# Statistical simulation using R

# Why simulation?

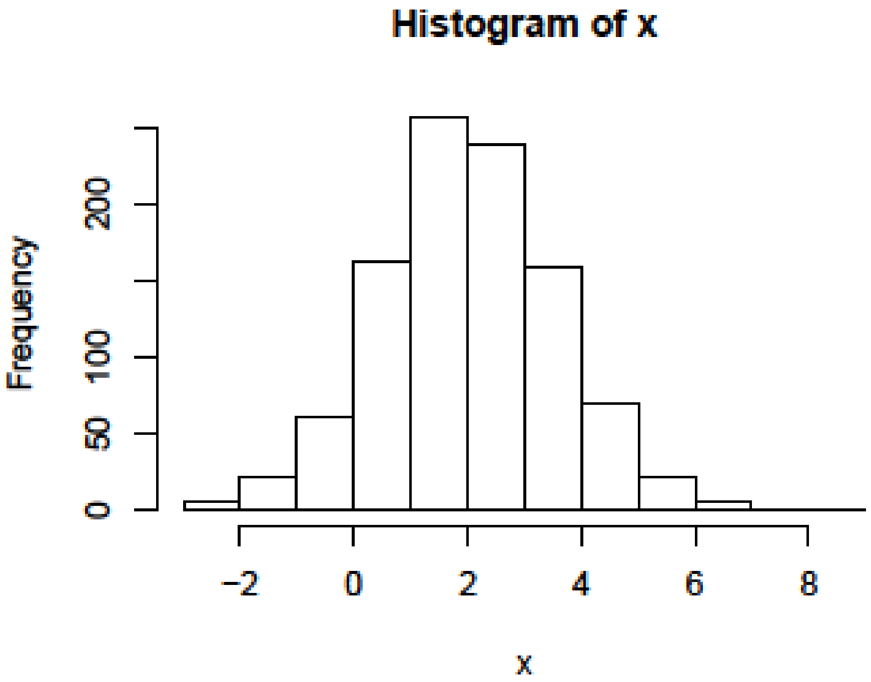
- Simulation is an important part of statistical research.
- Statistical theories/methods are all based on assumptions. So most theorems state something like “**if** the data follow these models/assumptions, **then** …”.
- The theories can hardly be verified in real world data because (1) the real data never satisfy the assumption; and (2) the underlying truth is unknown (no “gold standard”).
- In simulation, data are “created” in a well controlled environment (model assumptions) and all truth are known. So the claim in the theorem can be verified.

# Simulation functions in R

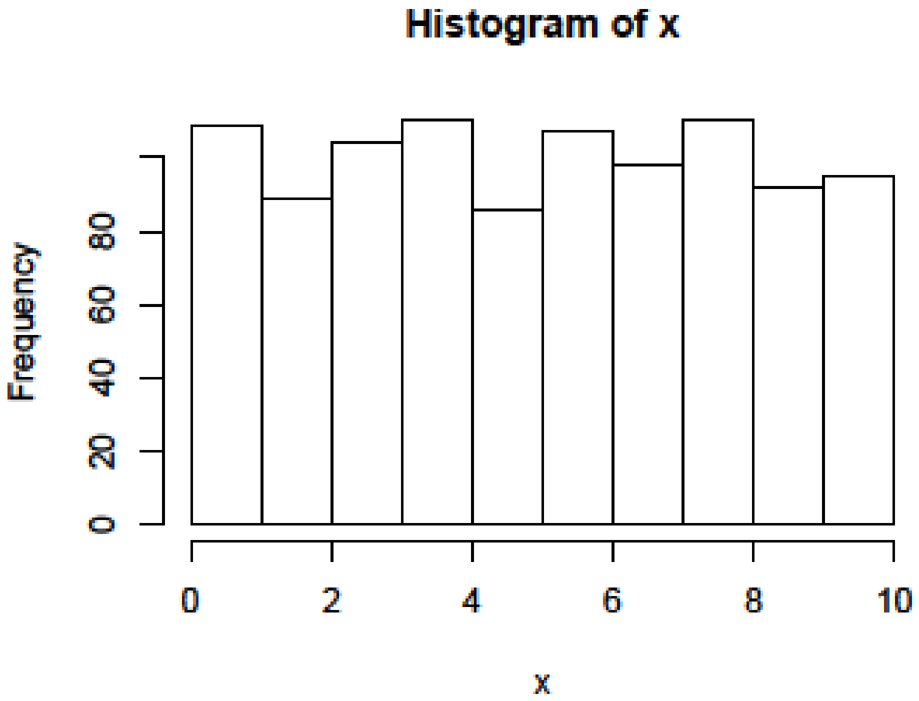
- The fundamental simulation functions are the random number generators (RNGs). The functions have names starting with “r”. Here is an (incomplete) list:
  - Continuous distribution: rnorm (Normal), rt (T), rf (F), rchisq (Chi-square), rexp (Exponential), runif (Uniform), rgamma (Gamma).
  - Discrete distribution: rpois (Poisson), rbinom (Binomial), rgeom(Geometric), rnbinom (Negative Binomial).
- For more complicated distributions, use MCMC method.
- Permutation function:
  - `sample`: take a sample from a population, with or without replacement.

