3
# assume as the population
rv_1 = rnorm(1000,15,3)
c(mean(rv_1),sd(rv_1),median(rv_1)) #true mean/sd/median
```

```
## [1] 15.094500  2.936941 15.076769
```

```
# generate random samples from rv_1 (the population)
s_1 = sample(rv_1,100)
c(mean(s_1),sd(s_1),median(s_1))
```

```
## [1] 15.385567  2.849332 15.221349
```

```
s_2 = sample(rv_1,100)
c(mean(s_2),sd(s_2), median(s_2))
```

```
## [1] 14.339807  2.549049 14.666027
```

```
s_3 = sample(rv_1,100)
c(mean(s_3),sd(s_3), median(s_3))
```

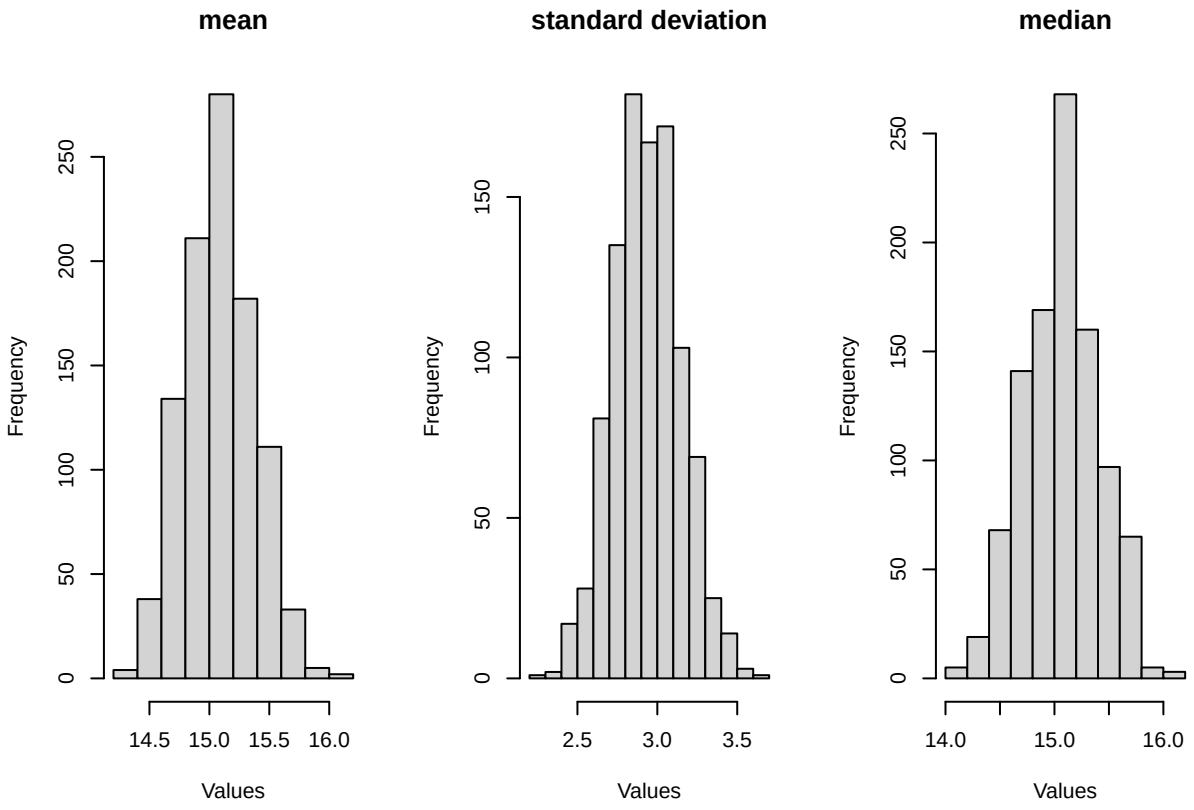
```
## [1] 15.534263  2.965561 15.694884
```

The estimates are close to the true values, thought not always the same. Let's repeat sampling many times ($N \to +\infty$) and plot the **sample means** ($\overline{X}$)–the *mean*s of all samples?

```
stats = matrix(NA,nrow = 1000,ncol = 3)

for (i in 1:1000){
  s = sample(rv_1,100) # sample size = 100
  stats[i,] = c(mean(s),sd(s),median(s))
}

par(mfrow=c(1,3))
hist(stats[,1],main="mean",xlab="Values")
hist(stats[,2],main="standard deviation",xlab="Values")
hist(stats[,3],main="median",xlab="Values")
```




We call these **sampling distributions (SMD)** of *means*, *sds*, and *medians*. All these distributions should be Normal. And then, it is straightforward to calculate the *mean* and *sd* of the distributions. For the first one, the sampling distribution of *means*:

```
mean(stats[,1])
```

```
## [1] 15.08768
```

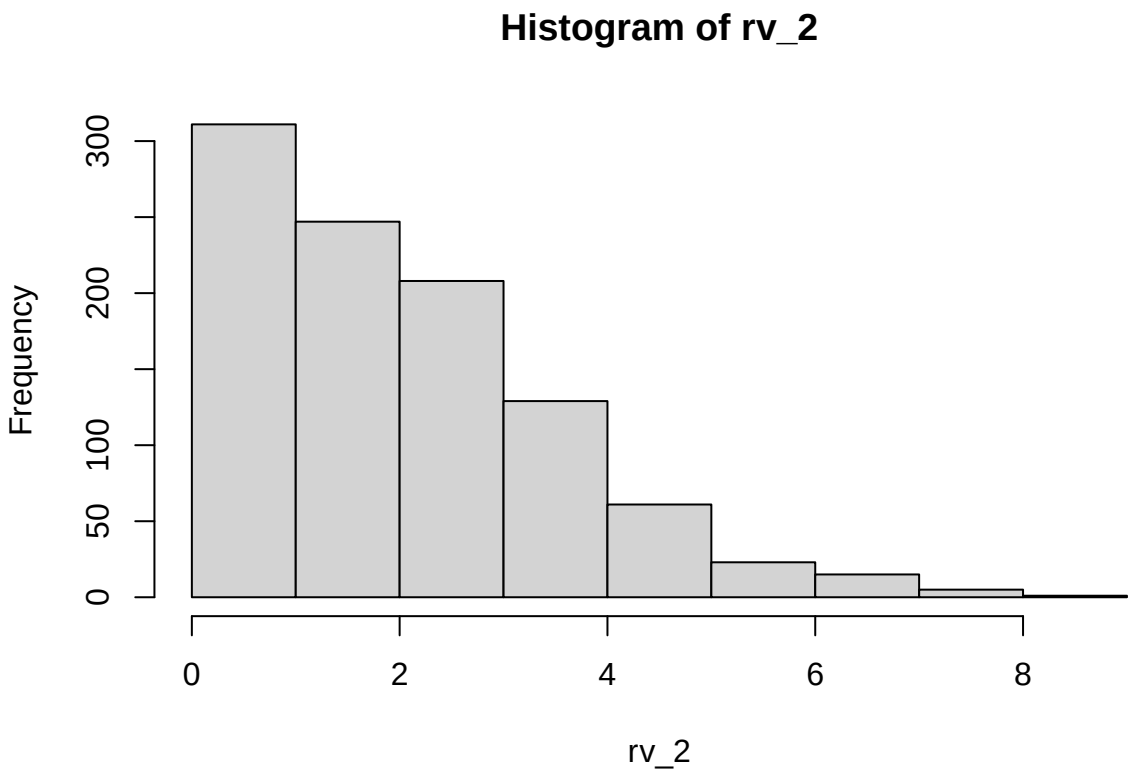
It is almost the same to the population mean. How about the means of of the second and third distributions (sampling distributions of standard deviations and medians)? The pattern remains!

```
c(mean(stats[,2]),mean(stats[,3]))
```

```
## [1]  2.936575 15.068199
```

Does the pattern apply to non-normal distributions? Let's simulate a Poisson variable:

```
rv_2 = rpois(1000,2.5)
hist(rv_2)
```

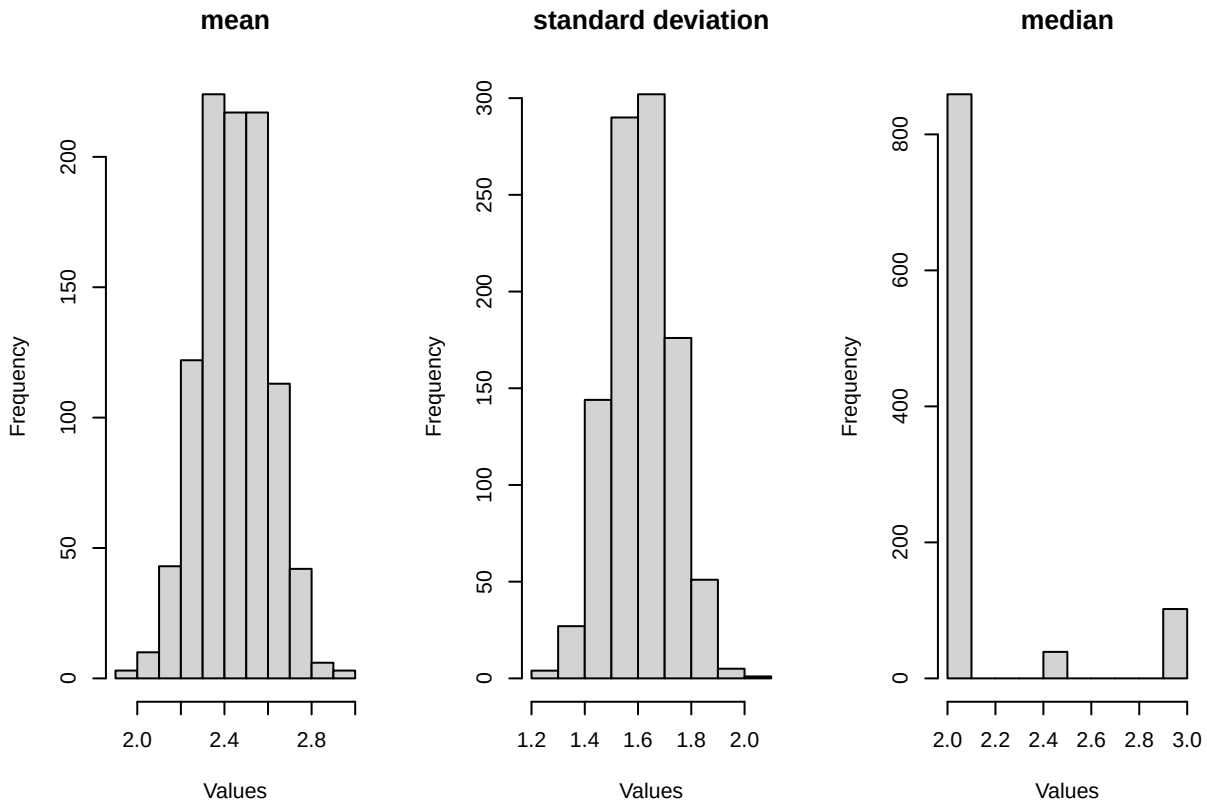



Repeat the previous process:

```
# samples
stats = matrix(NA,nrow = 1000,ncol = 3)

for (i in 1:1000){
  s = sample(rv_2,100) # sample size = 100
  stats[i,] = c(mean(s),sd(s),median(s))
}

par(mfrow=c(1,3))
hist(stats[,1],main="mean",xlab="Values")
hist(stats[,2],main="standard deviation",xlab="Values")
hist(stats[,3],main="median",xlab="Values")
```




```
c(mean(stats[,1]),mean(stats[,2]),mean(stats[,3]))
```

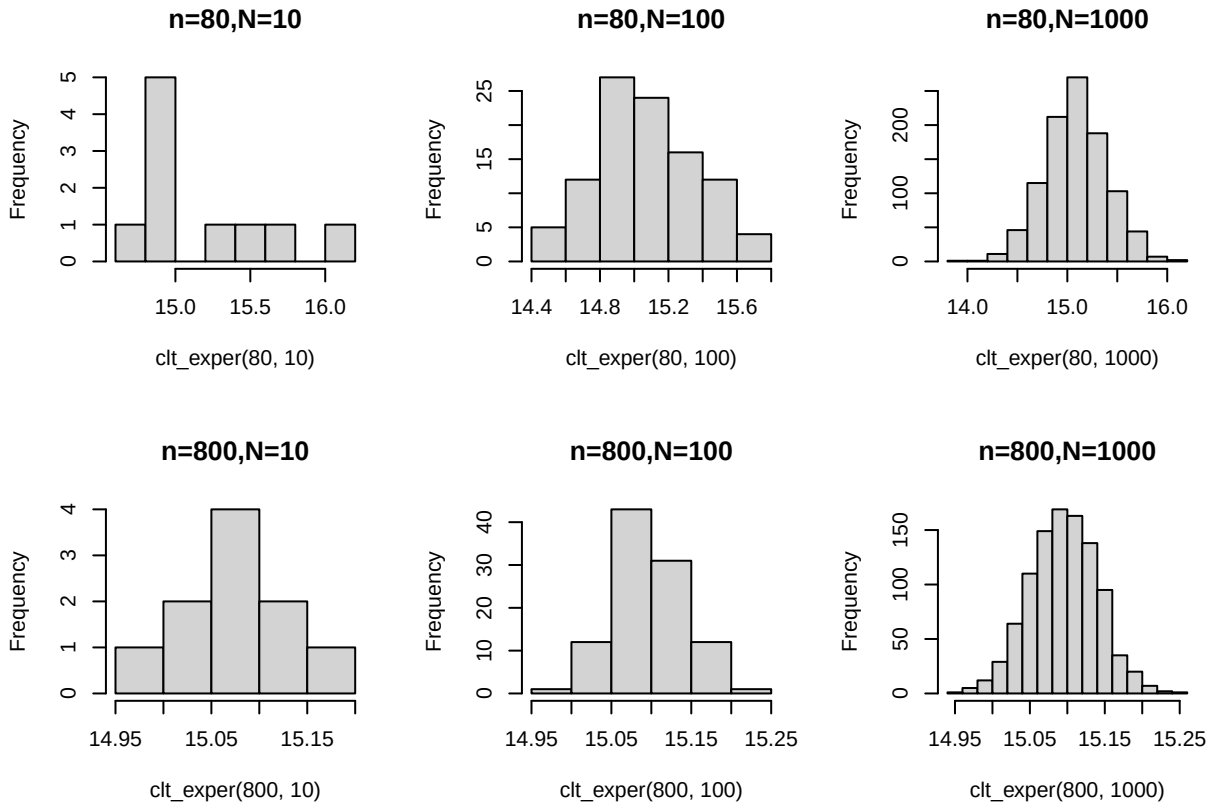
```
## [1] 2.451240 1.612488 2.121500
```

Sample size $n$ vs. sampling times $N$: $N \to +\infty$, $SMD \to \mathsf{Norm}(\mu, \sigma/\sqrt{n})$; $n$ approaches the population size, the *standard deviation* of the distribution approaches to 0. Why? **Central Limit Theorem (CLT)**: $\mu_s = \mu$, $\sigma_s = \frac{\sigma}{\sqrt{n}}$, where $\mu_s$ is the mean of the sampling distribution, $\sigma_s$ is the standard deviation of the sampling distribution.

```
# define a function and change n and N:
clt_exper = function(n,N){
  means = c()
for (i in 1:N){
  s = sample(rv_1,n)
  means = c(means,mean(s))
}
  return(means)
}

# plot the histograms
par(mfrow=c(2,3))
hist(clt_exper(80,10),main = "n=80,N=10")
hist(clt_exper(80,100),main = "n=80,N=100")
hist(clt_exper(80,1000),main = "n=80,N=1000")
hist(clt_exper(800,10),main = "n=800,N=10")
```

```
hist(clt_exper(800,100),main = "n=800,N=100")
hist(clt_exper(800,1000),main = "n=800,N=1000")
```




As you can see, large sampling times $N$ leads to more "normal" distributions, while large sample size $n$ leads to smaller variance of the distributions.

```
c(mean(clt_exper(80,1000)),mean(clt_exper(800,1000)))
```

```
## [1] 15.09292 15.09384
```

```
c(sd(clt_exper(80,1000)),sd(clt_exper(800,1000)))
```

```
## [1] 0.32126187 0.04654973
```

The increase of sample size didn't change the means, however, it decreased the *standard deviation* from 0.32 to 0.05

**4. What are the relationships between standard errors, sampling errors, confidence interval, and confidence level?**

The difference between the sample statistic and the population parameter is considered the **sampling error**. In a sampling distribution of means, where $\mu_s \to \mu$, when $\sigma_s \to 0$ $(n \to N)$.

```r
sdm = clt_exper(800,1000)
ms = mean(sdm)
quantile(sdm,probs=c(0.025,0.975))
```

```
##     2.5%    97.5%
## 15.00263 15.18365
```

That means 95% of the sample means are within [14.95,15.14]. Given the sample distribution is a Normal distribution, 95% of the sample means should be around $\mu$: $[\mu-1.96\sigma, \mu+1.96\sigma] \approx [\mu_s-1.96\sigma_s, \mu_s+1.96\sigma_s]$. The standard deviation of the sampling distribution ($\sigma_s$) is called **standard error** ($SE$).

