Simulation
(An Introduction)

1. Introduction

2. Random Number Generator

3. Simulation Studies in Statistics
   - Simulation: Comparing Estimators of Mean
   - Simulation: Coverage Probability of Confidence Intervals
   - Simulation: Properties of Hypothesis Tests
   - END

1 Introduction

2 Random Number Generator

3 Simulation Studies in Statistics
- Simulation: Comparing Estimators of Mean
- Simulation: Coverage Probability of Confidence Intervals
- Simulation: Properties of Hypothesis Tests
- END

Definition

$\bar{X}$: $X_1, \ldots, X_n$ iid $\sim$ distn with μ; var $= \sigma^2$.

$\Rightarrow$ $\bar{X} \sim N\left(\mu; \frac{\sigma^2}{n}\right)$ if n is large enough.

- Simulation (in statistics) is using artificially generate data in order to test out a hypothesis or statistical method.
- Advantages of simulation: it is cheap; it is much faster than traditional data collection; in good conditions the results of simulation can approximate real results.
- Disadvantage: the results from simulation can approximate the real-results only, not the real results.

Example 1: The t-distribution (Theory)

t_{df} ...

Definition

If $Z \sim N(0,1)$ and $U \sim \chi^2_k$ and Z and U are independent, then the distribution of $Z/\sqrt{U/k}$ is called the t **distribution** with k degrees of freedom.

$\frac{Z}{\sqrt{U/k}} : t$

- Let $X_1, ..., X_n$ be a sample of n independent $N(\mu, \sigma^2)$ rv.
- Let $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$ (sample mean).
- Let $S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2$ (sample variance).
- Then, it can be proved that:

ST2131

$$\frac{\bar{X}-\mu}{S/\sqrt{n}} \sim t_{n-1}$$

Example 1: The t-distribution (Simulation)

$N \to ((X_1, \ldots, X_n)) \sim N(0, \sigma^2) \to \bar{X}, S$

$n = 9 \to t_8$

$\rightarrow \boxed{\dfrac{\bar{X} - 0}{S/3}} \to t$

- A sample of size $n = 9$ is taken from $N(0, \sigma^2)$.

- Consider the statistic

$\mu = 0$

$\bar{X}$; S for each set/sample.

$$t = \frac{\bar{X} - \mu}{S/\sqrt{9}} \sim t_8$$

$N \to t = \bar{X}/(S/3)$ for each set

How do we "view" this result by simulation? The steps below can help.

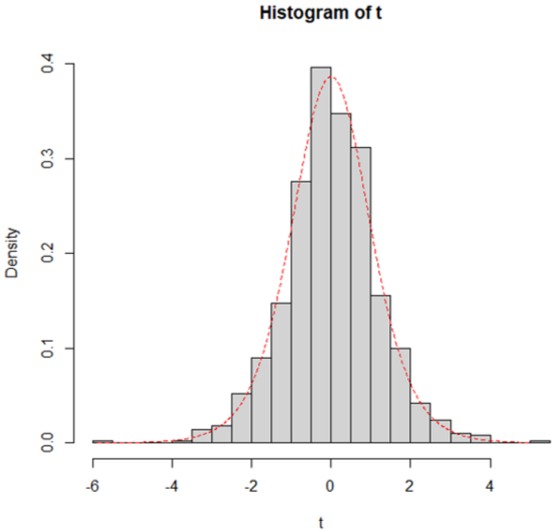
1. Generate N random samples (N can be any number, better to be a large one, like 1000), each sample of size $n = 9$ (fixed) from a normal distribution with mean 0 (fixed), and a (self random chosen) standard deviation σ. $= 2$

2. For each sample, compute the statistic t (and must save all the values/realizations of t).

3. Construct a histogram for the N realizations of the statistic t.

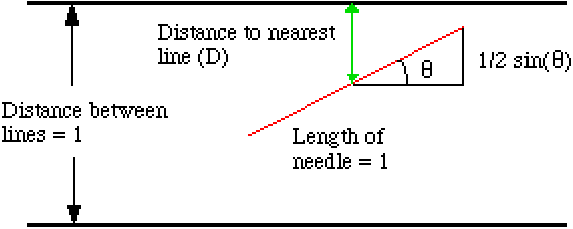
The t-distribution (Simulation)

The histogram of N realizations of t and the plot of t_8 distribution (in red).