# Examples of RNGs

```
> x=rnorm(1000, mean=2, sd=1.5)
> mean(x)
[1] 2.012532
> sd(x)
[1] 1.48
> hist(x)
```



```
> x=runif(1000, min=0, max=10)
> mean(x)
[1] 4.946862
> hist(x)
```



# Random number seed

- There is something called “seed” in random number generation. It specifies the “state” of the random number generators (RNG). Under the same seed the RNG will generate the same random numbers.
- It is always a good practice to specify and save the seed, so that the simulation results will be reproducible.
- You should also vary the seed to make sure you obtain similar results under different seeds.
- To set the seed, use `set.seed` function, for example,

```
> set.seed(123456)
```

# Steps in statistical simulation

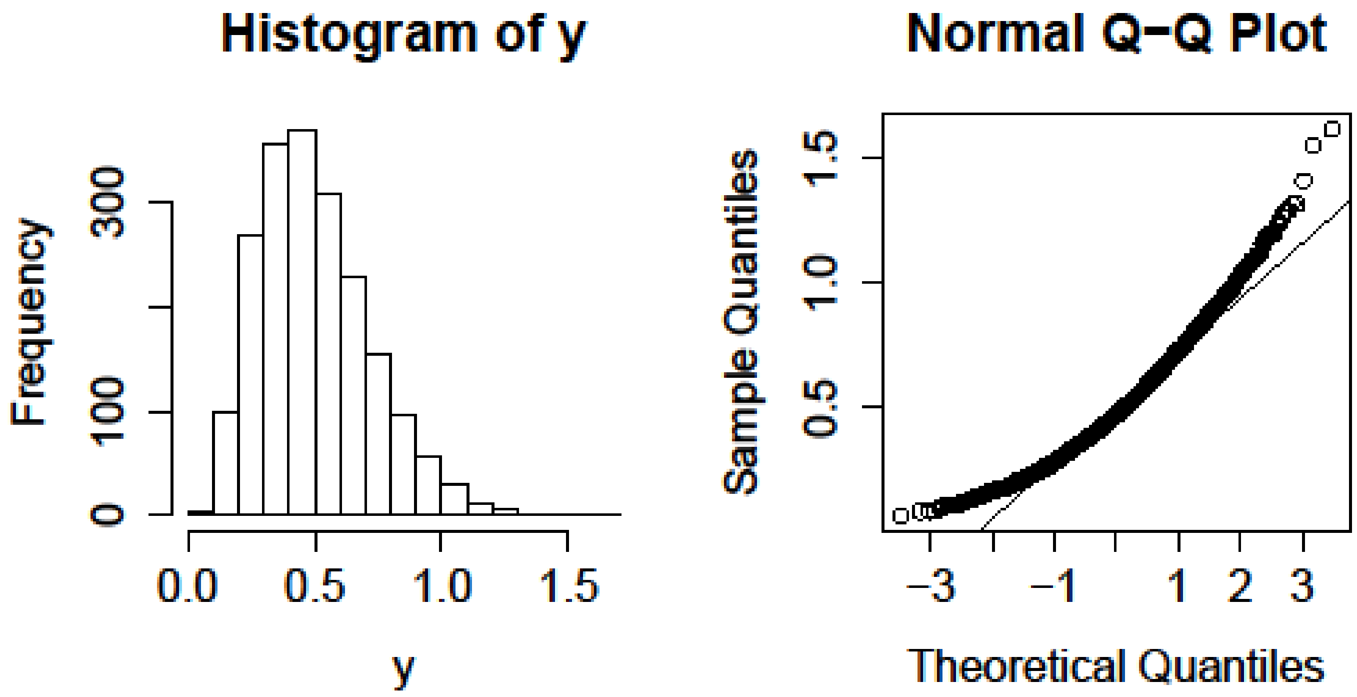
- Clearly write down the data generative model:
  - The model should include definition of variables, distribution assumptions, dependency of variables, etc.
  - Identify known constants (fixed), model parameters (fixed), and random variables (random).
- Follow the model to generate data (often in a step-wise, hierarchical way).
- Apply some procedure to the simulated data, obtain results (statistics, estimates, etc.).
- Compare results to the truth.