```r
ms-1.96*sd(sdm)
```

```
## [1] 15.00266
```

```r
ms+1.96*sd(sdm)
```

```
## [1] 15.18643
```

In practice, it is nearly impossible to observe the sampling distribution directly, therefore, we don't know the *mean* of the sampling distribution ($\mu_s$), the *standard deviation* of sampling distribution ($\sigma_s$), population *mean* ($\mu$), or population *standard deviation* ($\sigma$). What we know is the *mean* of a sample ($\overline{x}$), and the *standard deviation* of a sample ($s$).

For example, in sample 1 (s_1), the *mean* is 15.39, *standard deviation* is 2.85. We know both are different from the population *mean* and *standard deviation* (15.09, 2.94). But if we say the *mean* of any sample 15.39 is the true *mean*, how likely we are wrong? We should based on the **CLT**, imaging the sampling distribution, and then we have ($SE = \sigma_s = \frac{\sigma}{\sqrt{n}}$):

$$CI_{1-\alpha} = \overline{x} \pm Z_{\alpha/2} \cdot SE = \overline{x} \pm Z_{\alpha/2}\frac{\sigma}{\sqrt{n}}$$

We usually call $\pm Z_{\alpha/2} \cdot SE$ **sampling error**. If $\alpha = 5\%, 1-\alpha = 95\%, Z_{\alpha/2} \approx 1.96, \sigma \approx s$ (it is just convenient to use $s$ to replace $\sigma$), we will have the **95% confidence interval**:

$$CI_{95\%} = \overline{x} \pm 1.96\frac{s}{\sqrt{n}}$$

```r
m = mean(s_1)
s = sd(s_1)
m-1.96*s/sqrt(100)
```

```
## [1] 14.8271
```

```r
m+1.96*s/sqrt(100)
```

```
## [1] 15.94404
```

You will find that the true mean 15.09 is within the 95% confidence interval [14.83,15.94]. Is it possible that for some ((random) samples, the true mean is not in the calculated confidence interval? The answer is yes and we even know the probability IS *5%*.

**5. What are the differences among T-test, Z-test, F-test, $\chi^2$-test etc.?**

A test statistic ($TS$) is a measure of the difference between the observed data and what we expected from the null hypothesis (by chance). The test statistic gets bigger (in absolute value) as the observed data looks unusual compared the null hypothesis. So a large test statistics cast doubt on the null hypothesis. There are many different kinds of tests.

$$TS = \frac{Observedvalue - Expectedvalue}{SE}$$

If $TS > 1.96 => p < .05$; $TS > 2.58 => p < .01$ Why? For example, in Z-test, $Z = \frac{\overline{x}-u_0}{SE} \sim \mathsf{Norm}(0,1)$. Therefore, $Z = \frac{\overline{x}-u_0}{SE} > 1.96$. And then, $(\overline{x}-u_0) > 1.96 \cdot SE => \overline{x} > u_0 + 1.96 \cdot SE$. In a Normal distribution, only 2.5% values could be larger than $1.96 \cdot SE$. So, it is very likely that $\overline{x} > u_0$. In a Z-test, $SE = s/\sqrt{n}$.

What are the differences? The distributions of the test statistics. When the sample size $n$ is relatively small, $T = \frac{\overline{x}-u_0}{SE} \sim \mathsf{StudentT}(\nu, 0, 1)$, where $\nu = n-1$ is the degree of freedom: $n \to +\infty$, then $\mathsf{StudenT} \to \mathsf{Norm}$

```
# whether mu>14
t.test(s_1,mu=14,alternative="less")
```

```
##
##  One Sample t-test
##
## data:  s_1
## t = 4.8628, df = 99, p-value = 1
## alternative hypothesis: true mean is less than 14
## 95 percent confidence interval:
##      -Inf 15.85867
## sample estimates:
## mean of x
##  15.38557
```

```
# whether the two samples have equal means:
t.test(s_1,s_2)
```

```
##
##  Welch Two Sample t-test
##
## data:  s_1 and s_2
## t = 2.7353, df = 195.59, p-value = 0.006804
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.2917745 1.7997456
## sample estimates:
## mean of x mean of y
##  15.38557  14.33981
```

```
# we don't assume equal length of the two vectors:
t.test(s_1,rv_1)
```

```
##
##  Welch Two Sample t-test
##
```

```
## data:  s_1 and rv_1
## t = 0.97123, df = 121.02, p-value = 0.3334
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.3022431  0.8843758
## sample estimates:
## mean of x mean of y
##  15.38557  15.09450
```

Before we perform a T/Z-test, we need to check assumptions:

- measurement scales: ratio or interval
- simple random sample
- normality (the distribution of sample means!)
- large sample size
- homogeneity of variance (equal variance)

Why does it require a Normal distribution? The t-test is invalid for small samples from non-normal distributions, but it is valid for large samples from non-normal distributions (see online answers).

Let's try Poisson distribution:

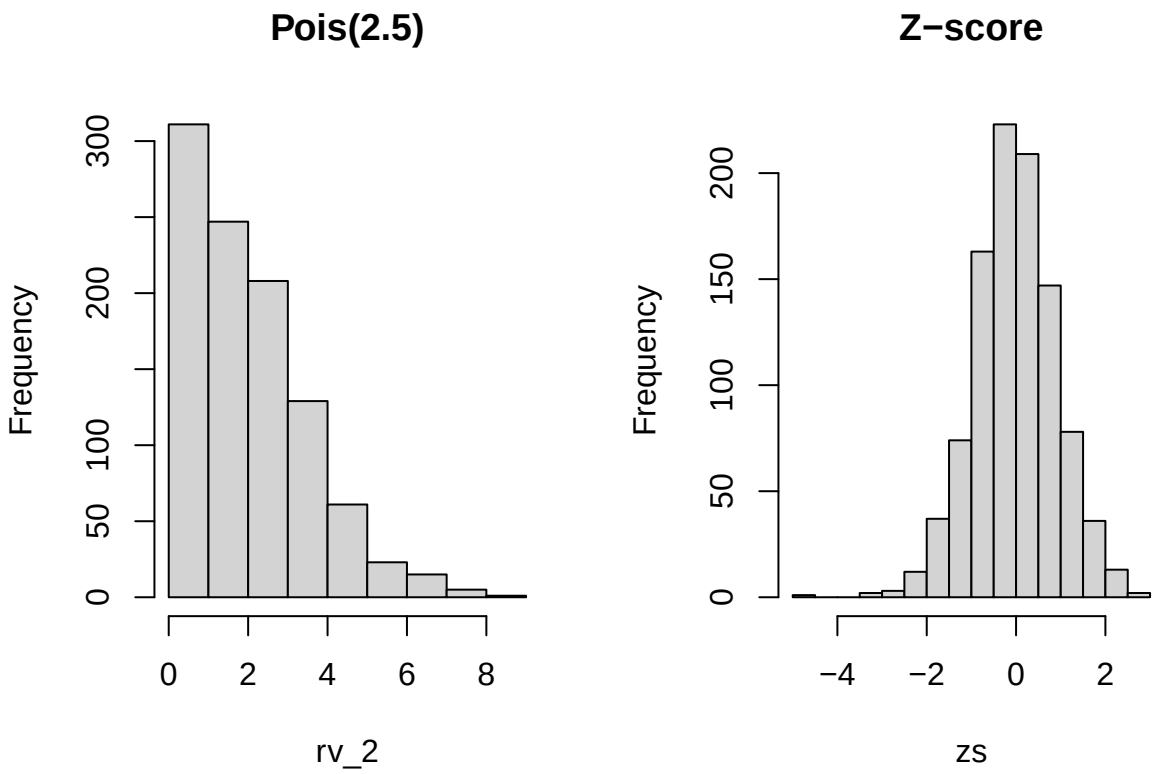
- generate a variable with a Poisson distribution
- draw a random sample (sp)
- calculate the *mean* difference between the sample and the population: `mean(sp)-mean(rv_2)`
- standard error is the population *sd* divided by the square root of sample size
- get the Z score

```
zs = c()

for (i in 1:1000){
  sp = sample(rv_2,100) # rv_2 here is a poisson distribution we generated before
  m = mean(sp)-mean(rv_2)
  se = sd(sp)/sqrt(100)
  z = m/se

  zs= c(zs,z)
}

par(mfrow=c(1,2))
hist(rv_2, main="Pois(2.5)")
hist(zs,main = "Z-score")
```




The above example shows that the Z-score of a skewed variable is also normally distributed.

Why does it require equal variance (only for two-sample tests)? $T = \frac{\overline{x}-u_0}{SE} \sim \mathsf{StudentT}(\nu, 0, 1)$, where $SE = s/\sqrt{(n)}$. We extend this to two-sample-mean comparison: $T = \frac{\overline{x}_1-\overline{x}_2}{SE} \sim \mathsf{StudentT}(\nu, 0, 1)$. It is easy to calculate the sample *means* ($\overline{x}_1$ and $\overline{x}_2$) and sample *standard deviations* ($s_1$ and $s_2$). The problem is how to calculate $SE$. If the two samples are independent to each other, according to the **Variance Sum Law**:

$$\sigma^2_{x1\pm x2} = \sigma^2_{x1} + \sigma^2_{x2}$$

Therefore, $SE_{x1-x2} = \sqrt{s_1^2/n_1 + s_2^2/n_2}$. If equal variances are assumed, then $SE_{x1-x2} = \sqrt{\frac{(n_1-1)s_1^2+(n_2-1)s_2^2}{n_1+n_2-2}}$ (pooled variance estimate). The following example tests two samples from different distributions (different *means* and *variances* too).

```
zs_un = c() # z scores unequal variance
zs_eq = c() # z scores equal variance

for (i in 1:1000){
  sp_1 = sample(rv_1,100) # rv_1 normal
  sp_2 = sample(rv_2,100) # rv_2 poisson

  # mean difference
  m = mean(sp_1)-mean(sp_2)

  # sample sds
  se_1 = sd(sp_1)
```

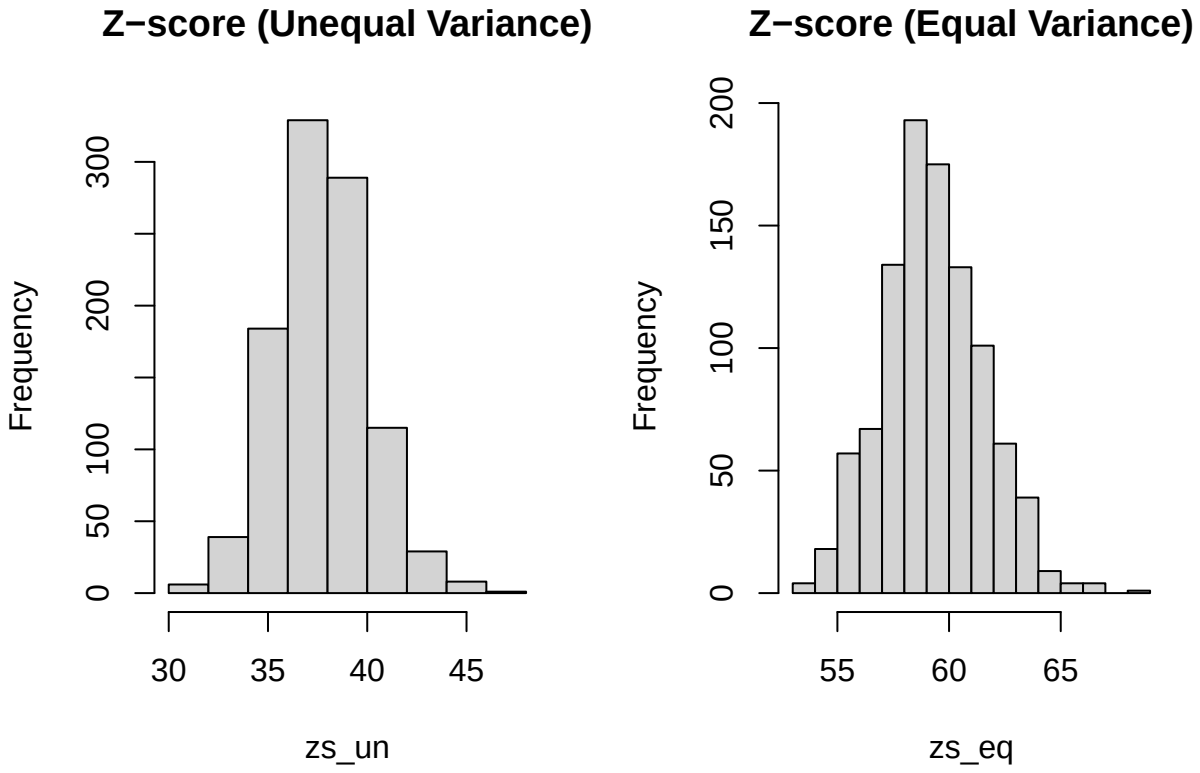
```
  se_2 = sd(sp_2)

  # unequal var.
  se_un = sqrt(se_1^2/100+se_2^2/100)
  # equal var.
  se_eq = sqrt(99*(se_1+se_2)/198)

  z_un = m/se_un
  z_eq = m/(se_eq*sqrt(2/100))

  zs_un = c(zs_un,z_un)
  zs_eq = c(zs_eq,z_eq)
}

par(mfrow=c(1,2))
hist(zs_un,main="Z-score (Unequal Variance)")
hist(zs_eq,main="Z-score (Equal Variance)")
```




```
c(mean(zs_un),mean(zs_eq))
```

```
## [1] 37.73098 59.27537
```

Both scores followed the Normal distribution, while it appears the equal variance method tends to overestimate the significance than the unequal variance method, because `mean(zs_eq) > mean(zs_un)`.

Based on "extensive" simulations from distributions either meeting or not meeting the assumptions imposed by a t-test, (normality and homogeneity of variance) that the Welch-tests performs equally well when the assumptions are met (i.e., basically same probability of committing alpha and beta errors) but outperforms the t-test if the assumptions are not met, especially in terms of power. Therefore, they recommend to always use the Welch-test if the sample size exceeds 30.

```
t.test(sp_1,sp_2,var.equal = T)
```

```
##
##  Two Sample t-test
##
## data:  sp_1 and sp_2
## t = 33.456, df = 198, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  11.57019 13.01962
## sample estimates:
## mean of x mean of y
##   14.7749    2.4800
```

```
t.test(sp_1,sp_2,var.equal = F)
```

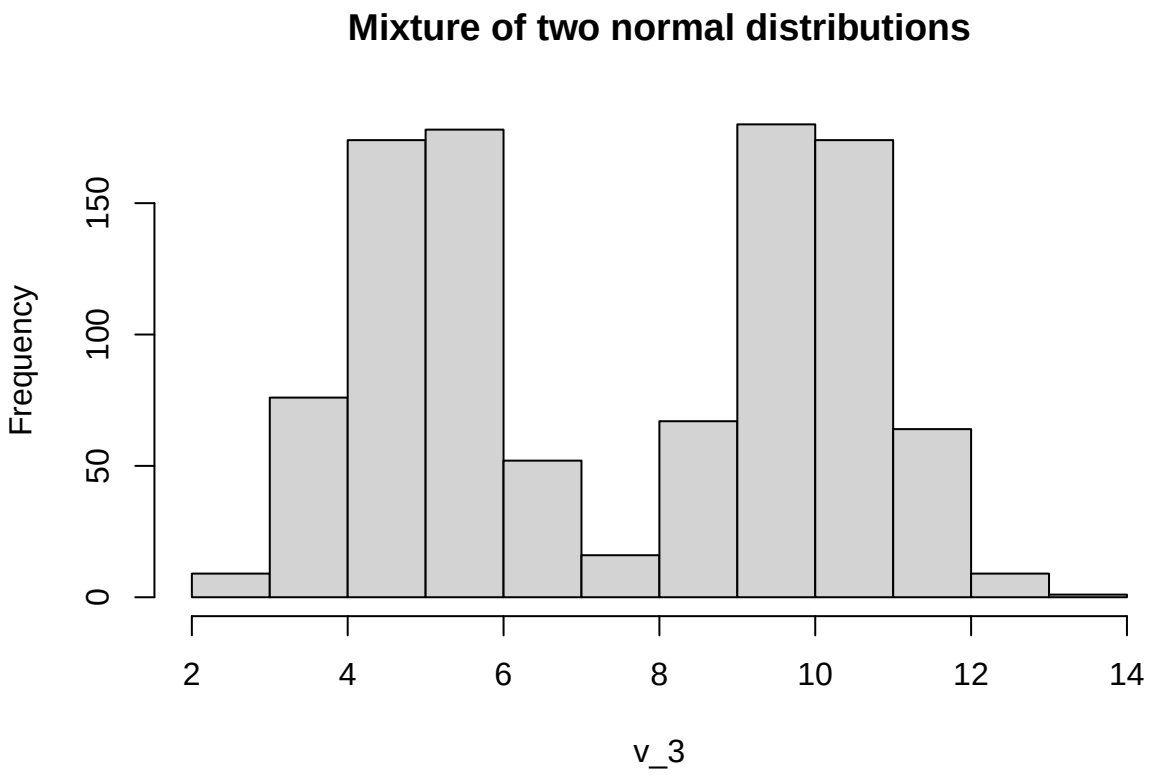
```
##
##  Welch Two Sample t-test
##
## data:  sp_1 and sp_2
## t = 33.456, df = 147.04, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  11.56864 13.02117
## sample estimates:
## mean of x mean of y
##   14.7749    2.4800
```

How about the two samples are dependent? Paired T-test. If more than two samples, use F-test in ANOVA (equal or unequal variances?). If both categorical variables, then $TS \to \chi^2(k)$, where $k$ is the degree of freedom.

**6. What are distribution-free tests (non-parametric tests)? We test the difference between means, can we test the difference between medians/variances? Can we test the difference between two categorical variables?**

The problem that we cannot always use T-test is not because of its non-normality or unequal variances but *mean* is not well defined. Furthermore, for some test statistics (e.g., *mean difference*), we know they follow certain types of distributions. However, for others, we don't know. The distribution in distribution-free does not mean the distribution of the variables but the distribution of the test-statistics. The below distribution is a mixture of two Normal distributions. The *mean* is not the most important characteristic.

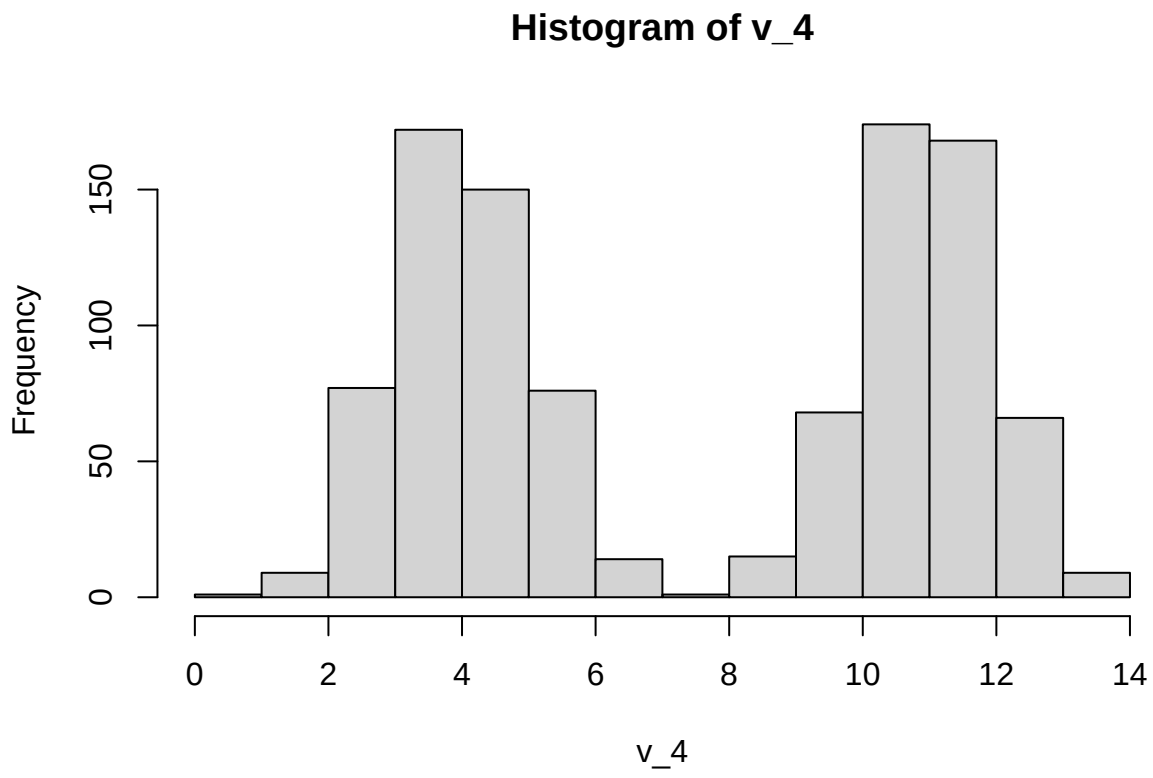
```
v_1 = rnorm(500,5,1)
v_2 = rnorm(500,10,1)
v_3 = c(v_1,v_2)
hist(v_3,main="Mixture of two normal distributions")
```




```
mean(v_3)
```

```
## [1] 7.480514
```

```
# we define another variable in similar way:
v_4 = c(rnorm(500,4,1),rnorm(500,11,1))
hist(v_4)
```




```
mean(v_4)
```

```
## [1] 7.478289
```

As expected, there is no difference in *means*.