Example 2: Buffon's Needle Experiment

- A needle of length l is thrown randomly onto a grid of parallel lines with distance $d > l$.



- **Question**: What is the probability that the needle intersects a line?
- The answer is $\dfrac{2l}{\pi d}$. Can we get the answer through simulations?

Example 2: Buffon's Needle Experiment

Main idea of the solution by experiment:

- Simulate the throwing of a needle into a grid of parallel lines, say N times.
- Count the number of times the needle intersects a line, say n times.
- Then n/N gives an estimate of the probability that the needle intersects a line.
- A website gives the visualization of the experiment

  `http://www.metablake.com/pi.swf`

Example 2: Buffon's Needle Experiment

How to get the solution by simulation?

1. Generate the position and the inclination of the needle.
   - Generate a random number, x, from $U(0, d/2)$. This number represents the location of the center of the needle.
   - Generate a random number, q, from $U(0, \theta)$. This number represents the angle between the needle and the parallel lines.

2. Check if the needle cuts a line.
   - The needle cuts a line if $x < l/2 \sin(\theta)$.
   - Create a variable $w = 1$ if $x < l/2 \sin(\theta)$ and 0 otherwise.

3. Repeat Steps 1 and 2 a total of N times.

4. Count the number of times $w = 1$, let say n times. An estimate of the probability of the needle intersecting a line is given by n/N.

Example 3: Comparing Estimators

sample mean
median
trimmed mean
win. mean

We have learnt about several robust estimators of location.

Question: Which estimator is better, the trimmed mean or the Winsorized mean?

Secondary Questions:

- Under what conditions (in terms of the underlying distributions) is the estimator better?
- What criteria to determine the performance of the estimator? Ans: in our course, we can use the Mean Square Error ($\text{MSE} = \text{Bias}^2 + \text{Variance}$).

It may be intractable to compute the (theoretical) MSE for each of the estimators under different underlying distributions.

Simulation is an alternative solution.

Question: How to design such a simulation study?

1 Introduction

2 Random Number Generator

3 Simulation Studies in Statistics
- Simulation: Comparing Estimators of Mean
- Simulation: Coverage Probability of Confidence Intervals
- Simulation: Properties of Hypothesis Tests
- END

A Set of Random Integer Numbers

- Computer can help us to choose a set of 10 random integer numbers, range from 1 to 100.

```
> sample(1:100,10,replace=T) #or
 [1] 41 59 64  7 41 18 76 85 40 29
> sample(1:100,10,replace=F)
 [1]  6 83 32 55 42 75 47 25 46 28
```

- That means, in the set of integers from 1 to 100, (1) it will randomly choose an integer. (2) if "replace = T", it will put that number back to the series of 1 to 100 and then choose the second random number. This process continues till the 10th integer is chosen. This allows the output to have repeating integers.

- If "replace = F", that means the chosen number will be removed from the series of 1 to 100, hence the output will have no repeating number.

- That sound simple!

Pseudo-random Numbers

Now, we want to generate a set of random numbers that follow Uniform(0,1) distribution. How to do that?

Definition

A sequence of uniform pseudo-random numbers $\{U_i\}$ is a deterministic sequence of number in $[0, 1]$ having the same relevant statistical properties as a sequence of random numbers.

- In R, we generate a set of 6 random numbers from $\sim Unif(0, 1)$ by

```
> runif(6,0,1)
[1] 0.08988458 0.91487180 0.80281471 0.24061284 0.68319329 0.495
```

rexp(   )

- In Python, we generate a set of 6 numbers that are iid $\sim Unif(0, 1)$ by

```
> # import numpy as np
> # x = np.random.uniform(0,1,6)
> # print(x)
```

- What's the logic behind when the random numbers above were generated?

Congruential Generators

- **Congruential generators** are defined by

$$X_{n+1} = (aX_n + c) \mod m$$

for a multiplier a, shift c, and modulus m. Here a, c and m are all integers.