# An example: central limit theorem

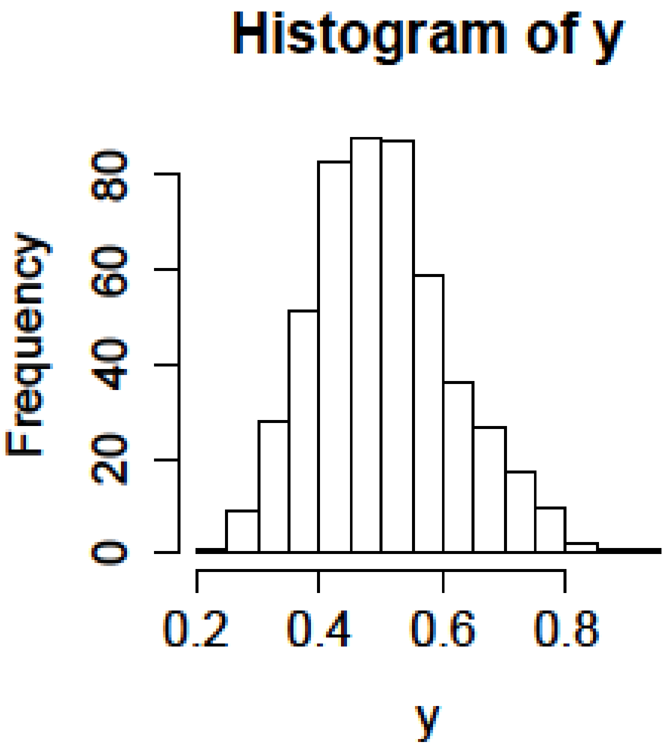
- **CLT**: mean of independent random variables, regardless of their distribution, converges to normal distribution.
- To verify this via simulation, it involves two setting:
  - Distribution of independent random variables.
  - Sample sizes.
- We can vary the parameters to understand the relationship of distribution and convergence rate.

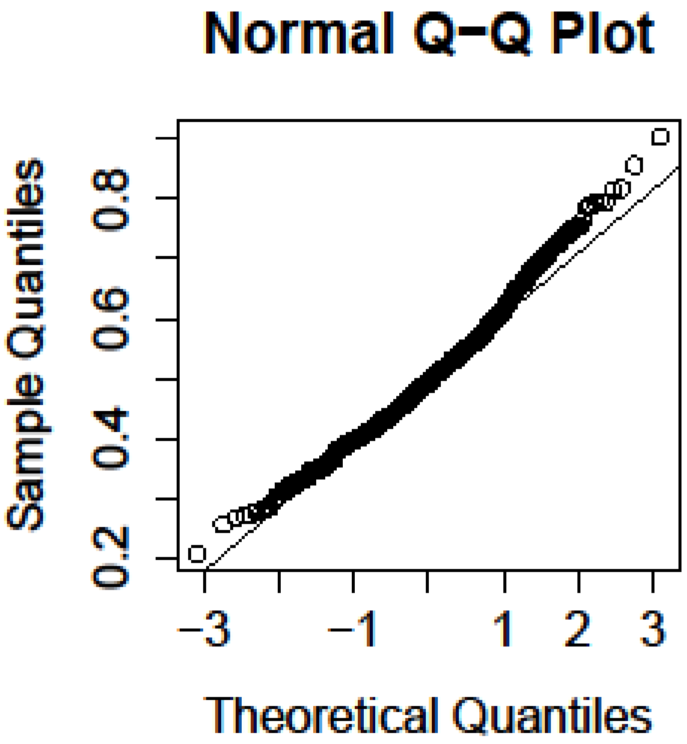
# Data is from exp(2), sample size is 5

```
X0=rexp(100000, rate=2)
n=5
X=matrix(X0, ncol=n)
y=rowMeans(X)
hist(y)
qqnorm(y); qqline(y)
```