```
t.test(v_3,v_4) # no difference of mean
```

```
##
##  Welch Two Sample t-test
##
## data:  v_3 and v_4
## t = 0.015516, df = 1844.9, p-value = 0.9876
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.2790143  0.2834643
## sample estimates:
## mean of x mean of y
##  7.480514  7.478289
```

How about quantiles? Are they different?

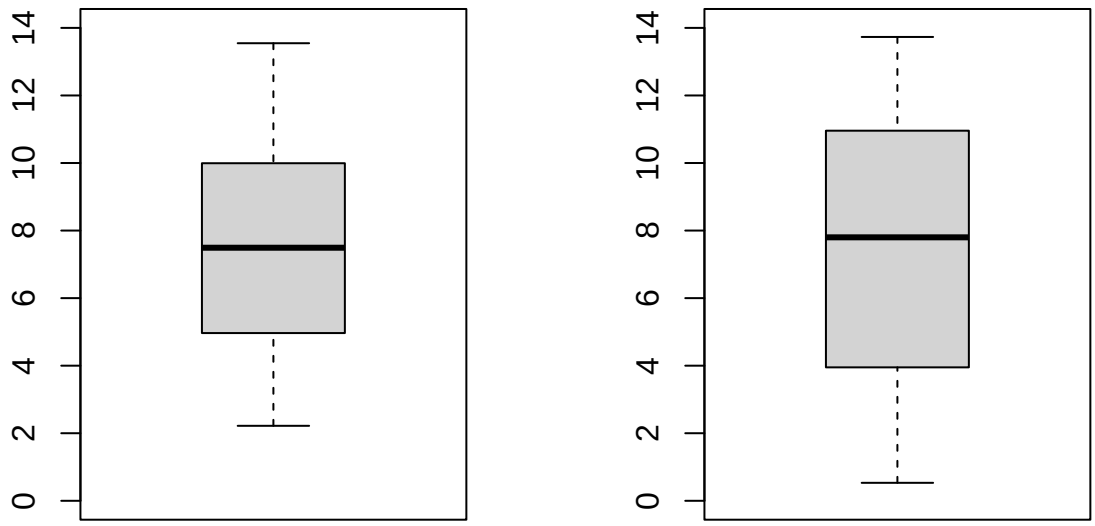
```
quantile(v_3)
```

```
##        0%       25%       50%       75%      100%
##  2.219744  4.965960  7.491794  9.993150 13.546034
```

```
quantile(v_4)
```

```
##         0%        25%        50%        75%       100%
##  0.5319756  3.9506611  7.7999385 10.9566565 13.7315041
```

```
par(mfrow=c(1,2))
boxplot(v_3,ylim=c(0,14))
boxplot(v_4,ylim=c(0,14))
```




Yes, quantiles are informative. And the boxplots also suggest that the variances are different. However, what are the standard errors ($SEs$) of quantiles (or other quantities if defined)? Even if we know the $SE$, it remains unknown about the distribution of the test statistics. An easy way is to to perform the randomization test, which is one of the most commonly used non-parametric tests.

The below code shows an example of testing variance difference using randomization test (use `var.test` for a parametric test):

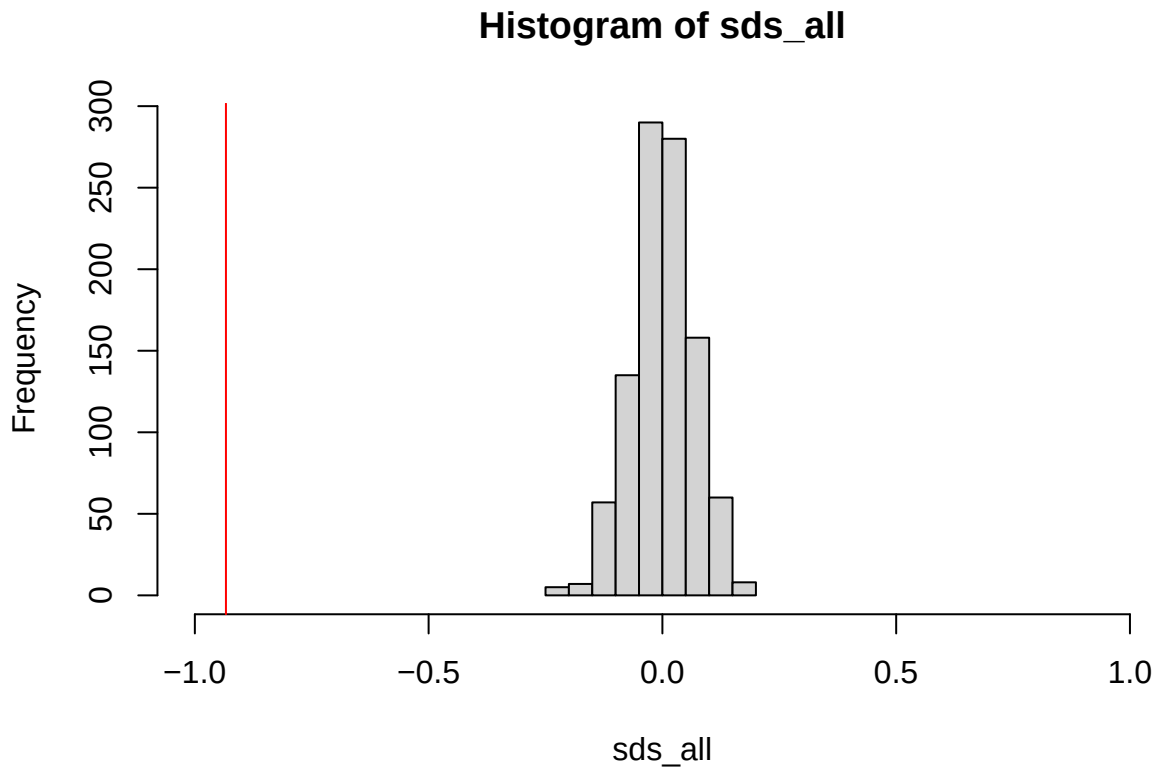
1. given any two variables: x1, x2
2. calculate the difference of standard deviation: `sd(x1)-sd(x2)`

3. Permutation
    - combine two variables to a single one: values = c(x1,x2)
    - randomly permute the values
    - split the permuted values into two groups and calculate the difference again
    - repeat many time
4. compare the real difference with the differences after permutation

```
sts = sd(v_3)-sd(v_4)
sds_all = c()

for (i in 1:1000){ #repeat many times
  values = c(v_3,v_4) # combine all values
  pv = sample(values) # randomly permute the values
  sds = sd(pv[1:500])-sd(pv[501:1000]) # split into two groups and calculate the difference of sds
  sds_all = c(sds_all,sds)
}

hist(sds_all,xlim=c(-1,1))
abline(v=sts,col="red")
```




```
table(sts<sds_all)/1000 # probability of true!
```

```
##
## TRUE
##    1
```

It means that the observed difference is smaller than 100% of the permuted differences. In other words, it is very unlikely the observed variance difference is caused by chance.

Another method is to use bootstrapping:

1. given two variables: x1, x2
2. calculate the difference: `sd(x1)-sd(x2)`
3. bootstrap:
    - resample from x1 and x2
    - calculate the difference in the subsamples
    - repeat many times
4. compare the real difference with the differences after permutation

We don't need to do step 3 from the scratch because we can use the `boot` package in r.

```
library(boot)
data = data.frame(v_3,v_4)
sts = function(d,i){
  d2=d[i,]
  dis = sd(d2$v_3)-sd(d2$v_4)
  return(dis)
}
boot_sd = boot(data,sts,R=1000) # define the data, define the test statistic, repetition times
boot_sd
```

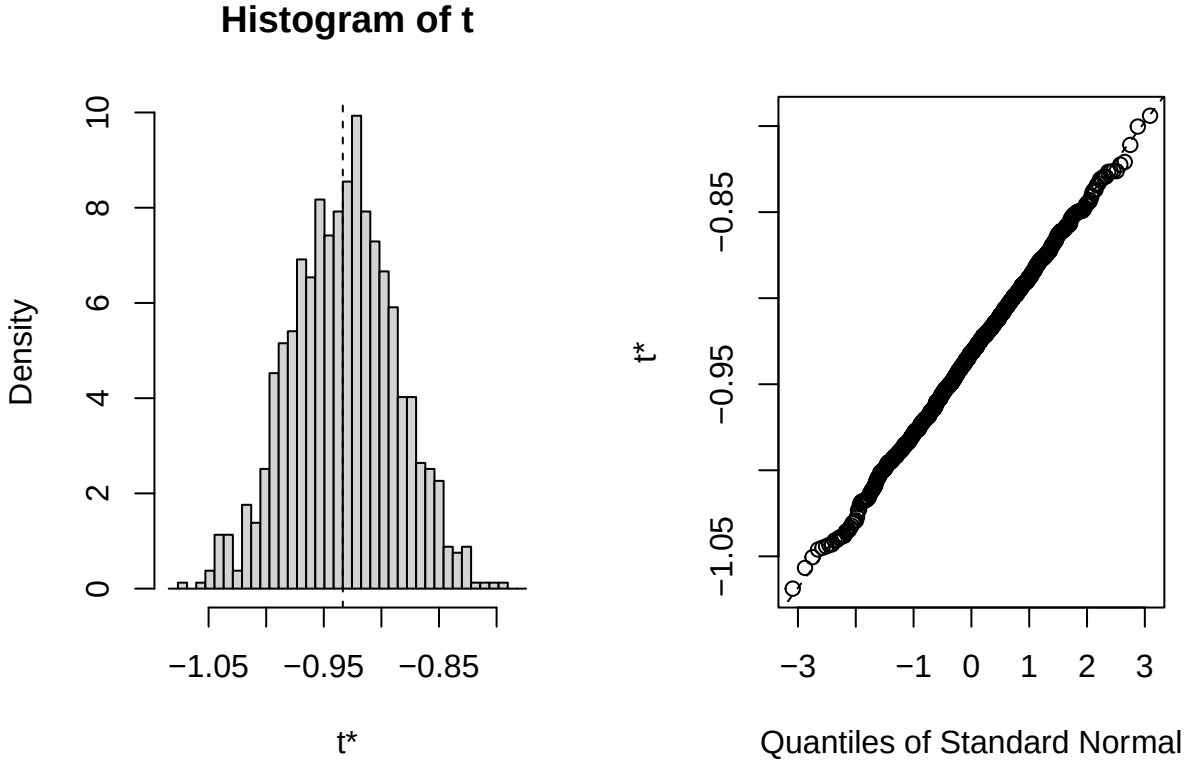
```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = data, statistic = sts, R = 1000)
##
##
## Bootstrap Statistics :
##       original       bias    std. error
## t1* -0.9334881 0.0005741538  0.04512716
```

```
boot.ci(boot.out = boot_sd, type = c("norm", "basic", "perc", "bca"))
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot_sd, type = c("norm", "basic", "perc",
##     "bca"))
```

```
##
## Intervals :
## Level      Normal              Basic
## 95%   (-1.0225, -0.8456 )   (-1.0196, -0.8438 )
##
## Level     Percentile            BCa
## 95%   (-1.0232, -0.8473 )   (-1.0302, -0.8493 )
## Calculations and Intervals on Original Scale
```

```
plot(boot_sd)
```




**Randomization tests** take the set of scores, randomize their ordering, and compute statistics from the results. **Permutation tests** do the same thing, but I reserve that label for tests in which we take all possible permutations of the data, rather than a subset or rearrangements. **Bootstrapping** resamples with replacement from a set of data and computes a statistic (such as the mean or median) on each resampled set. Bootstrapping is used primarily for parameter estimation. Bootstrapping is primarily focused on estimating population parameters, and it attempts to draw inferences about the population(s) from which the data came. Randomization approaches, on the other hand, are not particularly concerned about populations and/or their parameters. Instead, randomization procedures focus on the underlying mechanism that led to the data being distributed between groups in the way that they are (click here for a more detailed introduction).

Randomization Tests (Contingency Tables): (Fisher's Exact Test). Fisher's exact test is a statistical test used to determine if there are nonrandom associations between two categorical variables.

```r
tbl = matrix(data=c(65, 45, 20, 40), nrow=2, ncol=2, byrow=T)
dimnames(tbl) = list(City=c('B', 'T'), Gender=c('M', 'F'))
tbl
```

```
##     Gender
## City  M  F
##    B 65 45
##    T 20 40
```

```r
chisq.test(tbl)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  tbl
## X-squared = 9.2985, df = 1, p-value = 0.002293
```

```r
fisher.test(tbl)
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  tbl
## p-value = 0.002162
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##  1.427537 5.912307
## sample estimates:
## odds ratio
##   2.870314
```

Odds ration: $OR = \frac{65/20}{45/40} = \frac{65\times 40}{45\times 20}$. The odds ratio shows how many times more positive cases (Male in city B + Female NOT in B) occur than negative cases (Male NOT in B + Female in B).

*Bootstrapping for correlation coefficients.* Recall that we don't assume normality of variables to calculate correlation coefficients (particularly Pearson's $r$). However, it might influence the significance test (or estimation of confidence interval).

```r
# generate two highly skewed variables
r1 = rweibull(1000,0.5)
r2 = 0.5*r1 + rweibull(1000,0.5)
data = data.frame(r1,r2)

# calculate the correlation coefficients
cor.test(r1,r2)
```

```
##
##  Pearson's product-moment correlation
##
## data:  r1 and r2
## t = 17.317, df = 998, p-value < 2.2e-16
```

```
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.4315397 0.5269640
## sample estimates:
##       cor
## 0.4806736
```

```
cor.test(rank(r1),rank(r2))
```

```
##
##  Pearson's product-moment correlation
##
## data:  rank(r1) and rank(r2)
## t = 19.504, df = 998, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.4789404 0.5688044
## sample estimates:
##       cor
## 0.5253357
```

Please note that the confidence interval of the rank correlation is more narrow than that of Pearson's correlation. Alternatively, we test the significance of the coefficients using bootstrapping (without assuming that how the sampling $r$s distributed).

```
# define the statistics:
rs_1 = function(d,i){
  d2=d[i,]
  r = cor(d2$r1,d2$r2,method="pearson")
  return(r)
}

rs_2 = function(d,i){
  d2=d[i,]
  r = cor(d2$r1,d2$r2,method="spearman")
  return(r)
}

# run bootstrapping
boot_r1 = boot(data,rs_1,R=1000) # define the data, define the test statistic, repetition times
boot_r2 = boot(data,rs_2,R=1000) # define the data, define the test statistic, repetition times

# obtain confidence interval
boot.ci(boot.out = boot_r1, type = c("norm", "basic", "perc", "bca"))
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot_r1, type = c("norm", "basic", "perc",
##     "bca"))
##
## Intervals :
```

```
## Level      Normal              Basic
## 95%   ( 0.3672,  0.5956 )   ( 0.3664,  0.5987 )
##
## Level     Percentile            BCa
## 95%   ( 0.3626,  0.5950 )   ( 0.3636,  0.5964 )
## Calculations and Intervals on Original Scale
```

```
boot.ci(boot.out = boot_r2, type = c("norm", "basic", "perc", "bca"))
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot_r2, type = c("norm", "basic", "perc",
##     "bca"))
##
## Intervals :
## Level      Normal              Basic
## 95%   ( 0.4724,  0.5787 )   ( 0.4716,  0.5770 )
##
## Level     Percentile            BCa
## 95%   ( 0.4737,  0.5791 )   ( 0.4737,  0.5791 )
## Calculations and Intervals on Original Scale
```

## III. Regression (OLS)

### 7. Can regression models (why plural?) replace all the above-mentioned tests in section II?

The commonly used Ordinary Least Square (OLS) regression is just one of the many regression models, as we will explain the details in section IV. But as we have already known that Logistic Regression is different from OLS regression.

In section II, we introduced several statistics, basically comparing the difference of *means*: Z and T for two groups and F for more than two groups. In a regression framework, the dependent variable is a continuous variable, while the independent variable is a categorical variable.

Compare `t.test` with `lm`:

```
head(sleep) # load the sleep data
```

```
##   extra group ID
## 1   0.7     1  1
## 2  -1.6     1  2
## 3  -0.2     1  3
## 4  -1.2     1  4
## 5  -0.1     1  5
## 6   3.4     1  6
```

```
t.test(extra~group,data=sleep,var.equal = T) # assume equal variance
```

```
##
##  Two Sample t-test
##
## data:  extra by group
## t = -1.8608, df = 18, p-value = 0.07919
## alternative hypothesis: true difference in means between group 1 and group 2 is not equal to 0
## 95 percent confidence interval:
##  -3.363874  0.203874
## sample estimates:
## mean in group 1 mean in group 2
##            0.75            2.33
```

```
summary(lm(extra~group,data=sleep))
```

```
##
## Call:
## lm(formula = extra ~ group, data = sleep)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -2.430 -1.305 -0.580  1.455  3.170
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.7500     0.6004   1.249   0.2276
## group2        1.5800     0.8491   1.861   0.0792 .
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.899 on 18 degrees of freedom
## Multiple R-squared:  0.1613, Adjusted R-squared:  0.1147
## F-statistic: 3.463 on 1 and 18 DF,  p-value: 0.07919
```

Note that both $t$ and $p$ values are similar. In addition, the mean difference 2.33-0.75 = 1.58, which is the regression coefficient.

How about the situation with more than 2 groups (using F-test in `avo`)?