- To initialize, we have to provide a seed $(n = 0)$ which is X_0.
If $c = 0$, generators having the form

$$X_{n+1} = aX_n \mod m$$

are called **multiplicative congruential generator**. If $c > 0$, they are called **linear congruential generator**.

- Uniform random numbers are obtained by

$$U_i = X_i/m$$

Some Properties of Congruential Generators

- The $m + 1$ values $\{X_0, X_1, ..., X_m\}$ cannot be distinct and at least one value must occur twice, as X_i and X_{i+k}, say.

  item $\{X_i, X_{i+1}, ..., X_{i+k-1}\}$ is repeated as $\{X_{i+k}, X_{i+k+1}, ..., X_{i+2k-1}\}$.

- The sequence $\{X_i\}$ is periodic with period $k < m$.
- For multiplicative generators, the maximal period is $m - 1$.
- If 0 ever occurs, it is repeated indefinitely.
- One of our primary objectives is to use a generator with as large period as possible, however that does not mean that a large period can help to guarantee a good generator.
- The values commonly used for parameters are: $m = 2^{32}$ or $m = 2^{31} - 1$, etc. $c = 1$ or $c = 0$ or $c = 12345$, etc. $a = 8121$ or $a = 22695477$, etc.

R function for Congruential Generators

- Creating a set of 10 random numbers that are from $Unif(0,1)$.

```
> cg <- function(n,a,m,c,x0){
+ series <- NULL
+ for (i in 1:n) {
+ x1 <- (a*x0+c)%%m
+ x0 <- x1
+ series <- c(series,x1/m) }
+ return(series) }
> cg(10,397204094,2^31-1,0,1234)
 [1] 0.24381116 0.08947511 0.38319371 0.72800325 0.72792771 0.175038
 [7] 0.40680994 0.90923700 0.34271140 0.55960111
```

Generate Uniform Random Numbers

We want to generate a set of n random numbers from $Unif(a, b)$ or $U(a, b)$ for short.

- In R:
  runif(n,a,b).
- In Python:
  import numpy as np
  np.random.uniform(a,b,n)
- In SAS:

set. seed (999)

```
data Ugen;
call streaminit(1234); /* 1234 is used as the seed, this helps the generated
random numbers reproducible */
do i = 1 to 10;   n= 10
    x = rand('uniform', 2, 3);
    output;
end;
keep x;
run;
proc print data=Ugen;
var x;
run;
```

a= 2   b= 3

10 values ~ U (2,3)

Generating Non-uniform Random Numbers: Inversion Method

cdf $F \to F^{-1}$

The theory of the inversion method is as below:

- If random variable X has a continuous distribution function (cdf) $F(x) = P(X \leq x)$ then a variable Y which is defined as $Y = F(X)$ will follow $U(0,1)$.

- If a variable $Y \sim U(0,1)$ and variable X has cdf F then the cdf of the variable $F^{-1}(Y)$ is F

That gives us a way to generate random numbers following a distribution function $F(x)$.

$X \sim$ expo has cdf: F $\quad F^{-1}$

10 values from exp (mean = 5)

rexp (10, rate = 1/5)

10 values

1. Generate U from $U(0,1)$
2. Set $X = F^{-1}(U)$ provided the inverse exists. Then X is a random number from the distribution F.

Exponential Distribution

Theoretically,

- If X follows an exponential distribution with parameter λ (rate) (which then has mean $1/\lambda$), then its cdf is

$$\boxed{F(x) = 1 - e^{-\lambda x}}$$

- Let $U \sim U(0,1)$, then $X = F^{-1}(U)$ will follow exponential distribution, where

$$F^{-1}(U) = \boxed{-\lambda \log(1 - U)}$$

Hence, to generate a random number that follows exponential distribution with $\lambda = 5$, we'll:

1. Randomly generate U from $U(0,1)$
2. Let $X = -5 \log(1 - U)$.
   Then X is a random number from the exponential distribution with rate $\lambda = 5$.