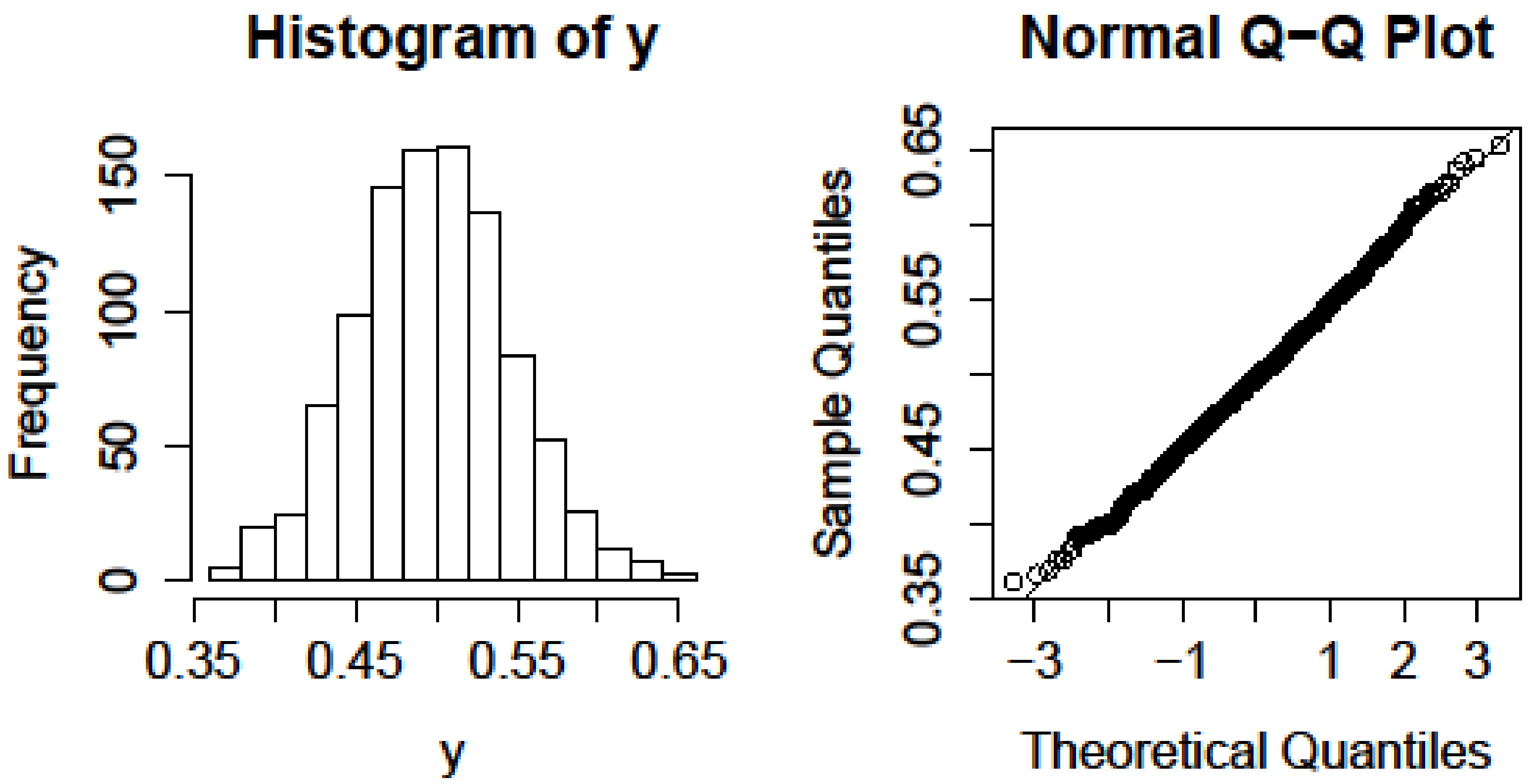


# Sample size is 20



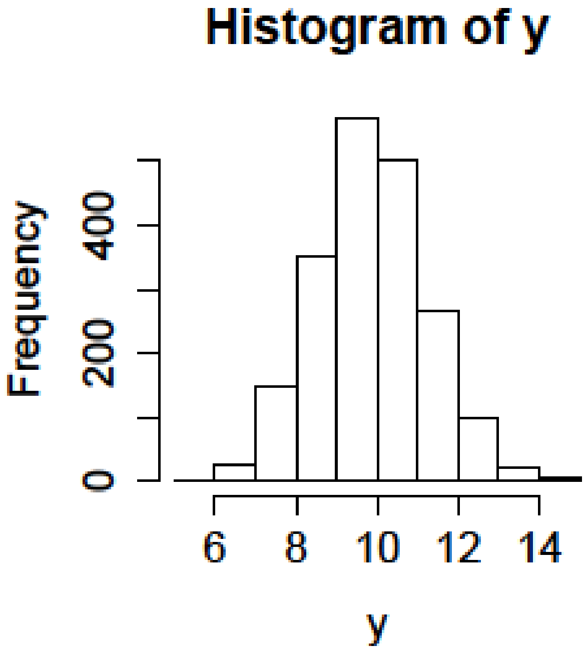


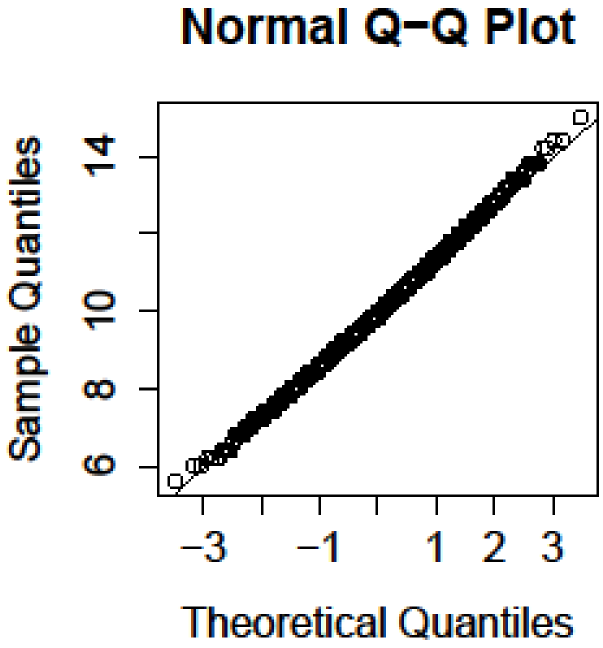
# Sample size is 100



# Data is