```
head(npk)
```

```
##   block N P K yield
## 1     1 0 1 1  49.5
## 2     1 1 1 0  62.8
## 3     1 0 0 0  46.8
## 4     1 1 0 1  57.0
## 5     2 1 0 0  59.8
## 6     2 1 1 1  58.5
```

```
aggregate(yield~block,npk,mean)
```

```
##   block  yield
## 1     1 54.025
## 2     2 57.450
## 3     3 60.775
## 4     4 50.125
## 5     5 50.525
## 6     6 56.350
```

```
summary(aov(yield~block,data=npk))
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## block        5  343.3   68.66   2.318 0.0861 .
## Residuals   18  533.1   29.61
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(lm(yield~block,data=npk))
```

```
##
## Call:
## lm(formula = yield ~ block, data = npk)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.2250 -3.4937 -0.5375  2.1062 11.8750
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)   54.025      2.721  19.855 1.09e-13 ***
## block2         3.425      3.848   0.890   0.3852
## block3         6.750      3.848   1.754   0.0964 .
## block4        -3.900      3.848  -1.013   0.3243
## block5        -3.500      3.848  -0.910   0.3751
## block6         2.325      3.848   0.604   0.5532
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.442 on 18 degrees of freedom
## Multiple R-squared:  0.3917, Adjusted R-squared:  0.2228
## F-statistic: 2.318 on 5 and 18 DF,  p-value: 0.08607
```

This is straightforward because `lm` actually reported F-test statistics as well. They are exactly the same thing!

How about non-normal distribution?

```
# generate data
g1 = rpois(500,2.5)
g2 = rpois(500,3.5)
dv = c(g1,g2)
groups = c(rep(1,500),rep(0,500))
d = data.frame(dv,groups)

# compare the two outputs
t.test(dv~groups,d)
```

```
##
##  Welch Two Sample t-test
##
## data:  dv by groups
## t = 8.5826, df = 959.72, p-value < 2.2e-16
## alternative hypothesis: true difference in means between group 0 and group 1 is not equal to 0
## 95 percent confidence interval:
##  0.7266082 1.1573918
## sample estimates:
## mean in group 0 mean in group 1
##           3.510           2.568
```

```
summary(lm(dv~groups,d))
```

```
##
## Call:
## lm(formula = dv ~ groups, data = d)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.510 -1.510 -0.510  1.432  6.490
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.51000    0.07761  45.226   <2e-16 ***
```

```
## groups      -0.94200    0.10976  -8.583   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.735 on 998 degrees of freedom
## Multiple R-squared:  0.06874,    Adjusted R-squared:  0.0678
## F-statistic: 73.66 on 1 and 998 DF,  p-value: < 2.2e-16
```

Testing for categorical variables (contingency tables) requires a regression model that we are less familiar with. We create a contingency table with two variables: city and gender. To test the dependence between the two, we use either $\chi^2$-test (parametric) or Fisher's exact test (non-parametric).

```
tbl = matrix(data=c(65, 45, 20, 40), nrow=2, ncol=2, byrow=T)
dimnames(tbl) = list(City=c('B', 'T'), Gender=c('M', 'F'))
tbl
```

```
##     Gender
## City  M  F
##    B 65 45
##    T 20 40
```

```
chisq.test(tbl)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  tbl
## X-squared = 9.2985, df = 1, p-value = 0.002293
```

```
fisher.test(tbl)
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  tbl
## p-value = 0.002162
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##  1.427537 5.912307
## sample estimates:
## odds ratio
##   2.870314
```

Since this is just a 2 by 2 table, theoretically speaking, we can use logistic regression if we know the pre-aggregated data. We consider the frequencies in the contingency table the dependent variable and the city and gender as independent variables. Given the dependent variable is a count variable, as we will explain in section IV, we use a poisson regression to fit the data.

```
## log linear model
ctbl = data.frame(freq = c(65, 45, 20, 40), city = c("B","B","T","T"), gender = c("M","F","M","F"))
summary(logLM <- glm(freq ~city*gender, family=poisson(link="log"), data=ctbl))
```

```
##
## Call:
## glm(formula = freq ~ city * gender, family = poisson(link = "log"),
##     data = ctbl)
##
## Deviance Residuals:
## [1]  0  0  0  0
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)     3.8067     0.1491  25.536  < 2e-16 ***
## cityT          -0.1178     0.2173  -0.542  0.58781
## genderM         0.3677     0.1939   1.896  0.05793 .
## cityT:genderM  -1.0609     0.3356  -3.161  0.00157 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 2.5378e+01  on 3  degrees of freedom
## Residual deviance: 8.8818e-16  on 0  degrees of freedom
## AIC: 30.036
##
## Number of Fisher Scoring iterations: 3
```

The interaction coefficient (cityT:genderM = -1.0609) indicate the dependence between gender and city. In fact, log-linear is an extension of the $\chi^2$-test.

**8. What is the degree of freedom? Is it possible to fit a regression model with $R^2$=1? What is over-fitting? Why should we avoid over-fitting?**

Degrees of freedom ($df$) are the number of independent values that a statistical analysis can estimate. You can also think of it as the number of values that are free to vary as you estimate parameters. Typically, the degrees of freedom equals your sample size minus the number of parameters you need to calculate during an analysis ($N-P$). Degrees of freedom is a combination of how much data you have and how many parameters you need to estimate. It indicates how much independent information goes into a parameter estimate. In this vein, it's easy to see that you want a lot of information to go into parameter estimates to obtain more precise estimates and more powerful hypothesis tests. So, you want many $df$!

```
# generate a data sets with 3 IVs and y
x1 = rnorm(100,5,1)
x2 = rnorm(100,1,1)
x3 = rnorm(100,2,2)
y = 2*x1+1.5*x2+1.1*x3+rnorm(100) # sample size = 100
data = data.frame(y,x1,x2,x3)

# fit a linear model with 2 variables and find the df.
summary(lm(y~x1+x2,data = data)) # df = 100-3 (why 3 not 2?)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2, data = data)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.2391 -1.5136 -0.2256  1.5646  4.1626
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.7320     1.0860   0.674    0.502
## x1            2.2485     0.2088  10.768  < 2e-16 ***
## x2            1.4443     0.2046   7.059 2.52e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.202 on 97 degrees of freedom
## Multiple R-squared:  0.6188, Adjusted R-squared:  0.6109
## F-statistic: 78.72 on 2 and 97 DF,  p-value: < 2.2e-16
```

Please note that the degree of freedom is 97, instead of 100-2 = 98. For a typical OLS regression, an intercept is estimated by default. How about if we only have 3 observations? Theoretically speaking, $df$ = 3-3 = 0. As presented below, NAs were produced in `lm`. Actually, a model with $df = 0$ is a just identified model with unique solution and the $R^2$ should be 1.

```
summary(lm(y~x1+x2,data = data[sample(100,3),])) # 3 observations
```

```
##
## Call:
## lm(formula = y ~ x1 + x2, data = data[sample(100, 3), ])
##
## Residuals:
## ALL 3 residuals are 0: no residual degrees of freedom!
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   36.604        NaN     NaN      NaN
## x1            -8.486        NaN     NaN      NaN
## x2            11.872        NaN     NaN      NaN
##
## Residual standard error: NaN on 0 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:    NaN
## F-statistic:   NaN on 2 and 0 DF,  p-value: NA
```

```
summary(lm(y~x1+x2,data = data[sample(100,4),])) # 4 observations
```

```
##
## Call:
## lm(formula = y ~ x1 + x2, data = data[sample(100, 4), ])
##
## Residuals:
##     36     71     42     18
##  1.611 -1.815  2.382 -2.178
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)    5.008     12.838   0.390    0.763
## x1             1.685      3.240   0.520    0.695
## x2             1.053      1.670   0.631    0.642
##
## Residual standard error: 4.038 on 1 degrees of freedom
## Multiple R-squared:  0.3584, Adjusted R-squared:  -0.9249
## F-statistic: 0.2792 on 2 and 1 DF,  p-value: 0.801
```

As $df \to 0$, the $R^2 \to 1$, even though it is not significant. The `lm` function only works for $df > 0$. Nevertheless, we can solve the linear system mathematically when $df = 0$ (we cannot do that when $df < 0$).

```
library(matlib)

slice = data[sample(100,3),] # select the first 3 observations
A = as.matrix(slice[,2:3]) # IVs as matrix
b = slice$y # dependent variable
Solve(A, b) # solve the function
```

```
## x1    =   2.59411265
##   x2  =   0.98304612
##    0  =  -0.65586308
```

In statistics, over-fitting is "the production of an analysis that corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably". An over-fitted model is a statistical model that contains more parameters than can be justified by the data (i.e., small $df$). The essence of over-fitting is to have unknowingly extracted some of the residual variation (i.e., the noise) as if that variation represented underlying model structure.

How is this possible? In our case, we know how the data was generated according to $y = 2x_1 + 1.5x_2 + 1.1x_3 + \varepsilon$. If we measured all $x_i$, we will find a perfect solution. Let's include all three predictors:

```
summary(lm(y~x1+x2+x3,data))
```

```
##
## Call:
## lm(formula = y ~ x1 + x2 + x3, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.3056 -0.6612  0.0131  0.6583  2.5440
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.81844    0.49789  -1.644    0.103
## x1           2.14061    0.09465  22.616   <2e-16 ***
## x2           1.61339    0.09299  17.350   <2e-16 ***
## x3           1.10465    0.05684  19.434   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9965 on 96 degrees of freedom
## Multiple R-squared:  0.9227, Adjusted R-squared:  0.9203
## F-statistic: 382.2 on 3 and 96 DF,  p-value: < 2.2e-16
```

The $R^2 =$ is not 1, given the existence of $\varepsilon \sim \mathsf{Norm}(0, 1)$. Otherwise, it is a deterministic models (in contrast to probabilistic). Let's generate y2 without the random error ($\varepsilon$).

```
y2 = 2*x1+1.5*x2+1.1*x3
summary(lm(y2~x1+x2+x3))
```

```
## Warning in summary.lm(lm(y2 ~ x1 + x2 + x3)): essentially perfect fit: summary
## may be unreliable
```

```
##
## Call:
## lm(formula = y2 ~ x1 + x2 + x3)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -2.042e-14 -5.371e-16  3.071e-16  1.057e-15  6.078e-15
##
## Coefficients:
##               Estimate Std. Error    t value Pr(>|t|)
## (Intercept) -1.421e-14  1.435e-15 -9.900e+00 2.44e-16 ***
## x1           2.000e+00  2.729e-16  7.329e+15  < 2e-16 ***
## x2           1.500e+00  2.681e-16  5.595e+15  < 2e-16 ***
## x3           1.100e+00  1.639e-16  6.712e+15  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.873e-15 on 96 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 4.185e+31 on 3 and 96 DF,  p-value: < 2.2e-16
```

Now, the $R^2$ is 1. Is this just-identified model ($df = 0$), a perfect model (deterministic), or over-fitted model? Let's show another $R^2 = 1$ example that it is an over-fitted model.

```
# create a categorical variable with 50 levels in the 'data'
data$v1 = as.factor(c(1:50,1:50))
summary(aov(x3~v1,data))
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## v1          49  186.4   3.804   1.523  0.071 .
## Residuals   50  124.9   2.498
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The new variable is not associated with any $x_i$. Regress v1 on y:

```
# fit the first 50 observations
model1<-lm(y~v1,data[1:50,])
summary(model1)$r.squared
```

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

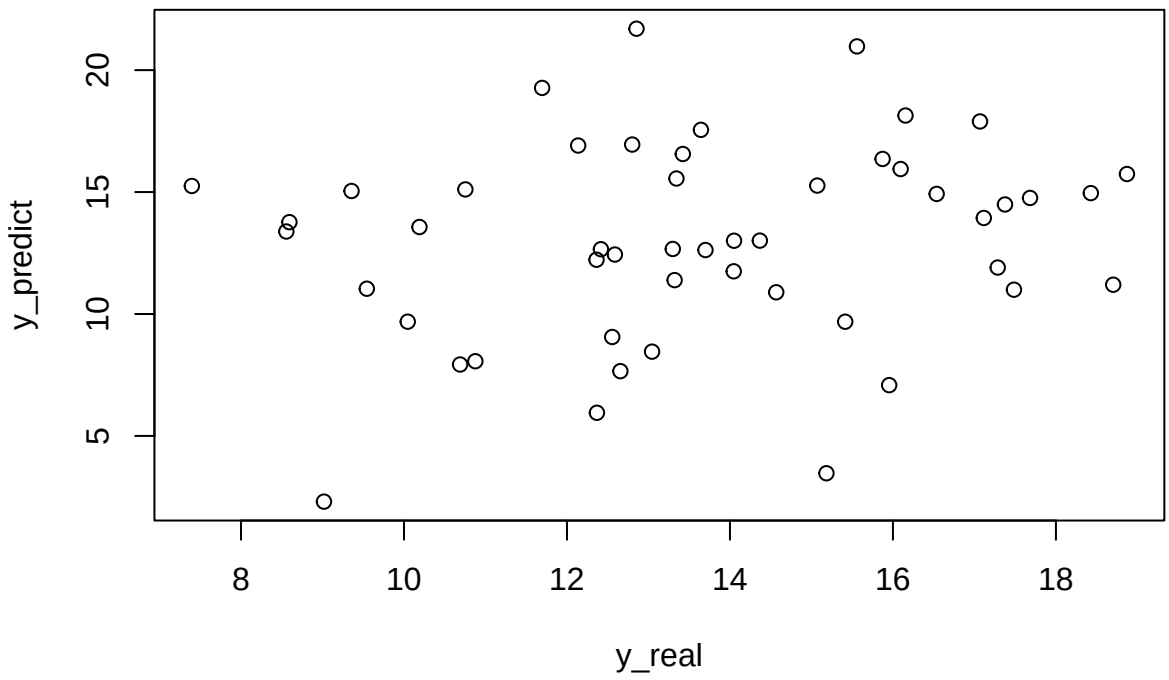
We can achieve $R^2 = 1$ without any information from $x_i$! Is this a good model? Let's use "model1" to predict the rest of 50 ys in the data set.

```
# predict using other 50 observations (xs)
y_predict = predict(model1,data[51:100,])

# compare predicted and true values
y_real = y[51:100]
cor(y_real,y_predict)^2 #correlation between predicted and true values
```

```
## [1] 0.03350821
```

```
plot(y_real,y_predict)
```




Even though model1 is a perfect model for the first 50 observations, the model is not applicable to the second 50 observations. In this case, we say model1 is an over-fitted model.