Exponential Distribution

Generate n random numbers from $Exp(\lambda)$ where λ is the rate:

- In R,
  rexp(n, λ)
- In Python,
  import numpy as np
  np.random.exponential($1/\lambda, n$)
- In SAS,

```
data Egen;
call streaminit(1234);
do i = 1 to 10; * here we use n = 10;
   x = rand('exponential', 1/5); *rate lambda = 5;
   output;
end;
keep x;
run;
proc print data=Egen;
var x;
run;
```

Weibull Distribution (Standard version)

- If a variable X follows a Weibull distribution with one parameter of shape α with its pdf
$$f(x) = \alpha x^{\alpha-1} e^{-x^{\alpha}}, \quad x > 0,$$
then its cdf is
$$F(x) = 1 - e^{-x^{\alpha}}, \quad x > 0.$$
(The standard version has the scale parameter fixed at 1)

- The inverse function of $F(x)$ is $F^{-1}(x) = (-log(1-x))^{1/\alpha}$.

- If we want to generate a random number that follows Weibull distribution, we can:
  1. Randomly generate U from $U(0,1)$
  2. Let $X = (-log(1-U))^{1/\alpha}$.
     Then X is a random number from the standard Weibull distribution with shape α.

Generating a Weibull Random Variable

We'll generate 10 random numbers that follow Weibull distribution with shape $\alpha = 4$.

- In R,
  ```
  x = rweibull(10, shape = 4 )
  ```
- In Python,
  ```
  import numpy as np
  x = np.random.weibull(4, 10)
  print(x)
  ```
- In SAS,
  ```
  data Weibullgen;
  call streaminit(1234);
  n = 10;
  do i = 1 to n;
  x = rand('weibull',4);
  output; end;keep x;run;
  ```

Generating a Normal Random Variable

cdf of Normal
pdf : $\frac{1}{2\pi\sqrt{\ }} \exp\{(x-\mu) \ldots$

- Normal distribution does not have cdf with explicit form, hence we cannot apply the inversion method to generate a random number following normal distribution from a uniform number.

- A random number that follows standard normal distribution is generated by an indirect method as below.
  1. Generate u_1 and u_2 from $U(0,1)$.
  2. Set $\theta = 2\pi u_1$.
  3. Set $R = (-2\log u_2)^{1/2}$.
  4. Set $X = R\cos(\theta)$ and $Y = R\sin(\theta)$.
  5. X and Y are independent standard normal variables.
  6. For simplicity, only X or Y is used.

Generating a Normal Random Variable

To generate n random numbers that follow normal distribution with mean μ and standard deviation σ, we do:

- In R

  ```
  n = 10; mu = 5; sigma = 1.5
  x = rnorm(n, mean = mu, sd = sigma)
  ```

- In Python,

  ```
  import numpy as np
  n = 10
  mu = 5
  sigma = 1.5
  x = np.random.normal(loc = mu, scale = sigma, size = n)
  ```

- In SAS,

  ```
  data Normalgen;
  call streaminit(1234);
  n = 10; mu = 5; sigma = 1.5;
  do i = 1 to n
  x = rand('normal',mu,sigma);
  output; end; keep x; run;
  ```

Distributions that related to Normal distribution

Certain random variables are related with the Normal distribution and can be generated using those relationships.

- Cauchy distribution: If Y and Z are independent and $Y \sim N(\mu, \sigma^2)$ while $Z \sim N(0,1)$, then $X = Y/Z$ follows a Cauchy (μ, σ^2) distribution.

G CP

Topic 6

- χ_1^2 distribution: if $Y \sim N(0,1)$ then $X = Y^2$ follows χ_1^2 distribution.
- χ_n^2 distribution: if $Y_1, Y_2, ..., Y_n$ are **independent** identically distributed $N(0,1)$, then

$$X = Y_1^2 + Y_2^2 + ... + Y_n^2 \sim \chi_n^2$$

- Student's t-distribution with p degrees of freedom t_p: If $Y \sim N(0,1)$ and $Z \sim \chi_p^2$, then $X = Y/\sqrt{Z/p} \sim t_p$.

$\boxed{10 \text{ values} \sim t_{20}}$ → rt(10, df=20)

- F-distribution $F_{m,n}$: if $Y_1 \sim \chi_m^2$, $Y_2 \sim \chi_n^2$ and they are independent, then

$$X = \frac{Y_1/m}{Y_2/n} \sim F_{m,n}.$$

Generating χ^2 distribution

We'll generate 10 random numbers that follow χ^2_3.