from Poisson(10), sample size is 5

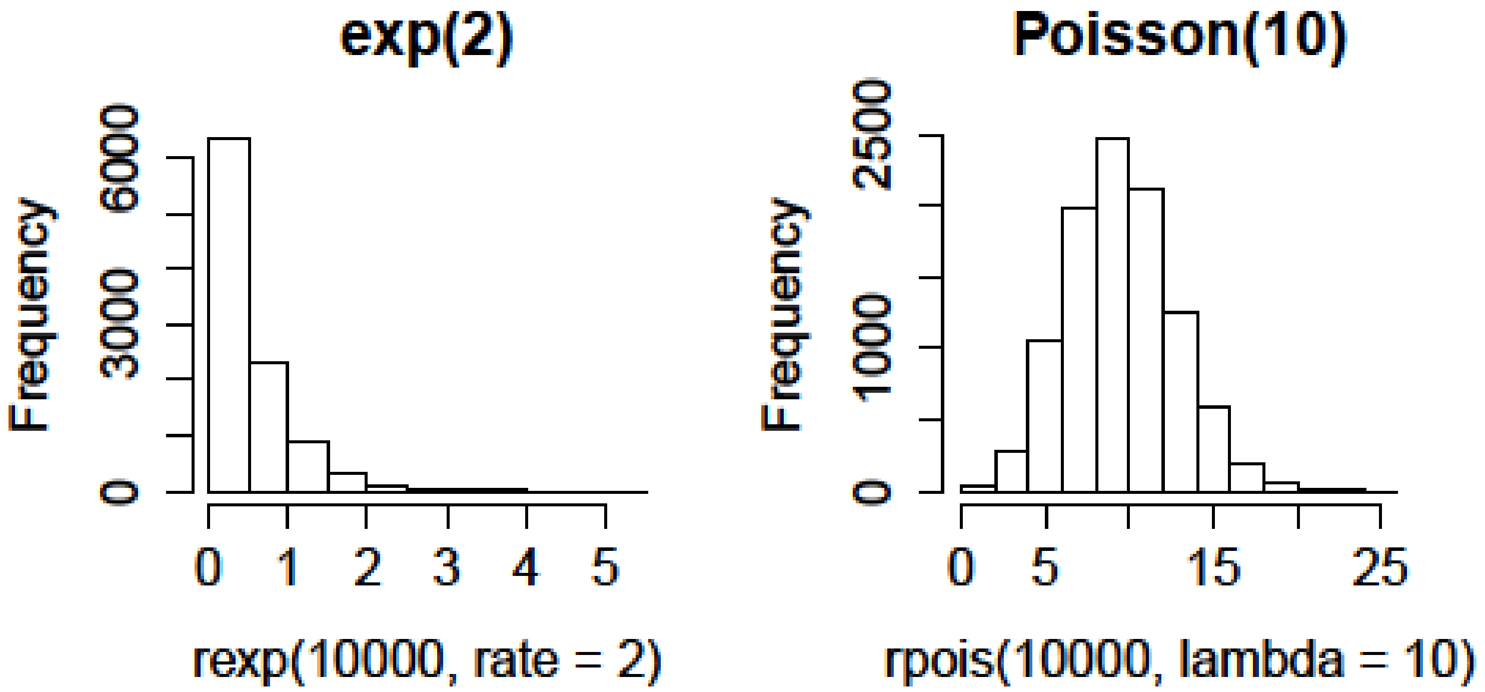
```
X0=rpois(10000, lambda=10)
n=5
X=matrix(X0, ncol=n)
y=rowMeans(X)
hist(y)
qqnorm(y); qqline(y)
```