**9. How to interpret regression coefficients (direction, magnitude/strength, significance, form)? How to interpret when predictors are categorical? How to compare regression coefficients?**

```
# to generate a data set with both continuous and categorical variables
x1 = rnorm(1000,10,2)
x2 = rnorm(1000,7,3)+0.5*x1
```

```r
x3 = rbinom(1000,1,0.3)
x4 = sample(1:3, size = 1000, replace = TRUE, prob = c(0.2,0.5,0.3))

#dummy coding for x4
x41 = ifelse(x4==1,1,0)
x42 = ifelse(x4==2,1,0)
x43 = ifelse(x4==3,1,0)

# dependence between x and y
y = -2.2*x1+2.2*x2+3.3*x3+4*x42+8*x43+rnorm(1000)

# create the data frame
data = data.frame(y,x1,x2,x3,x4,x41,x42,x43)
```

Let's fit an OLS regression using `lm` and try to interpret the coefficients according to the direction (positive or negative), magnitude (weak or strong), form (linear or nonlinear), and significance (significant or not).

```r
summary(m <- lm(y~x1+x2+x3+factor(x4)))
```

```
##
## Call:
## lm(formula = y ~ x1 + x2 + x3 + factor(x4))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3584 -0.6282 -0.0032  0.6211  3.0564
##
## Coefficients:
##             Estimate Std. Error  t value Pr(>|t|)
## (Intercept) -0.11189    0.18107   -0.618    0.537
## x1          -2.20224    0.01643 -134.054   <2e-16 ***
## x2           2.21060    0.01044  211.646   <2e-16 ***
## x3           3.39508    0.06807   49.873   <2e-16 ***
## factor(x4)2  4.01667    0.08179   49.111   <2e-16 ***
## factor(x4)3  7.93990    0.08820   90.022   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9854 on 994 degrees of freedom
## Multiple R-squared:  0.9833, Adjusted R-squared:  0.9832
## F-statistic: 1.168e+04 on 5 and 994 DF,  p-value: < 2.2e-16
```

The coefficient for $x_3$ indicates the difference of $y$ between $x_3 = 1$ and $x_3 = 0$. Then, what does the coefficient for $x4 = 3$ mean? Does it indicate the difference between $x4 = 3$ and $x4 = 1 or 2$?

```r
#calculate the means for each category
aggregate(y~x4,data,mean)
```

```
##   x4        y
## 1  1 6.244224
## 2  2 9.432633
## 3  3 13.627507
```

Using standardized coefficients $\beta s$ for comparisons. It is easy to calculate $\beta s$ using the `lm.beta` function.

```
library(lm.beta)
betas = lm.beta(m)
betas
```

```
##
## Call:
## lm(formula = y ~ x1 + x2 + x3 + factor(x4))
##
## Standardized Coefficients::
## (Intercept)          x1          x2          x3 factor(x4)2 factor(x4)3
##   0.0000000  -0.5918520   0.9338013   0.2046776   0.2642496   0.4839834
```

We see that the absolute value of raw coefficient of x1 is close to that of x2, however, it is smaller in terms of the standardized coefficients. So, can we say $|\beta_{x_2}| - |\beta_{x_1}|$ and $x_2$ shows a greater impact on $y$? We still need a significance test, which requires the $SE$. The full version of the **Variance Sum Law**:

$$Var(c_1 b_1 + c_2 b_2) = c_1^2 Var(b_1) + c_2^2 Var(b_2) + 2c_1 c_2 Cov(b_1, b_2)$$

We can obtain the variance-covariance matrix using the `vcov` function in R:

```
vc = vcov(m) # variance-covariance matrix
vc
```

```
##               (Intercept)            x1            x2            x3
## (Intercept)  0.0327847624 -1.903702e-03 -7.208136e-04 -1.401625e-03
## x1          -0.0019037021  2.698803e-04 -6.277071e-05  5.997433e-06
## x2          -0.0007208136 -6.277071e-05  1.090930e-04  9.603204e-07
## x3          -0.0014016251  5.997433e-06  9.603204e-07  4.634131e-03
## factor(x4)2 -0.0044805144 -5.947661e-05  3.654123e-05 -8.958214e-05
## factor(x4)3 -0.0050139671 -1.744380e-06  3.223652e-05 -3.973950e-05
##               factor(x4)2   factor(x4)3
## (Intercept) -4.480514e-03 -5.013967e-03
## x1          -5.947661e-05 -1.744380e-06
## x2           3.654123e-05  3.223652e-05
## x3          -8.958214e-05 -3.973950e-05
## factor(x4)2  6.689116e-03  4.654743e-03
## factor(x4)3  4.654743e-03  7.779206e-03
```

In this case, $SE_{b_{x_2} - b_{x_1}} = \sqrt{Var_{x_1} + Var_{x_2} + 2Cov(x_1, x_2)}$. Therefore, the standard error of the difference could be calculated as below:

```
# the difference between the two unstandardized coefficients
diff = abs(m$coefficients['x2'])-abs(m$coefficients['x1'])
var_x1 = vc[2,2] # variance
var_x2 = vc[3,3] # variance
cov_x1x2 = vc[3,2] # covariance
se_diff = sqrt(var_x1+var_x2+2*cov_x1x2) # standard error of the difference
z = diff/se_diff # z score
names(z) = NULL
z
```

```
## [1] 0.5249465
```

It is less than 1.96, so it is not significant. We can use the following code to obtain the $p$ value:

```
pvalue = pnorm(-abs(z))
pvalue
```

```
## [1] 0.2998102
```

We can do something even more interesting, such as to test whether $b_{x_1} > 2.2$.

```
diff=abs(m$coefficients['x1'])-2.2
var_x1 = vc[2,2]
var_2.2 = 0
cov_x12.2 = 0
se_diff = sqrt(var_x1+var_2.2+2*cov_x12.2)
z = diff/se_diff
names(z) = NULL
pvalue = pnorm(-abs(z))
c(z,pvalue)
```

```
## [1] 0.1363163 0.4457856
```

Whether $b_{x_4=3} > 2b_{x_4=2}$?

```
diff=abs(m$coefficients['factor(x4)3']-2*m$coefficients['factor(x4)2'])
var_x43 = vc[6,6]
var_x42 = vc[5,5]
cov_x4 = vc[5,6]
se_diff = sqrt(var_x43+4*var_x42+4*cov_x4)
z = diff/se_diff
names(z) = NULL
pvalue = pnorm(-abs(z))
c(z,pvalue)
```

```
## [1] 0.4053082 0.3426255
```

Will standardization of variables change the variance-covariance matrix?

```
summary(m1<-lm(y~x1+x2,data = data))
```

```
##
## Call:
## lm(formula = y ~ x1 + x2, data = data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -8.362 -2.188 -0.203  2.467  8.195
##
## Coefficients:
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.80028    0.58031   9.995   <2e-16 ***
## x1          -2.21515    0.05634 -39.316   <2e-16 ***
## x2           2.17989    0.03585  60.813   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.385 on 997 degrees of freedom
## Multiple R-squared:  0.8019, Adjusted R-squared:  0.8015
## F-statistic:  2018 on 2 and 997 DF,  p-value: < 2.2e-16
```

```
# standardized all variables
y_s = (data$y-mean(data$y))/sd(data$y)
x1_s = (data$x1-mean(data$x1))/sd(data$x1)
x2_s = (data$x2-mean(data$x2))/sd(data$x2)
summary(m2<-lm( y_s ~ x1_s+x2_s-1)) # without intercept
```

```
##
## Call:
## lm(formula = y_s ~ x1_s + x2_s - 1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.10053 -0.28796 -0.02672  0.32471  1.07860
##
## Coefficients:
##      Estimate Std. Error t value Pr(>|t|)
## x1_s -0.59532    0.01513  -39.34   <2e-16 ***
## x2_s  0.92083    0.01513   60.84   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4453 on 998 degrees of freedom
## Multiple R-squared:  0.8019, Adjusted R-squared:  0.8015
## F-statistic:  2020 on 2 and 998 DF,  p-value: < 2.2e-16
```

```
lm.beta(m2)
```

```
##
## Call:
## lm(formula = y_s ~ x1_s + x2_s - 1)
##
## Standardized Coefficients::
##       x1_s       x2_s
## -0.5953206  0.9208314
```

The coefficients in the regression model based on the standardized variables (m2) are equal to the standardized coefficients. Check the variance-covariance matrices:

```
vcov(m1)
```

```
##              (Intercept)            x1            x2
```

```
## (Intercept)  0.336757352 -0.0227216826 -0.0081786313
## x1          -0.022721683  0.0031744467 -0.0007376377
## x2          -0.008178631 -0.0007376377  0.0012849171
```

```
vcov(m2)
```

```
##                x1_s          x2_s
## x1_s  2.290498e-04 -8.365682e-05
## x2_s -8.365682e-05  2.290498e-04
```

They are different. Any implications?

### 10. Why is the I.I.D. assumption essential?

A regression model could be write as a combination of two parts: random part + systematic part.

$$Y_i \sim \mathsf{Norm}(\mu_i, \sigma)$$
$$\mu_i = X_i\beta_1 + \beta_0$$

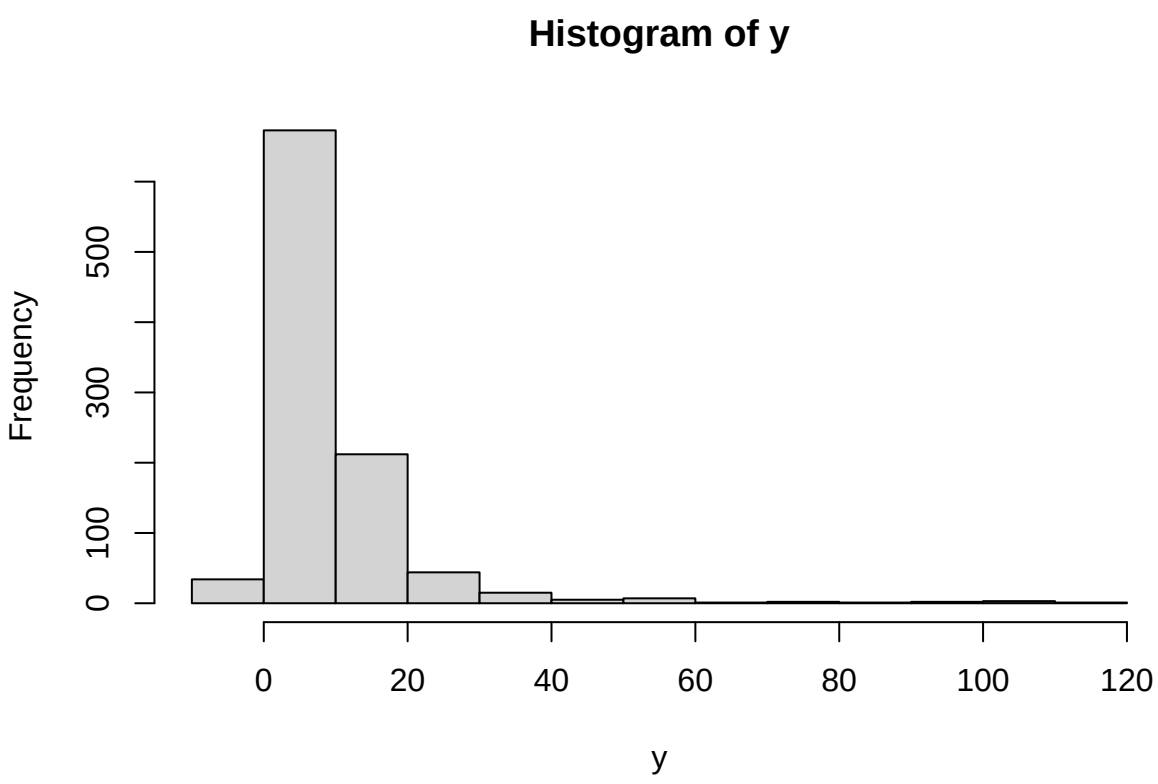
I.I.D refers to independent and identical distribution.

- Each draw from the bag must be independent. This means that the value you get on one draw does not depend in any way on other draws. Not repeated measures.
- Each observation is drawing from the same bag. In other words, you are drawing values from the same distribution. The shape of this distribution does not matter, even though some people will claim that it must be normally distributed. However, regardless of the shape, all observations must be drawing from an identically shaped distribution.

### 11. What if residuals are normally distributed, but the dependent variable is not? Is this possible?

Let's create a variable y that is determined by two skewed variables and a normally distributed random error.

```
x1 = rexp(1000,1.5)
x2 = rweibull(1000,0.5)
y = 7*x1+2*x2+rnorm(1000)
hist(y)
```




```
summary(lm(y~x1+x2))
```

```
##
## Call:
## lm(formula = y ~ x1 + x2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4346 -0.6721  0.0048  0.6975  2.8581
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.073214   0.045904  -1.595    0.111
## x1           7.007799   0.045737 153.218   <2e-16 ***
## x2           2.006097   0.005801 345.791   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9894 on 997 degrees of freedom
## Multiple R-squared:  0.993,  Adjusted R-squared:  0.993
## F-statistic: 7.043e+04 on 2 and 997 DF,  p-value: < 2.2e-16
```

As you can see, the dependent variable is skewed while the OLS regression estimated the coefficients correctly. The assumption is not about the shape of the dependent variable but how the random error of generating the dependent variable (the random part in the generation process).

**12. Why effect size is important? Is it true that larger coefficients indicate larger effect sizes? How to measure the unique effect of an independent variable on the dependent variable without any confounding effects of other independent variables (or can stepwise regression models solve the problem)?**

```
summary(m)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2 + x3 + factor(x4))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3584 -0.6282 -0.0032  0.6211  3.0564
##
## Coefficients:
##             Estimate Std. Error  t value Pr(>|t|)
## (Intercept) -0.11189    0.18107   -0.618    0.537
## x1          -2.20224    0.01643 -134.054   <2e-16 ***
## x2           2.21060    0.01044  211.646   <2e-16 ***
## x3           3.39508    0.06807   49.873   <2e-16 ***
## factor(x4)2  4.01667    0.08179   49.111   <2e-16 ***
## factor(x4)3  7.93990    0.08820   90.022   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9854 on 994 degrees of freedom
## Multiple R-squared:  0.9833, Adjusted R-squared:  0.9832
## F-statistic: 1.168e+04 on 5 and 994 DF,  p-value: < 2.2e-16
```

```
betas
```

```
##
## Call:
## lm(formula = y ~ x1 + x2 + x3 + factor(x4))
##
## Standardized Coefficients::
## (Intercept)          x1          x2          x3 factor(x4)2 factor(x4)3
##   0.0000000  -0.5918520   0.9338013   0.2046776   0.2642496   0.4839834
```

There are many effect size measures. $\beta$ could be used as effect size, but it is really hard to interpret. In regression models, $R^2$ is most intuitive (variance explained), but it is a overall measure.

**Squared Semi-partial correlation** ($sr^2$) tells us how much of the unique contribution of an independent variable to the total variation in dependent variable. In other words, it explains increment in R-square ($\Delta R^2$) when an independent variable is added.

```
a = summary(lm(y~x3+factor(x4),data))$r.squared #0.18
b = summary(lm(y~x1+x3+factor(x4),data))$r.squared #0.27
deta = b-a
deta
```

```
## [1] 0.06233502
```

Therefore, $x_1$ increased $R^2$ by 0.062335?

```
c = summary(lm(y~x1+x2+x3+factor(x4),data))$r.squared #0.98
deta2 = c-b
deta2
```

```
## [1] 0.7542091
```

Therefore, $x_2$ increased $R^2$ by 0.7542091? Why is this method problematic? Let's change the input order...

```
cor.test(data$x1,data$x2)
```

```
##
##  Pearson's product-moment correlation
##
## data:  data$x1 and data$x2
## t = 12.394, df = 998, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.3102662 0.4177682
## sample estimates:
##       cor
## 0.3652343
```

```
b_ = summary(lm(y~x2+x3+factor(x4),data))$r.squared
b_-a # variance x2 explained
```

```
## [1] 0.5139728
```

```
c - b_ # variance x1 explained
```

```
## [1] 0.3025714
```

It's a different story, which suggests that $x_1$ increased $R^2$ by 0.3025714, while $x_2$ $R^2$ by 0.5139728. The correct way to calculate the **Squared Semi-partial correlation** ($sr^2$) is to use the "one-less" approach:

- fit the full model with all predictors and calculate the overall $R^2_full$
- remove one predictor ($x_i$) from the full model and then fit the "one-less" model
- calculate the $R^2_{(i)}$ for the "one-less" model and $\Delta R^2_i = R^2_full - R^2_{(i)}$

```
# total
ovall = summary(lm(y~x1+x2+x3+factor(x4),data))$r.squared

# one less by removing x1 from m
ovall - summary(lm(y~x2+x3+factor(x4),data))$r.squared
```

```
## [1] 0.3025714
```

```r
# one less by removing x2 from m
ovall - summary(lm(y~x1+x3+factor(x4),data))$r.squared
```

```
## [1] 0.7542091
```

```r
# one less by removing x3 from m
ovall - summary(lm(y~x1+x2+factor(x4),data))$r.squared
```

```
## [1] 0.04187959
```

```r
# one less by removing x4 from m
ovall - summary(lm(y~x1+x2+x3,data))$r.squared
```

```
## [1] 0.1387724
```

In fact, $x_1$ can explain 0.3025714 variance. Why is the sum of all $R^2s$ larger than 100%?

**13. Is it possible to estimate the main effects from a regression model with interaction terms?**

Let's fit a model with interaction:

```r
summary(full <- lm(y~x1*x3+x2,data))
```

```
##
## Call:
## lm(formula = y ~ x1 * x3 + x2, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.9760 -1.8088 -0.1382  2.7011  6.5929
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.739153   0.600184   7.896 7.57e-15 ***
## x1          -2.212682   0.058927 -37.550  < 2e-16 ***
## x3           3.383495   1.028071   3.291  0.00103 **
## x2           2.180970   0.031801  68.581  < 2e-16 ***
## x1:x3        0.003995   0.101137   0.039  0.96850
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.002 on 995 degrees of freedom
## Multiple R-squared:  0.8445, Adjusted R-squared:  0.8439
## F-statistic:  1351 on 4 and 995 DF,  p-value: < 2.2e-16
```

Can we say the main effect of x1 is -2.21?

When $x_3 = 0$:

```
summary(lm(y~x1*x3+x2,data[data$x3==0,]))
```

```
##
## Call:
## lm(formula = y ~ x1 * x3 + x2, data = data[data$x3 == 0, ])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.2782 -1.7773 -0.1635  2.7101  6.5269
##
## Coefficients: (2 not defined because of singularities)
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.98922    0.61054   8.172 1.43e-15 ***
## x1          -2.18826    0.05993 -36.514  < 2e-16 ***
## x3                NA         NA      NA       NA
## x2           2.14016    0.03762  56.896  < 2e-16 ***
## x1:x3             NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.99 on 698 degrees of freedom
## Multiple R-squared:  0.8339, Adjusted R-squared:  0.8334
## F-statistic:  1752 on 2 and 698 DF,  p-value: < 2.2e-16
```

When $x_3 = 1$,

```
summary(lm(y~x1*x3+x2,data[data$x3==1,]))
```

```
##
## Call:
## lm(formula = y ~ x1 * x3 + x2, data = data[data$x3 == 1, ])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.5250 -1.7666 -0.1308  2.7100  5.5993
##
## Coefficients: (2 not defined because of singularities)
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.43589    0.95285   7.804 1.04e-13 ***
## x1          -2.26025    0.09010 -25.085  < 2e-16 ***
## x3                NA         NA      NA       NA
## x2           2.28039    0.05921  38.512  < 2e-16 ***
## x1:x3             NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.016 on 296 degrees of freedom
## Multiple R-squared:  0.8475, Adjusted R-squared:  0.8465
## F-statistic: 822.5 on 2 and 296 DF,  p-value: < 2.2e-16
```

It demonstrates that it is not the main effect but the effect when $x_3 = 0$. The main effects could be obtained in the following ways (the real value is in between):

```r
# margins based on the full model
library(margins)
margins(full)
```

```
##      x1    x3    x2
##  -2.211 3.423 2.181
```

```r
# fit a model without interaction term
summary(lm(y~x1+x2+x3,data))
```

```
##
## Call:
## lm(formula = y ~ x1 + x2 + x3, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.9844 -1.8080 -0.1365  2.7012  6.5914
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.72723    0.51850   9.117   <2e-16 ***
## x1          -2.21145    0.04995 -44.278   <2e-16 ***
## x2           2.18094    0.03178  68.636   <2e-16 ***
## x3           3.42327    0.20727  16.516   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.001 on 996 degrees of freedom
## Multiple R-squared:  0.8445, Adjusted R-squared:  0.844
## F-statistic:  1803 on 3 and 996 DF,  p-value: < 2.2e-16
```

The marginal effect represents the difference of (two) predictions for an (infinitesimal) change in x (the focal term). The average marginal effect represents the average slope of that predictor. In other words: the average marginal effects is one value per parameter (term), thus it can be considered as an **adjusted regression coefficient**, while predicted values usually predict the average outcome for different values of x - you usually don't have just one coefficient in the latter case that represents the overall effect of x.