- In R.
  ```
  x = rchisq(10, df = 3)
  ```
- In Python,
  ```
  import numpy as np
  x = np.random.chisquare(df = 3, size = 10)
  print(x)
  ```
- In SAS.
  ```
  data Chisqgen;
  call streaminit(1234);
  n = 10; df = 3;
  do i = 1 to n;
  x = rand('chisq',df);
  output; end;keep x;run;
  ```

Generating Binomial Distribution

We'll generate 10 random numbers that follows $Bin(100, p = 0.3)$.

- In R,
  ```
  x = rbinom(n = 10, size = 100 ,prob = 0.3)
  ```
  Note: argument 'size' is the number of trials of the Binomial distribution.

- In Python,
  ```
  import numpy as np
  x = np.random.binomial(n = 100, p = 0.3, size = 10)
  ```
  Note: In Python, argument 'n' is the number of trials of the distribution while argument 'size' is the number of observations that we need to generate.

- In SAS,
  ```
  data Bingen;
  call streaminit(1234);
  p = 0.3; n = 100; *n = number of trials of the distribution;
  do i = 1 to 10; *10 is the number of the observations that we want to
  generate;
  x = rand('binom',p,n);
  output; end;keep x;run;
  ```

Generating Poisson Distribution

We'll generate 10 random numbers that follow $Poi(3)$, with mean $\lambda = 3$.

- In R,
```
rpois(10, 3)
```

- In Python,
```
import numpy as np
x = np.random.poisson(lam = 3, size = 10)
print(x)
```

- In SAS,
```
data Poisgen;
call streaminit(1234);
do i = 1 to 10; *n = 10;
x = rand('poisson',3); *lambda = 3;
output; end;keep x;run;
```

1. Introduction

2. Random Number Generator

3. Simulation Studies in Statistics
   - Simulation: Comparing Estimators of Mean
   - Simulation: Coverage Probability of Confidence Intervals
   - Simulation: Properties of Hypothesis Tests
   - END

Purposes of using Simulation at our level

- Study properties of estimators.

biasness of an estimator
var of " " ".

$\rightarrow \bar{X} \Rightarrow \mu$

estimator

- Study properties of hypothesis testing procedures.

sig. level α.
type of error $\rightarrow$ type 1 | error
$\rightarrow$ type 2 |
$\rightarrow$ power of a test.

What constitutes a simulation study in statistics?

- Simulation involves numerical techniques performed on computers.
- It involves random sampling from probability distributions.

Some issues that matter in Statistics

- Is an estimator biased in finite samples? Is it still consistent under departures from assumptions? What is its sampling variance?

Some issues that matter in Statistics

$\mu \leftarrow \bar{X}$

$q_{0.5}$

$\bar{X}_{0.1}$

- Is an estimator biased in finite samples? Is it still consistent under departures from assumptions? What is its sampling variance?
- How does it compare to competing estimators on the basis of bias, precision, etc.?

Some issues that matter in Statistics

- Is an estimator biased in finite samples? Is it still consistent under departures from assumptions? What is its sampling variance?
- How does it compare to competing estimators on the basis of bias, precision, etc.?
- Does a procedure for constructing a confidence interval for a parameter achieve the advertised nominal level of coverage? Like, 95% of the the built 95% CIs cover the true parameter?

Some issues that matter in Statistics

- Is an estimator biased in finite samples? Is it still consistent under departures from assumptions? What is its sampling variance?
- How does it compare to competing estimators on the basis of bias, precision, etc.?
- Does a procedure for constructing a confidence interval for a parameter achieve the advertised nominal level of coverage? Like, 95% of the the built 95% CIs cover the true parameter?
- Does a hypothesis testing procedure attain the advertised level of significance or size?

Some issues that matter in Statistics

- Is an estimator biased in finite samples? Is it still consistent under departures from assumptions? What is its sampling variance?
- How does it compare to competing estimators on the basis of bias, precision, etc.?
- Does a procedure for constructing a confidence interval for a parameter achieve the advertised nominal level of coverage? Like, 95% of the the built 95% CIs cover the true parameter?
- Does a hypothesis testing procedure attain the advertised level of significance or size?
- If it does, what power is possible against different alternatives to the null hypothesis? Do different test procedures deliver different power?