# Compare exp(2) and Poisson(10)

- Poisson(10) is closer to normal, so it converges faster.



# Another example: Type I error

- In hypothesis testing, if the null hypothesis is true, result p-values should follow a uniform distribution.
- **Type I error** is defined as the incorrect rejection rate under true null hypothesis (false positive rate).
- For example:
  - Assume data are from Normal distribution: $X \sim N(\mu, 1)$
  - Null hypothesis is: $H_0 : \mu = 0$
  - If data are generated from true null, and the threshold of p-value for rejection is 0.05, the chance of incorrect rejection will be 5%.

# Simulation

- We can construct a simulation to verify the type I error under true null.
- Steps are:
  1. Randomly generate data from null.
  2. Perform hypothesis test, save p-value.
  3. Repeat steps 1&2 for a number of times.
  4. Compute the percentage of p-value less than the threshold.

```
nsims=1000
pvals=rep(0, nsims)
for(i in 1:nsims) {
  x = rnorm(10)
  pvals[i] = t.test(x)$p.value
}

> mean(pvals<0.05)
[1] 0.045
```

# Another example: linear regression

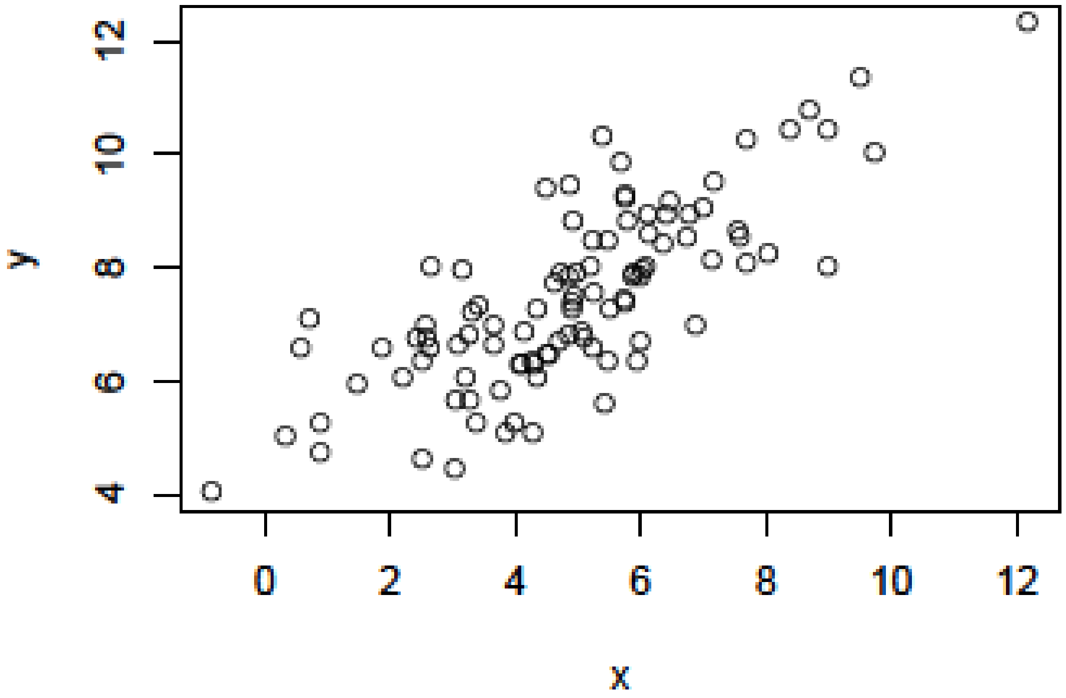
- Assume data follow a linear model:

$$x_i \sim N(5,4); \varepsilon_i \sim N(0,1); i = 1,\ldots,100$$

$$y_i = b_0 + b_1 x_i + \varepsilon_i$$

- In the model:
  - Constants are: sample size (100), distribution of *X* and *ε*.
  - Parameters are $b_0$ and $b_1$.
  - Data need to be generated are *X* and *Y*.
- Steps for generating data:
  - Specify the constants and parameters.
  - Generate *X* and *ε*.
  - Generate Y based on *X* and *ε*.

```
n=100
b0=5; b1=0.5
x=rnorm(n, mean=5, sd=2)
eps=rnorm(n, mean=0, sd=1)
y=b0+b1*x+eps
plot(x, y)
```



# Estimated regression coefficient

- So far we only generated the data. The simulated data can be used to demonstrate some theoretical results.
- For example, in linear regression, the OLS estimates for $b_1$ is:

$$\hat{b}_1 = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2}$$

- The estimates follow Normal distribution asymptotically with:

$$E[\hat{b}_1] = b_1,\ var(\hat{b}_1) = \frac{\sigma^2}{\sum_i (x_i - \bar{x})^2}$$

# Steps for simulation

- We can verify these theoretical results through the following simulation:
  1. Specify and fix the parameters $b_0$ and $b_1$.
  2. Generate and fix $X$;
  3. Generate $\varepsilon$ and then $Y$, then estimate $b_{1.}$
  4. Repeat step 3 for a certain times (such as 1000), then compute the mean and variance of the estimates.
  5. Compare with the theoretical results.

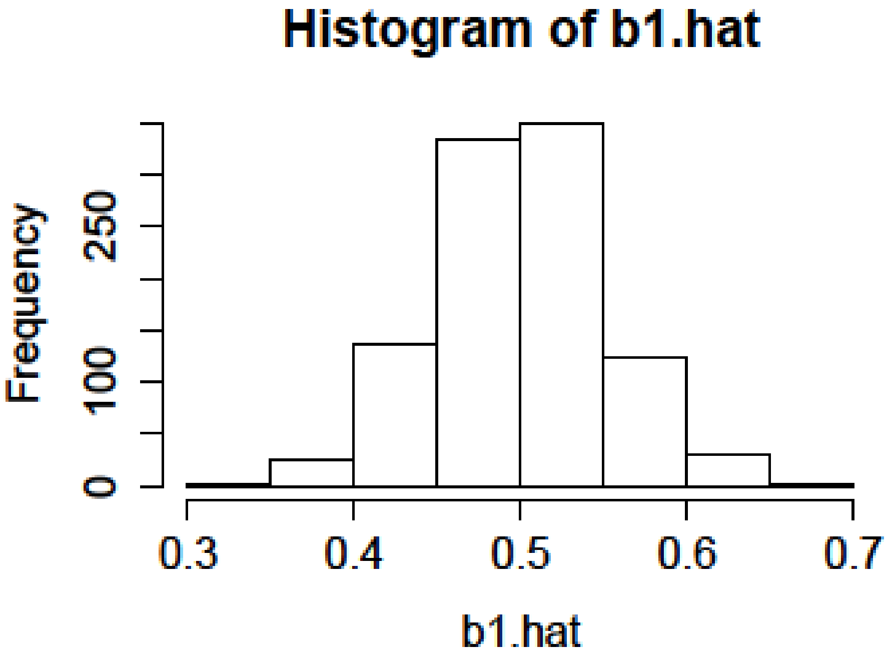
```
simData=function(b0=5, b1=0.5){
  eps=rnorm(n, mean=0, sd=1)
  b0+b1*x+eps
}

estB1=function(X, Y) {
  a=X-mean(X)
  b=Y-mean(Y)
  sum(a*b)/sum(a^2)
}

nsims=1000
n=100
x=rnorm(n, mean=5, sd=2)
b1.hat=numeric(1000)
for(isim in 1:nsims) {
  y=simData()
  b1.hat[isim]=estB1(x, y)
}
```