## IV. Regression (GLM)

### 14. How to deal with non-normally distributed dependent variables? How to interpret the coefficients?

As mentioned in section III, there are many regression models in addition to OLS regression. Most regression models could be expressed as the combination of a random plus a systematic part to generate the dependent variable. As we presented in Question 10, the dependent variable in OLS regression is generated according to a normal distribution and the *mean* varies with other predictors systematically. In this sense, as long as we know the generating process (or underlying distribution) of the dependent variable, we can model the data in a regression form. We call them Generalized Regression Models (GLMs). Some commonly used models include:

- Logistic Regression (DV is a binary variable, which is generated from a Bernoulli distribution)
- Poisson Regression (DV is a count variable, which is generated from a Poisson distribution)
- Negative Binomial (DV is a over-dispersed count variable, which is generated from a Negative Binomial distribution)
- Beta Regression (DV is a percentage/proportion variable, which is generated from a Beta distribution)

#### Logistic Regression

$$Y_i \sim \mathsf{Bern}(\pi_i)$$
$$\mathsf{logit}(\pi_i) = \log(\frac{\pi_i}{1-\pi_i}) = X_i\beta_1 + \beta_0$$

In logistic regression, the dependent variable $Y_i$ is generated from a Bernoulli distribution with a single parameter $\pi_i$, which is the probability of occurrence (the event of interest). The **link function** is logit (not log). It connects the predictors $X_i$ to the parameter $\pi_i$.

$\beta_0$ is the log odds of the event of interest, when $X_i = 0$, $e^{\beta_0}$ is the odds. $\beta_1 = \log(odds_{x+1} - \log(odds_x)$, or $e^{\beta_1} = \frac{odds_{x+1}}{odds_x}$. It reflects the change in log odds.

```
# load the admission data for an example
mydata <- read.csv("https://stats.idre.ucla.edu/stat/data/binary.csv")
head(mydata)
```

```
##   admit gre  gpa rank
## 1     0 380 3.61    3
## 2     1 660 3.67    3
## 3     1 800 4.00    1
## 4     1 640 3.19    4
## 5     0 520 2.93    4
## 6     1 760 3.00    2
```

The `glm` function in R can fit most generalized linear models. For logistic regression, we set the `family = "binomial"`:

```
library(jtools)
mylogit <- glm(admit ~ gre + gpa + factor(rank), data = mydata, family = "binomial")
jtools::summ(mylogit)
```

| | |
|---|---|
| Observations | 400 |
| Dependent variable | admit |
| Type | Generalized linear model |
| Family | binomial |
| Link | logit |

| | |
|---|---|
| $\chi^2(5)$ | 41.46 |
| Pseudo-$R^2$ (Cragg-Uhler) | 0.14 |
| Pseudo-$R^2$ (McFadden) | 0.08 |
| AIC | 470.52 |
| BIC | 494.47 |

| | Est. | S.E. | z val. | p |
|---|---|---|---|---|
| (Intercept) | -3.99 | 1.14 | -3.50 | 0.00 |
| gre | 0.00 | 0.00 | 2.07 | 0.04 |
| gpa | 0.80 | 0.33 | 2.42 | 0.02 |
| factor(rank)2 | -0.68 | 0.32 | -2.13 | 0.03 |
| factor(rank)3 | -1.34 | 0.35 | -3.88 | 0.00 |
| factor(rank)4 | -1.55 | 0.42 | -3.71 | 0.00 |

Standard errors: MLE

The `jtools::summ` function are extremely useful when you want to obtain the *Pseudo-$R^2$*. Since the relationships in logistic regression are non-linear. Usually, they are less intuitive to be interpreted. We can use the `margins` and `ggpredict` to estimate the predicted probabilities, which will help us to interpret the results.

```
library(margins)
library(ggeffects)

margins(mylogit,variables = "rank")
```

```
##    rank2   rank3   rank4
##  -0.1566 -0.2872 -0.3212
```

```
ggpredict(mylogit,terms = "rank")
```

```
## # Predicted probabilities of admit
##
## rank | Predicted |       95% CI
## -------------------------------
## 1    |      0.51 | [0.38, 0.64]
## 2    |      0.35 | [0.27, 0.43]
## 3    |      0.22 | [0.15, 0.30]
## 4    |      0.18 | [0.11, 0.30]
##
## Adjusted for:
## * gre = 580.00
## * gpa =   3.39
```

For example, the marginal effects report that compare to rank1, rank2's probability to be admitted is 15.7% lower.

**Poisson Regression**

$$Y_i \sim \text{Pois}(\lambda_i)$$
$$\log(\lambda_i) = X_i\beta_1 + \beta_0$$

$\beta_0$ is the logged average of $Y$, when $X_i = 0$. $e^{\beta_0}$ is the average of $Y$. $\beta_1 = \log(\lambda_{x+1}) - \log(\lambda_x)$, or $e^{\beta_1} = \frac{\lambda_{x+1}}{\lambda_x}$.

```
head(warpbreaks)
```

```
##   breaks wool tension
## 1     26    A       L
## 2     30    A       L
## 3     54    A       L
## 4     25    A       L
## 5     70    A       L
## 6     52    A       L
```

```
mypois <-glm(breaks ~ wool+tension, data = warpbreaks,family = poisson)
summ(mypois)
```

| | |
|---|---|
| Observations | 54 |
| Dependent variable | breaks |
| Type | Generalized linear model |
| Family | poisson |
| Link | log |

| | |
|---|---|
| $\chi^2(3)$ | 86.98 |
| Pseudo-$R^2$ (Cragg-Uhler) | 0.80 |
| Pseudo-$R^2$ (McFadden) | 0.15 |
| AIC | 493.06 |
| BIC | 501.01 |

| | Est. | S.E. | z val. | p |
|---|---|---|---|---|
| (Intercept) | 3.69 | 0.05 | 81.30 | 0.00 |
| woolB | -0.21 | 0.05 | -3.99 | 0.00 |
| tensionM | -0.32 | 0.06 | -5.33 | 0.00 |
| tensionH | -0.52 | 0.06 | -8.11 | 0.00 |

Standard errors: MLE

**Negative Binomial**:

$$Y_i \sim \text{NegBin}(\mu_i, r)$$
$$\log(\mu_i) = X_i\beta_1 + \beta_0$$

Both Poisson regression and Negative Binomial regression are models for count data. Therefore, the interpretation of coefficients of Negative Binomial is similar to that of Poisson regression. The difference is that $E(Y) = Var(Y)$ in Poisson distribution, while $E(Y) \neq Var(Y)$ in Negative Binomial distribution. This

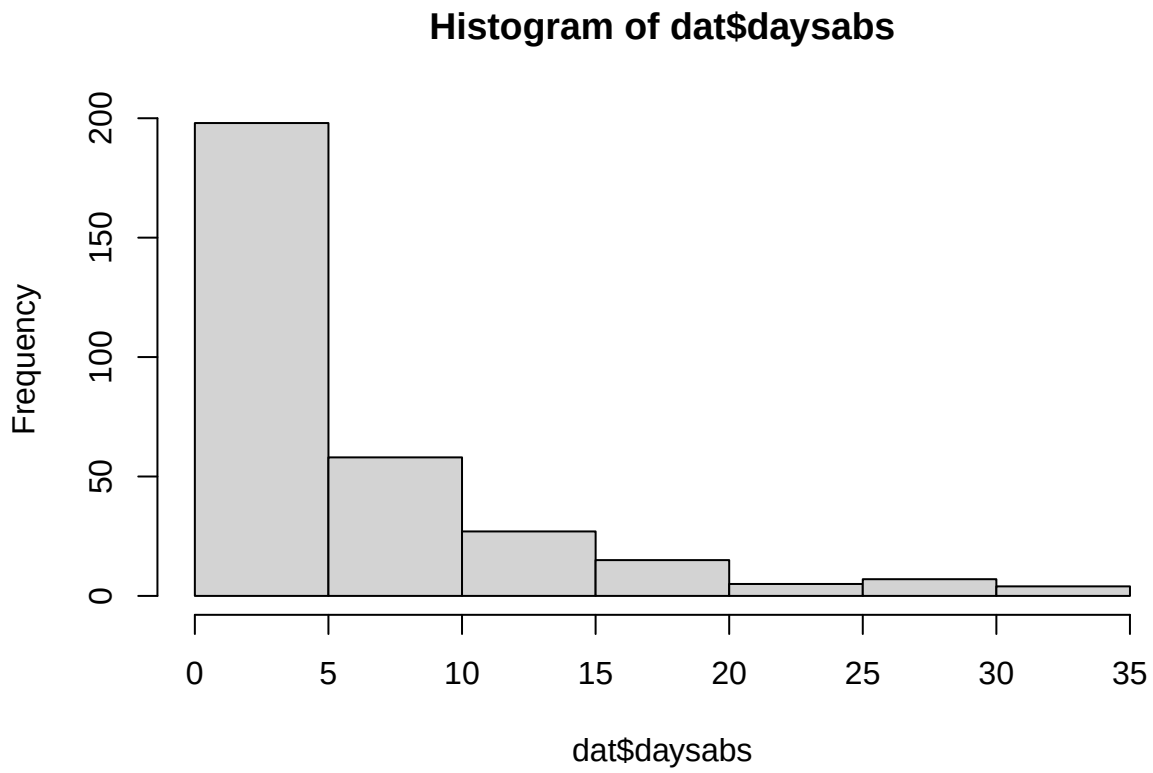
is governed by the dispersion parameter $r$. For large reciprocal dispersion parameter $r$, $Var(Y) \approx E(Y)$, NegBin $\rightarrow$ Pois. The Negative Binomial distribution is especially useful to model highly skewed data.

Unfortunately, we cannot use `glm` to fit a Negative Binomial model. Instead, we need to use `glm.nb` function from the `MASS` package:

```
library(MASS)
library(haven) # to read stata data
dat <- read_stata("https://stats.idre.ucla.edu/stat/stata/dae/nb_data.dta")
head(dat)
```

```
## # A tibble: 6 x 5
##      id     gender  math daysabs  prog
##   <dbl>  <dbl+lbl> <dbl>   <dbl> <dbl>
## 1  1001 2 [male]      63       4     2
## 2  1002 2 [male]      27       4     2
## 3  1003 1 [female]    20       2     2
## 4  1004 1 [female]    16       3     2
## 5  1005 1 [female]     2       3     2
## 6  1006 1 [female]    71      13     2
```

```
hist(dat$daysabs)
```




```
summ(mynegbin <- glm.nb(daysabs ~ math + prog, data = dat))
```

| | |
|---|---|
| Observations | 314 |
| Dependent variable | daysabs |
| Type | Generalized linear model |
| Family | Negative Binomial(1.0222) |
| Link | log |

| | | | | |
|---|---|---|---|---|
| $\chi^2()$ | 0.17 | 0.03 | 1741.52 | 1756.51 |
| Pseudo-$R^2$ (Cragg-Uhler) | 0.17 | 0.03 | 1741.52 | 1756.51 |
| Pseudo-$R^2$ (McFadden) | 0.17 | 0.03 | 1741.52 | 1756.51 |
| AIC | 0.17 | 0.03 | 1741.52 | 1756.51 |
| BIC | 0.17 | 0.03 | 1741.52 | 1756.51 |

| | Est. | S.E. | z val. | p |
|---|---|---|---|---|
| (Intercept) | 3.49 | 0.23 | 15.48 | 0.00 |
| math | -0.01 | 0.00 | -2.72 | 0.01 |
| prog | -0.68 | 0.10 | -6.98 | 0.00 |

Standard errors: MLE

The theta parameter estimated above is the dispersion parameter. Note that R parameterizes this differently from SAS, Stata, and SPSS. The R parameter (theta) is equal to the inverse of the dispersion parameter estimated in these other software packages. Thus, the theta value of 1.022 seen here is equivalent to the 0.978 because $1/1.022 = 0.978$.

**Beta Regression**

$$Y_i \sim \mathsf{Beta}(\mu_i, \phi)$$

$$\mathsf{logit}(\mu_i) = \log(\frac{\mu_i}{1-\mu_i}) = X_i\beta_1 + \beta_0$$

Beta regression also uses logit as the link function. The distribution ranges from 0 to 1 (0 and 1 are not usually excluded). Therefore, it is useful to model percent and proportion data. We use `betareg` to fit the model. The parameter $\phi$ is a precision parameter: the higher $\phi$ the lower the variance for given *mean* $\mu$.

```
library(betareg)
data("FoodExpenditure", package = "betareg")
head(FoodExpenditure)
```

```
##     food income persons
## 1 15.998 62.476       1
## 2 16.652 82.304       5
## 3 21.741 74.679       3
## 4  7.431 39.151       3
## 5 10.481 64.724       5
## 6 13.548 36.786       3
```

```
# food/income is a proportion
mybeta <- betareg(I(food/income) ~ income + persons, data = FoodExpenditure)
summary(mybeta)
```

```
##
```

```
## Call:
## betareg(formula = I(food/income) ~ income + persons, data = FoodExpenditure)
##
## Standardized weighted residuals 2:
##     Min      1Q  Median      3Q     Max
## -2.7818 -0.4445  0.2024  0.6852  1.8755
##
## Coefficients (mean model with logit link):
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.622548   0.223854  -2.781 0.005418 **
## income      -0.012299   0.003036  -4.052 5.09e-05 ***
## persons      0.118462   0.035341   3.352 0.000802 ***
##
## Phi coefficients (precision model with identity link):
##       Estimate Std. Error z value Pr(>|z|)
## (phi)    35.61       8.08   4.407 1.05e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Type of estimator: ML (maximum likelihood)
## Log-likelihood: 45.33 on 4 Df
## Pseudo R-squared: 0.3878
## Number of iterations: 28 (BFGS) + 4 (Fisher scoring)
```

```
margins(mybeta)
```

```
## Average marginal effects

## betareg(formula = I(food/income) ~ income + persons, data = FoodExpenditure)

##    income persons
##  -0.00248 0.02389
```

Therefore, one more person leads to 2.4% increase in food consumption given the income (food/income).

**15. Can we model the variance (instead of mean) of the dependent variable? E.g., the variance of the salary of elder people is smaller than the variance of the salary of young people, given equal salary.**

Recall the expression for normal regression. We simply change the systematic part by including $\sigma$:

$$Y_i \sim \mathsf{Norm}(\mu, \sigma_i)$$
$$\sigma_i = X_i\beta_1 + \beta_0$$

In fact, we can model both $\mu_i$ and $\sigma_i$ simultaneously. We can fit the models using the `gamlss` (Generalized Additive Models for Location, Scale and Shape) package in R (see more example here).

```
library(gamlss)
head(mtcars)
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

Let's fit an OLS regression model:

```
summ(lm(mpg~hp+wt,mtcars),model.info = FALSE, model.fit = FALSE)
```

| | Est. | S.E. | t val. | p |
|---|---|---|---|---|
| (Intercept) | 37.23 | 1.60 | 23.28 | 0.00 |
| hp | -0.03 | 0.01 | -3.52 | 0.00 |
| wt | -3.88 | 0.63 | -6.13 | 0.00 |

Standard errors: OLS

Model $\sigma$ using `gamlss`. There are two formulas, one for the *mean* and another for the *standard deviation*. Set the first part as in `lm`. If you really believe that the *mean* is not related to other variables, `~1` indicates that only the intercept is included in the model (thus is a constant). The second part is `sigma.formula`, using `=~` to specify the predictors. You can specify the family, which means you can go beyond Normal distributions. Here, `family = NO()`, indicating a Normal distribution.

```
myglass1 = gamlss(mpg~1,sigma.formula = ~ hp+wt, family = NO(),
                  data=mtcars,
                  control = gamlss.control(trace=FALSE))
myglass2 = gamlss(mpg~hp+wt,sigma.formula = ~ hp+wt, family = NO(),
                  data=mtcars,
                  control = gamlss.control(trace=FALSE))
summary(myglass1)
```

```
## ******************************************************************
## Family:  c("NO", "Normal")
##
## Call:  gamlss(formula = mpg ~ 1, sigma.formula = ~hp + wt,
##     family = NO(), data = mtcars, control = gamlss.control(trace = FALSE))
##
## Fitting method: RS()
##
## ------------------------------------------------------------------
## Mu link function:  identity
## Mu Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  15.0982     0.3599   41.95   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
## Sigma link function:  log
## Sigma Coefficients:
```

```
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.356603   0.355505   9.442 3.37e-10 ***
## hp          -0.013336   0.003307  -4.033 0.000385 ***
## wt           0.060041   0.146335   0.410 0.684713
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
## No. of observations in the fit:  32
## Degrees of Freedom for the fit:  4
##       Residual Deg. of Freedom:  28
##                       at cycle:  13
##
## Global Deviance:     192.8026
##             AIC:     200.8026
##             SBC:     206.6656
## ******************************************************************
```

```
summary(myglass2)
```

```
## ******************************************************************
## Family:  c("NO", "Normal")
##
## Call:  gamlss(formula = mpg ~ hp + wt, sigma.formula = ~hp +
##     wt, family = NO(), data = mtcars, control = gamlss.control(trace = FALSE))
##
##
## Fitting method: RS()
##
## ------------------------------------------------------------------
## Mu link function:  identity
## Mu Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 35.97612    2.47576  14.531 5.40e-14 ***
## hp          -0.02782    0.00713  -3.901 0.000604 ***
## wt          -3.69296    0.79155  -4.666 8.13e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
## Sigma link function:  log
## Sigma Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.224105   0.545910   2.242   0.0337 *
## hp          -0.003228   0.002775  -1.163   0.2553
## wt           0.039559   0.224216   0.176   0.8613
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
## No. of observations in the fit:  32
## Degrees of Freedom for the fit:  6
##       Residual Deg. of Freedom:  26
##                       at cycle:  8
```

```
##
## Global Deviance:     146.998
##              AIC:    158.998
##              SBC:    167.7924
## ****************************************************************
```

Another (maybe more intuitive/flexible) way is to use Bayes approach. It is easy to set the model using the `brms` packages in R. The code turns out to be very slow. Please use at your own risk.

```
# library(brms)
# myBayes <- brm(bf(mpg ~ hp + wt, sigma ~ hp+wt),
#              data = mtcars,
#              family = gaussian)
# summary(myBayes)
```

**16. What is wrong with using Z-statistics (and associated p-values) of the coefficient of a multiplicative term to test for a statistical interaction in nonlinear models with categorical dependent variables (e.g., logistic regression)?**

In linear models, the interaction term tests the difference between the two slopes. However, interactions in non-linear models (e.g., logistic regression) could be more complicated. Critically, in binary probit and logit, the equality of regression coefficients across groups does not imply that the marginal effects of a predictor on the probability are equal (Long & Mustillo, 2021).

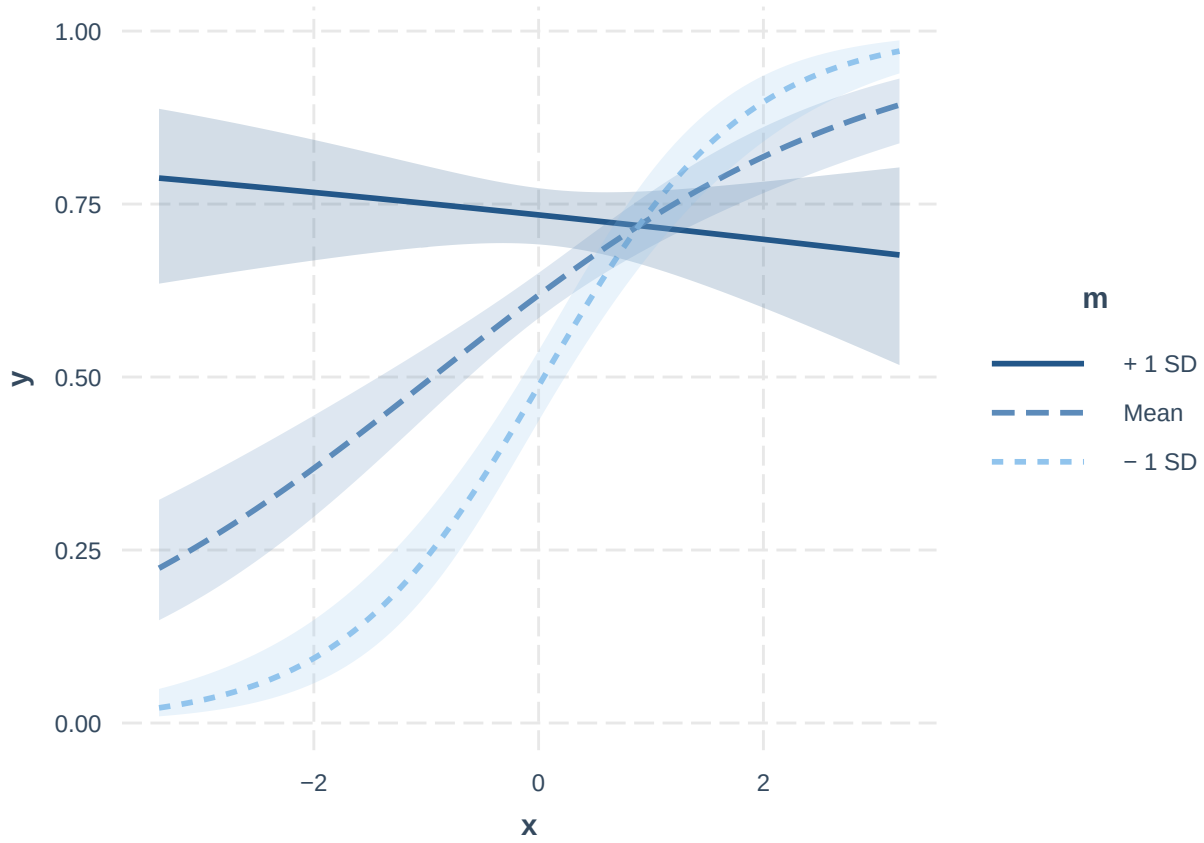
```
# generate a data set with binary y
x <- rnorm(1000)
m <- rnorm(1000)
prob <- binomial(link = "logit")$linkinv(.25 + .3*x + .3*m + -.5*(x*m) + rnorm(1000))
y <- rep(0, 1000)
y[prob >= .5] <- 1
summ(logit_fit <- glm(y ~ x * m, family = binomial),model.info = FALSE, model.fit = FALSE)
```

| | Est. | S.E. | z val. | p |
|---|---|---|---|---|
| (Intercept) | 0.48 | 0.07 | 6.75 | 0.00 |
| x | 0.52 | 0.07 | 7.07 | 0.00 |
| m | 0.56 | 0.08 | 6.85 | 0.00 |
| x:m | -0.62 | 0.09 | -6.88 | 0.00 |

Standard errors: MLE

The interaction term x:m is significant. However, due to the non-linear (log) transformation, the slope differs at different values for x, thus, the **marginal effect** or **association** (in terms of probabilities) is not constant across values of x. Let's plot out the interaction effect:

```
library(interactions)
interact_plot(logit_fit, pred = x, modx = m, interval = T)
```




Let's calculate the marginal effect for x:

```
summary(ef <- margins(logit_fit))
```

```
##  factor    AME     SE      z      p  lower  upper
##       m 0.1130 0.0137 8.2625 0.0000 0.0862 0.1398
##       x 0.1043 0.0123 8.5108 0.0000 0.0803 0.1283
```

On average, a unit-change in x changes the predicted probability that the outcome equals 1 by 0.1 (see here).

It might be less intuitive to interpret average marginal effects, in particular for non-Gaussian models, because it is harder to understand an average effect where we actually have varying effects across the range of the focal term. Instead, it would be better to look at predictions at different values of the focal term(s), which is what `ggeffects` returns by default:

```
ggpredict(logit_fit, "x")
```

```
## # Predicted probabilities of y
##
##  x | Predicted |       95% CI
## ----------------------------
## -4 |      0.17 | [0.10, 0.27]
## -3 |      0.26 | [0.18, 0.35]
## -2 |      0.37 | [0.30, 0.44]
## -1 |      0.49 | [0.44, 0.54]
```

```
##  0 |      0.62 | [0.59, 0.65]
##  1 |      0.73 | [0.69, 0.77]
##  2 |      0.82 | [0.77, 0.86]
##  4 |      0.93 | [0.87, 0.96]
##
## Adjusted for:
## * m = 0.01
```

The non-linear relationship makes the interaction effect vary. We can estimate the marginal effects (of x) based on different levels of the moderator (m):

```
summary(margins(logit_fit,at = list(m=c(-2.5,0,2.5)),variables = "x"))
```

```
##  factor       m     AME     SE       z      p   lower   upper
##       x -2.5000  0.2895 0.0167 17.3014 0.0000  0.2567  0.3223
##       x  0.0000  0.1154 0.0146  7.9035 0.0000  0.0868  0.1440
##       x  2.5000 -0.1307 0.0301 -4.3461 0.0000 -0.1897 -0.0718
```

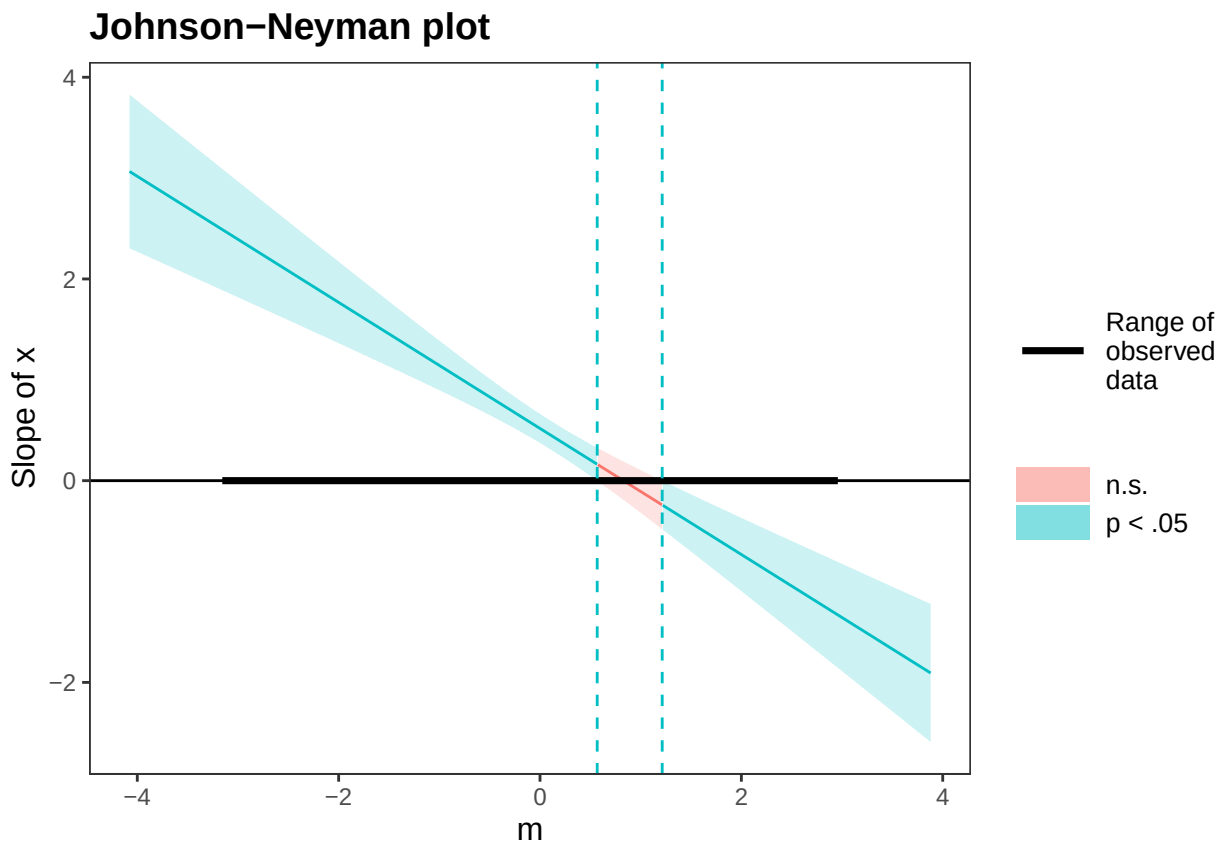
Or we can formally conduct the Johnson-Neyman intervals and simple slopes analysis. The "classic" way of probing an interaction effect is to calculate the slope of the focal predictor at different values of the moderator. When the moderator is binary, this is especially informative, e.g., what is the slope for men vs. women? But you can also arbitrarily choose points for continuous moderators.

With that said, the more statistically rigorous way to explore these effects is to find the **Johnson-Neyman interval**, which tells you the range of values of the moderator in which the slope of the predictor is significant vs. nonsignificant at a specified alpha level.

The `sim_slopes` function will by default find the **Johnson-Neyman interval** and tell you the predictor's slope at specified values of the moderator; by default either both values of binary predictors or the mean and the mean +/- one standard deviation for continuous moderators.

```
sim_slopes(logit_fit,pred = x, modx = m, jnplot = TRUE)
```

```
## JOHNSON-NEYMAN INTERVAL
##
## When m is OUTSIDE the interval [0.57, 1.21], the slope of x is p < .05.
##
## Note: The range of observed values of m is [-3.12, 2.92]
```




```
## SIMPLE SLOPES ANALYSIS
##
## Slope of x when m = -0.94699290 (- 1 SD):
##
##   Est.   S.E.   z val.      p
## ------ ------ -------- ------
##   1.11   0.12     9.16   0.00
##
## Slope of x when m =  0.01038015 (Mean):
##
##   Est.   S.E.   z val.      p
## ------ ------ -------- ------
##   0.51   0.07     6.99   0.00
##
## Slope of x when m =  0.96775320 (+ 1 SD):
##
##    Est.   S.E.   z val.      p
## ------- ------ -------- ------
##   -0.09   0.11    -0.83   0.41
```

## V. Regression (Causal Inference)

### 17. Everyone knows that correlation is not causation, when regression coefficients could be interpreted as causal effects? Is including the lagged explanatory variable a way to go?

Regression models have two different purposes. One is for description, another is for causal inference. For the purpose of description, we emphasize the overall fit (e.g., $R^2$). For the purpose of causal inference, we place more emphasis on the "accurate" estimation of a focal variable on the dependent variable (i.e., the treatment effect). So, a natural question is whether it is possible to estimate the treatment effect even though the overall model fit is not so good (e.g., relatively low $R^2$).

**Example 1 (independent)**: Let's create three random variables (x1, x2, Treatment). They are independent to each other. The focal variable is "Treatment", while the dependent variable is created based on x1 and Treatment.

```
x1 = rnorm(1000,2,1)
x2 = rnorm(1000,3,1)
Treatment = rnorm(1000,5,2)
y = 2*x1+3*Treatment+rnorm(1000)
```

We can run a regression model including all three variables as predictors of y.

```
library(jtools)
summ(lm(y~x1+x2+Treatment),model.info = FALSE, model.fit = FALSE)
```

| | Est. | S.E. | t val. | p |
|---|---|---|---|---|
| (Intercept) | 0.04 | 0.15 | 0.26 | 0.80 |
| x1 | 2.04 | 0.03 | 61.79 | 0.00 |
| x2 | 0.01 | 0.03 | 0.26 | 0.79 |
| Treatment | 2.98 | 0.02 | 187.89 | 0.00 |

Standard errors: OLS

For sure, the $R^2$ is close to 1, and the coefficients are close to the true values. How about if only regress Treatment on y (imaging that we have no way to observe or measure x1)?

```
summ(lm(y~Treatment),model.info = FALSE, model.fit = FALSE)
```

| | Est. | S.E. | t val. | p |
|---|---|---|---|---|
| (Intercept) | 4.20 | 0.19 | 22.33 | 0.00 |
| Treatment | 2.96 | 0.03 | 85.15 | 0.00 |

Standard errors: OLS

The coefficient of Treatment on y remains accurate, when other predictors are independent to Treatment.

**Example 2 (dependent)**: Let's create three random variables. This time, x1 is positively correlated with x2. Actually x1 leads to x2 ($x_1 \rightarrow x_2$). Treatment is negatively dependent on x2 ($x_2 \rightarrow$Treatment).

```
x1 = rnorm(1000,2,1)
x2 = 3.1*x1 + rnorm(1000)
Treatment = -0.9*x2 + rnorm(1000)
y = 2*x1+3*Treatment+rnorm(1000)
summ(lm(y~Treatment+x1+x2),model.info = FALSE, model.fit = FALSE)
```

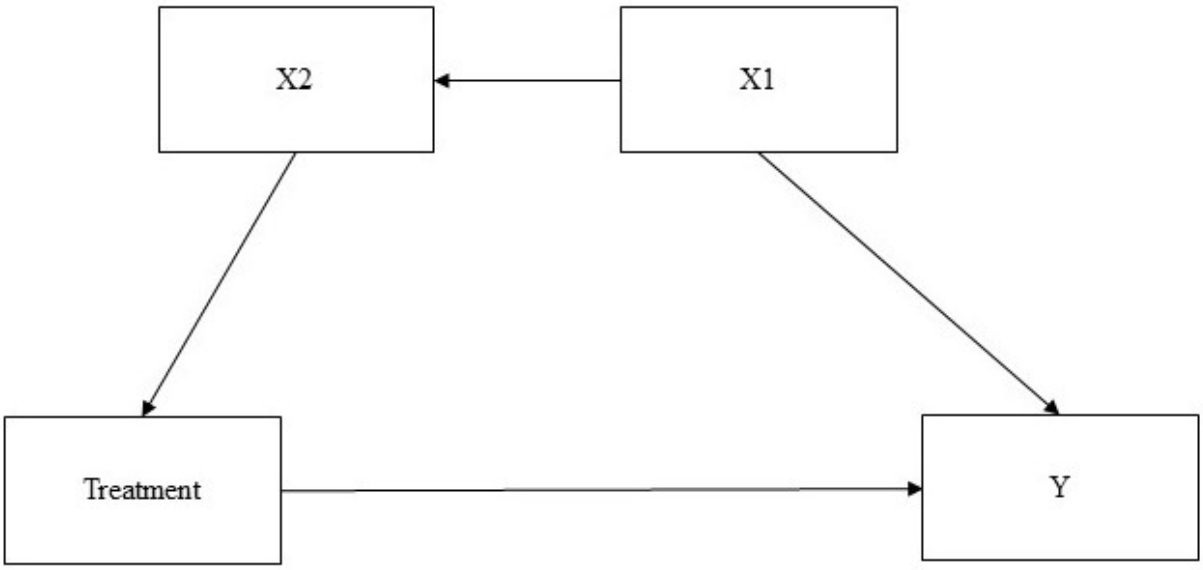



Figure 1: Causal Graph.

| | Est. | S.E. | t val. | p |
|---|---|---|---|---|
| (Intercept) | -0.02 | 0.07 | -0.26 | 0.80 |
| Treatment | 3.01 | 0.03 | 100.23 | 0.00 |
| x1 | 1.99 | 0.10 | 19.84 | 0.00 |
| x2 | 0.01 | 0.04 | 0.32 | 0.75 |

Standard errors: OLS

If we measured all predictors (x1, x2, and Treatment) and included them correctly in the regression model, the estimated coefficient of Treatment should be accurate as presented above. If we miss both x1 and x2, the estimated effect of Treatment will be biased:

```
summ(lm(y~Treatment),model.info = FALSE, model.fit = FALSE)
```

| | Est. | S.E. | t val. | p |
|---|---|---|---|---|
| (Intercept) | 0.80 | 0.09 | 9.36 | 0.00 |
| Treatment | 2.43 | 0.01 | 180.98 | 0.00 |

Standard errors: OLS

But, how to correct the bias? Should we control for both x1 and x2? The answer is "not necessary".