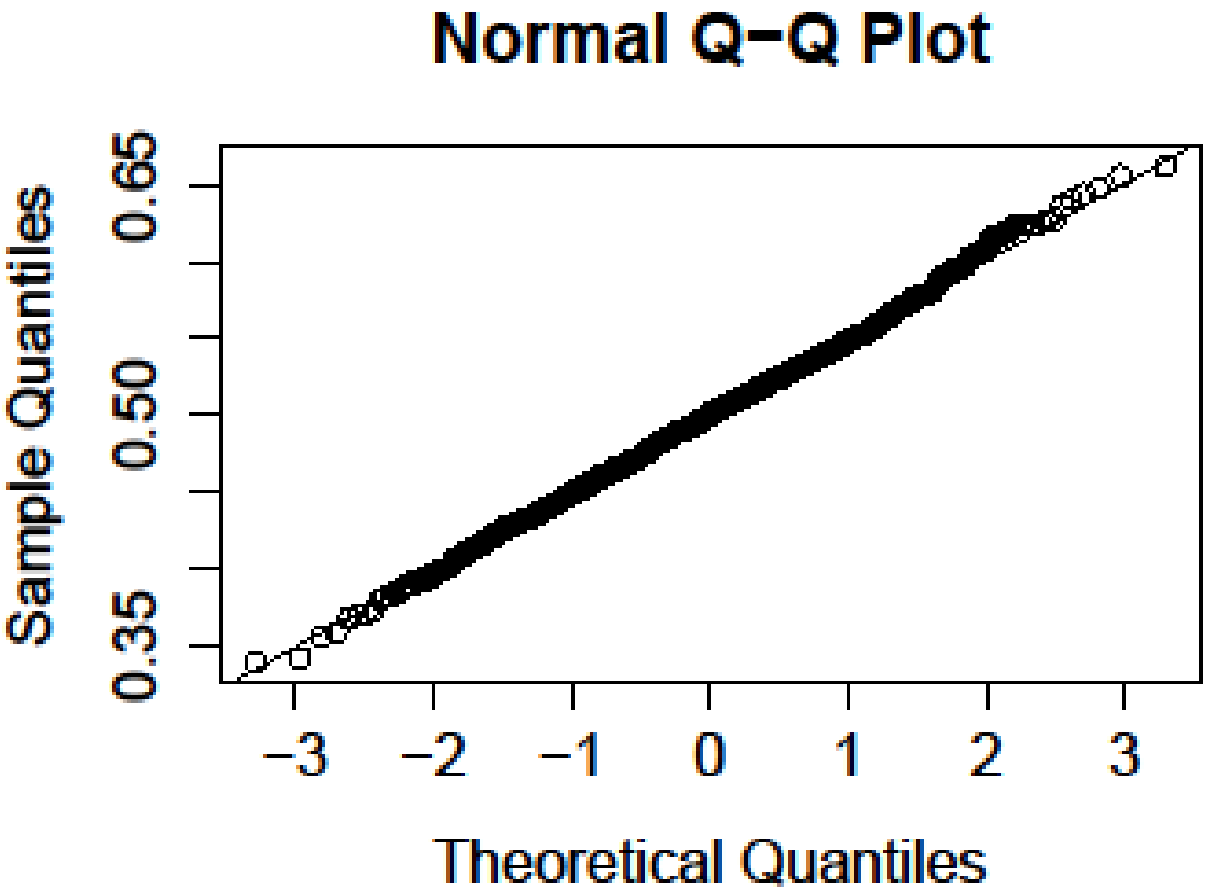
# Results

```
> mean(b1.hat) ##
[1] -0.0001239591
> var(b1.hat) ## empirical variance of estimated b1
[1] 0.002622967
> 1/sum((x-mean(x))^2) ## ## theoretical variance of the estimated b1.
[1] 0.00260201
> hist(b1.hat)
```



## Show normality of the estimates

```
> qqnorm(b1.hat)
> qqline(b1.hat)
```



# Tips for doing good simulation

- Run for multiple times and average the results.
- The simulated data should be as similar to the real data as possible. If possible, the simulation parameters should be obtained from real data.
- Set and save the random seed, so that the results are reproducible.

# Tips (cont.)

- Programming wise, I recommend:
  - Write the procedure for generating data in a functions with all parameters as arguments, so that the data can be re-generated in one line and the model parameters can be easily changed.
  - Write the method part in another function(s). It takes the results object from the simulation function and produce necessary results.

# Promotion for my other classes

- Bioconductor: http://web1.sph.emory.edu/users/hwu30/teaching/bioc/bioc.html
  - Very useful skills for analyzing high-throughput genetics/genomics data.
- Advanced statistical computing: http://web1.sph.emory.edu/users/hwu30/teaching/statcomp/statcomp.html
  - Must-know computing methods for all statisticians.
  - A little advanced.