```
m0 <- lm(y~Treatment+x2)
m1 <- lm(y~Treatment+x1)
export_summs(m0,m1)
```

Both models can estimate the treatment effect correctly (However, the estimation of $x_s$ is biased). Why? This is rooted in the so-called **Backdoor Criterion** in causal inferences: Given an ordered pair of variables (X, Y) in a directed acyclic graph G, a set of variables Z satisfies the backdoor criterion relative to (X, Y) if no node in Z is a descendant of X, and Z blocks every path between X and Y that contains an arrow into X.

| | Model 1 | Model 2 |
|---|---|---|
| (Intercept) | 0.37 *** | -0.02 |
| | (0.08) | (0.07) |
| Treatment | 3.04 *** | 3.00 *** |
| | (0.04) | (0.02) |
| x2 | 0.62 *** | |
| | (0.03) | |
| x1 | | 2.01 *** |
| | | (0.07) |
| N | 1000 | 1000 |
| R2 | 0.98 | 0.98 |

*** p < 0.001; ** p < 0.01; * p < 0.05.

Question:

- Why we also consider demographics (age/gender) as control variables? (exogenous variables without any arrow directed to them)

**Example 3 (lagged variable)**: If we don't know either x1 or x2, is a lagged variable of the Treatment sufficient to identify the causal effect? It works in some special conditions only!

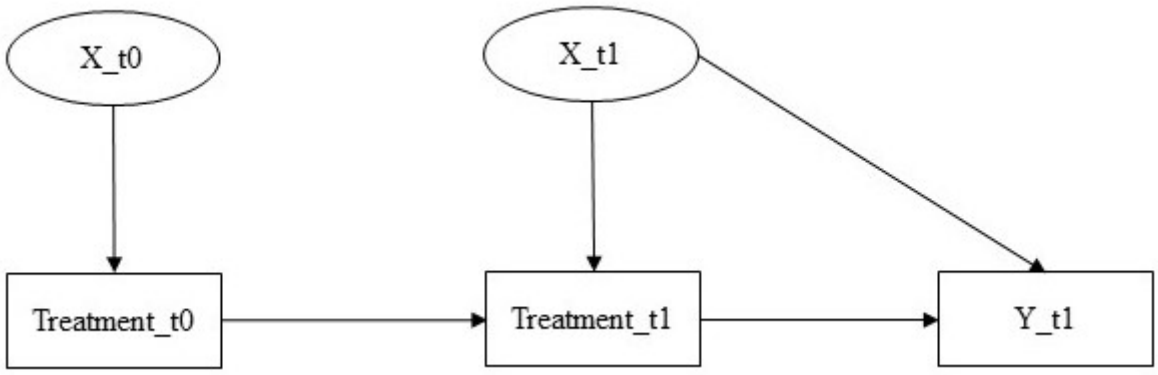



Figure 2: Another Causal Graph.

Note that x_t0 and x_t1 are unobservable.

```
# two independent confounding variables
x_t0 = rnorm(1000,1,1.5)
x_t1 = rnorm(1000,2,1.5)
```

```
# treatment and its lagged variable [simultaneous influences from x on treatment]
Treatment_t0 = 1.5*x_t0 + rnorm(1000,3,1.2)
Treatment_t1 = 2*Treatment_t0 + 1.5*x_t1 + rnorm(1000) # lagged var. should be auto-correlated

# define y
y = 2*Treatment_t1 + 3*x_t1 + rnorm(1000) # the treatment effect is 2

summ(lm(y~Treatment_t1+Treatment_t0),model.info = FALSE, model.fit = FALSE)
```

| | Est. | S.E. | t val. | p |
|---|---|---|---|---|
| (Intercept) | 1.01 | 0.16 | 6.19 | 0.00 |
| Treatment_t1 | 3.70 | 0.03 | 134.73 | 0.00 |
| Treatment_t0 | -3.42 | 0.06 | -55.50 | 0.00 |

Standard errors: OLS

No! The estimated effect of Treatment_t01 is bias. Notice that x_t0 actually is an **instrumental variable** of x_t1 -> y. We need to use two stage least square regression (or IV regression) to estimate the treatment effect.

```
library(ivreg)
summary(ivreg(y~Treatment_t1|Treatment_t0))
```

```
##
## Call:
## ivreg(formula = y ~ Treatment_t1 | Treatment_t0)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.1865  -3.0373  -0.1003   3.0326  15.2743
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   6.00005    0.39231   15.29   <2e-16 ***
## Treatment_t1  1.99438    0.03036   65.68   <2e-16 ***
##
## Diagnostic tests:
##                  df1 df2 statistic p-value
## Weak instruments   1 998      3967  <2e-16 ***
## Wu-Hausman         1 997      3080  <2e-16 ***
## Sargan             0  NA        NA      NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.673 on 998 degrees of freedom
## Multiple R-Squared: 0.8791,  Adjusted R-squared: 0.8789
## Wald test:  4314 on 1 and 998 DF,  p-value: < 2.2e-16
```

The coefficient is very close to the true value 2. So, is this the solution? No, because it is very unlikely x_t0 is not correlated in x_t1. And it is likely that x_t0 can influence y directly (cross-lagged influence).

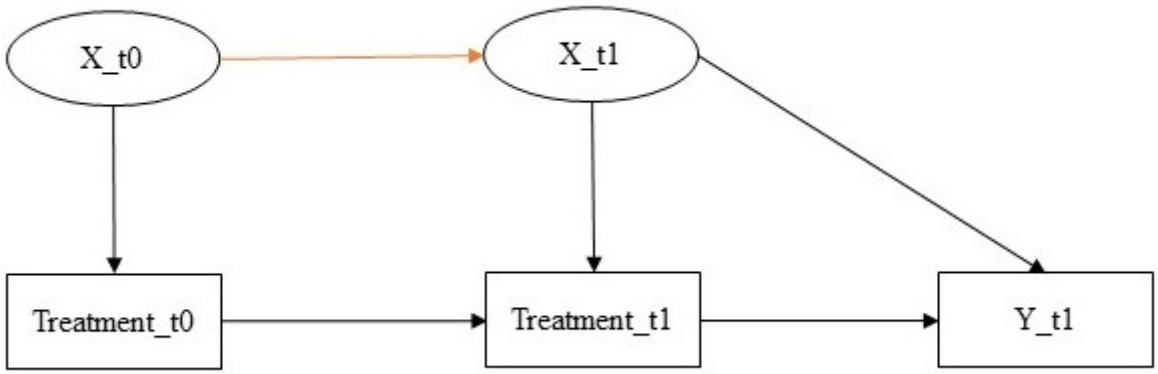



Figure 3: Revised Causal Graph.

```
# confounding variables
x_t0 = rnorm(1000,1,1.5)
x_t1 = 1.8*x_t0+rnorm(1000) #x_t0->x_t1

# treatment and its lagged variable [simultaneous influences from x on treatment]
treatment_t0 = 1.5*x_t0 + rnorm(1000,3,1.2)
treatment_t1 = 2*treatment_t0 + 1.5*x_t1 + rnorm(1000) # lagged var. should be auto-correlated

# define y
y = 2*treatment_t1 + 3*x_t1 + rnorm(1000)

summ(lm(y~treatment_t1+treatment_t0),model.info = FALSE, model.fit = FALSE)
```

| | Est. | S.E. | t val. | p |
|---|---|---|---|---|
| (Intercept) | -1.41 | 0.17 | -8.45 | 0.00 |
| treatment_t1 | 3.68 | 0.03 | 137.87 | 0.00 |
| treatment_t0 | -2.88 | 0.09 | -30.36 | 0.00 |

Standard errors: OLS

```
summary(ivreg(y~treatment_t1|treatment_t0))
```

```
##
## Call:
## ivreg(formula = y ~ treatment_t1 | treatment_t0)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.5128 -1.9747 -0.0374  2.0725 10.3754
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -4.45223    0.16054  -27.73   <2e-16 ***
## treatment_t1  2.83978    0.01096  259.02   <2e-16 ***
##
## Diagnostic tests:
```

```
##                  df1 df2 statistic p-value
## Weak instruments   1 998   11858.7  <2e-16 ***
## Wu-Hausman         1 997     921.4  <2e-16 ***
## Sargan             0  NA        NA      NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.111 on 998 degrees of freedom
## Multiple R-Squared: 0.9871,  Adjusted R-squared: 0.987
## Wald test: 6.709e+04 on 1 and 998 DF,  p-value: < 2.2e-16
```

Both `lm` and `ivreg` are incorrect! How about if x does not change over time, i.e., x_t0 = x_t1.

```
# confounding variables
x = rnorm(1000,3,1.5)

# treatment and its lagged variable [simultaneous influences from x on treatment]
treatment_t0 = 1.5*x + rnorm(1000,3,1.2)
treatment_t1 = 2*treatment_t0 + 1.5*x + rnorm(1000) # lagged var. should be auto-correlated

# define y
y = 2*treatment_t1 + 3*x + rnorm(1000)

summ(lm(y~treatment_t1+treatment_t0),model.info = FALSE, model.fit = FALSE)
```

| | Est. | S.E. | t val. | p |
|---|---|---|---|---|
| (Intercept) | -1.08 | 0.18 | -5.99 | 0.00 |
| treatment_t1 | 3.06 | 0.04 | 80.37 | 0.00 |
| treatment_t0 | -1.43 | 0.11 | -13.33 | 0.00 |

Standard errors: OLS

Does not work!

**18. Is a random experiment always better than other methods to identify causality? Why or why not?**

- Plausibility and ethics: sex manipulation?
- Experiments also have assumptions: Randomization rules out all confounding variables
- Compliance
- Heterogeneous effects (lack of external validity):

Suppose the treatment effect (conditions) varies across gender (`gender*cond`).

```
cond = sample(c(0,1),1000,replace = T)
gender = sample(c(0,1),1000,replace = T)
y = 2*cond -1*gender*cond + rnorm(1000)

pp = data.frame(y,cond,gender)
# a random experiment based on population
table(pp$cond)/1000
```

```
##
##     0     1
## 0.506 0.494
```

```
table(pp$gender)/1000
```

```
##
##     0     1
## 0.482 0.518
```

```
summ(lm(y~cond,data = pp),model.info = FALSE, model.fit = FALSE)
```

| | Est. | S.E. | t val. | p |
|---|---|---|---|---|
| (Intercept) | 0.02 | 0.05 | 0.53 | 0.59 |
| cond | 1.43 | 0.06 | 22.02 | 0.00 |

Standard errors: OLS

It does not influence the estimation of the treatment effect if we have a representative sample. However, it does not work with a biased sample.

```
# a random experiment from a biased sample
male = pp[pp$gender==1,]
female = pp[pp$gender==0,]
ps = rbind(male[sample(nrow(male),400),],female[sample(nrow(female),100),])

table(ps$cond)/500
```

```
##
##     0     1
## 0.496 0.504
```

```
table(ps$gender)/500
```

```
##
##   0   1
## 0.2 0.8
```

```
summ(lm(y~cond,data = ps),model.info = FALSE, model.fit = FALSE)
```

| | Est. | S.E. | t val. | p |
|---|---|---|---|---|
| (Intercept) | 0.02 | 0.07 | 0.31 | 0.76 |
| cond | 1.28 | 0.09 | 13.88 | 0.00 |

Standard errors: OLS

**19. Is it always better to control more variables than less? Should we remove non-significant variables from the regression?**

**Collider**: $x_1 \rightarrow x \leftarrow y$. Don't control for colliders. Controlling for colliders creates spurious correlations.

```r
x1 = rnorm(1000)
x2 = rnorm(1000)
y = 3.5*x2 + rnorm(1000)

x = 3*x1+2*y+rnorm(1000) # x is a collider

m0 <- lm(y~x1+x2) # correct (x1 does not influence y)
m1 <- lm(y~x1+x2+x) # wrong (x1 influences y conditioning on x)
export_summs(m0,m1)
```

| | Model 1 | Model 2 |
|---|---|---|
| (Intercept) | -0.04 | -0.02 |
| | (0.03) | (0.01) |
| x1 | 0.01 | -1.20 *** |
| | (0.03) | (0.02) |
| x2 | 3.54 *** | 0.67 *** |
| | (0.03) | (0.05) |
| x | | 0.40 *** |
| | | (0.01) |
| N | 1000 | 1000 |
| R2 | 0.93 | 0.98 |

*** p < 0.001; ** p < 0.01; * p < 0.05.

x is significantly related to y but should not be included in the regression. It makes the irrelevant variable x1 now related to y significantly.

A related question, should we include non-significant variables? If x is an irrelevant variable (to y):

```r
x1 = rnorm(1000)
x2 = rnorm(1000)
# if x is really irrelevant but correlated with x1 and x2
x = 2.5*x1+3.5*x2+rnorm(1000)
y = 2.5*x1 +3.5*x2 + rnorm(1000)

m0 <- lm(y~x1+x2+x)
m1 <- lm(y~x1+x2)
export_summs(m0,m1)
```

The example demonstrated that if x is really not related to y (not caused by sampling or estimation in models), there are just some minor differences with or without including x. It is always safe the include all variables, because usually we don't know whether the insignificant variables are really non-relevant to y.

Nevertheless, there are some (minor) drawbacks by including really non-relevant variables: it will decrease $df$, increase $SE$! Not parsimony! If x is NOT correlated with x1 and x2, $SE$s will not change much (why?).

| | Model 1 | Model 2 |
|---|---|---|
| (Intercept) | -0.04 | -0.04 |
| | (0.03) | (0.03) |
| x1 | 2.53 *** | 2.54 *** |
| | (0.09) | (0.03) |
| x2 | 3.43 *** | 3.45 *** |
| | (0.12) | (0.03) |
| x | 0.00 | |
| | (0.03) | |
| N | 1000 | 1000 |
| R2 | 0.95 | 0.95 |

*** p < 0.001; ** p < 0.01; * p < 0.05.

One more example, x is indeed related to y, however, the effect size is small and thus it might be nonsignficant in the regression.

```
x1 = rnorm(1000,3,1)
x2 = rnorm(1000,5,2)
x = 3*x1+2*x2+5*rnorm(1000)
y = 2*x1 + 3*x2 + 0.02*x+ rnorm(1000,6,2.5)

m0 <- lm(y~x1+x2+x) # x is not significant but it is relevant
m1 <- lm(y~x1+x2) # removing the variable will bias the estimation
export_summs(m0,m1)
```

| | Model 1 | Model 2 |
|---|---|---|
| (Intercept) | 6.39 *** | 6.39 *** |
| | (0.33) | (0.33) |
| x1 | 1.98 *** | 2.02 *** |
| | (0.09) | (0.08) |
| x2 | 2.98 *** | 3.01 *** |
| | (0.05) | (0.04) |
| x | 0.01 | |
| | (0.02) | |
| N | 1000 | 1000 |
| R2 | 0.86 | 0.86 |

*** p < 0.001; ** p < 0.01; * p < 0.05.

**20. Should we control for mediators to estimate the treatment effect?**

- direct effects
- indirect effects
- total effects = direct + indirect effects

```
x = rnorm(1000,5,1)
m = 2*x+3*rnorm(1000)
y = 3*x+5*m+7*rnorm(1000,6,3)

# the traditional way of mediation test:
m0 <- lm(y~x)
m1 <- lm(y~x+m)
export_summs(m0,m1)
```

The tradition way of mediation test is to compare the regression models with/without the mediator (i.e., `m0`, `m1`). Without mediator in `m0`, the coefficient of x is 13.72, while it is 4.23 in `m1`, indicating the existence of partially mediation. This method is not precise, without estimation of the size and test of significance. It would be better to estimate using the *SEM* approach.

| | Model 1 | Model 2 |
|---|---|---|
| (Intercept) | 38.18 *** | 37.92 *** |
| | (3.95) | (3.26) |
| x | 13.72 *** | 4.23 *** |
| | (0.78) | (0.78) |
| m | | 4.75 *** |
| | | (0.22) |
| N | 1000 | 1000 |
| R2 | 0.24 | 0.48 |

*** p < 0.001; ** p < 0.01; * p < 0.05.

```
library(lavaan)
data = data.frame(y,x,m)
mod <- "# a path
         m ~ a * x

         # b path
         y ~ b * m

         # c direct path
         y ~ c * x

         # indirect and total effects
         ab := a * b
         total := c + ab"

fsem <- sem(mod, data = data, se = "bootstrap", bootstrap = 1000)
summary(fsem)
```

```
## lavaan 0.6-9 ended normally after 31 iterations
##
##   Estimator                                         ML
##   Optimization method                           NLMINB
##   Number of model parameters                         5
##
##   Number of observations                          1000
##
## Model Test User Model:
##
##   Test statistic                                 0.000
##   Degrees of freedom                                 0
##
## Parameter Estimates:
##
```

```
##   Standard errors                             Bootstrap
##   Number of requested bootstrap draws              1000
##   Number of successful bootstrap draws             1000
##
## Regressions:
##                    Estimate  Std.Err  z-value  P(>|z|)
##   m ~
##     x          (a)    2.000    0.089   22.395    0.000
##   y ~
##     m          (b)    4.747    0.231   20.528    0.000
##     x          (c)    4.228    0.817    5.174    0.000
##
## Variances:
##                    Estimate  Std.Err  z-value  P(>|z|)
##    .m                 8.875    0.389   22.802    0.000
##    .y               430.283   18.615   23.115    0.000
##
## Defined Parameters:
##                    Estimate  Std.Err  z-value  P(>|z|)
##     ab                9.494    0.618   15.350    0.000
##     total            13.722    0.821   16.723    0.000
```

The total effect is 13.72, which is consistent with `m0`. This is the overall treatment effect. `ab` is the indirect effect (the effect of x on y via m). It is possible that x could influence y via many mechanisms. Unless, you're interested in a particular mechanism, we estimate the treatment effect without controlling for mediators.