Some issues that matter in Statistics

- Is an estimator biased in finite samples? Is it still consistent under departures from assumptions? What is its sampling variance?
- How does it compare to competing estimators on the basis of bias, precision, etc.?
- Does a procedure for constructing a confidence interval for a parameter achieve the advertised nominal level of coverage? Like, 95% of the the built 95% CIs cover the true parameter?
- Does a hypothesis testing procedure attain the advertised level of significance or size?
- If it does, what power is possible against different alternatives to the null hypothesis? Do different test procedures deliver different power?
- To answer these questions in the absence of analytical results, we can use simulation.

Monte Carlo Simulation Approximation

$\mu \leftarrow \bar{X} \leftarrow$ distn of $\bar{X} \leftarrow$ some conditions

- When trying to understand a particular parameter, we try to understand the properties of its estimator.

$\bar{X} \sim \boxed{N(\mu;\ \sigma^2/n)}$ n

- The estimator has a true sampling distribution under some conditions about sample (like finite size, etc) and conditions about population (like population follow some specific distribution, etc).

- We want to know this true sampling distribution in order to address the issues mentioned in the previous slide.

- However, very often the true sampling distribution is not possible.

- Hence, we approximate the sampling distribution of an estimator under a particular set of conditions.

Steps of MC Simulation for Approximation

samples

- Generate M independent datasets under the determined conditions.

Steps of MC Simulation for Approximation

N samples → $\bar{x}$ for each sample.

M

- Generate M independent datasets under the determined conditions.

- Compute the numerical value of the estimator/statistic T from each dataset. As such, we have $T_1, ..., T_M$.

Steps of MC Simulation for Approximation

- Generate M independent datasets under the determined conditions.

- Compute the numerical value of the estimator/statistic T from each dataset. As such, we have $T_1, ..., T_M$.

- If M is large enough, the set of simulated estimators $T_1, ..., T_M$ should be good approximations to the true properties of the estimator or test statistic.

M

1. Introduction

2. Random Number Generator

3. Simulation Studies in Statistics
   - Simulation: Comparing Estimators of Mean
   - Simulation: Coverage Probability of Confidence Intervals
   - Simulation: Properties of Hypothesis Tests
   - END

General Ideas

Consider an estimator T for a parameter θ. $= \mu$; $T = \bar{X}$

M samples, each sample → get $\bar{X}$ → M, $\bar{X}$

- T_m is the value of T from the m-th dataset, $m = 1, ..., M$.

General Ideas

Consider an estimator T for a parameter θ.

- T_m is the value of T from the m-th dataset, $m = 1, ..., M$.
- The mean of all T_m's is an estimate of the true mean of the sampling distribution of the estimator T.

General Ideas

Consider an estimator T for a parameter θ.

- T_m is the value of T from the m-th dataset, $m = 1, ..., M$.
- The mean of all T_m's is an estimate of the true mean of the sampling distribution of the estimator T.
- Simulation allows us to study the properties of the estimator T when the theoretical approach is difficult or untractable.

Examples: Estimators of μ

- Say we are interested to compare three estimators for the mean μ of a population distribution based on a random sample $Y_1, Y_2, ..., Y_n$. The 3 estimators are:

Examples: Estimators of μ

- Say we are interested to compare three estimators for the mean μ of a population distribution based on a random sample $Y_1, Y_2, ..., Y_n$. The 3 estimators are:
  - Sample mean, T^1

Examples: Estimators of μ

- Say we are interested to compare three estimators for the mean μ of a population distribution based on a random sample $Y_1, Y_2, ..., Y_n$. The 3 estimators are:
  - Sample mean, T^1
  - Sample median, T^2

Examples: Estimators of μ

- Say we are interested to compare three estimators for the mean μ of a population distribution based on a random sample $Y_1, Y_2, ..., Y_n$. The 3 estimators are:
  - Sample mean, T^1
  - Sample median, T^2
  - Sample 10% trimmed mean, T^3

Examples: Estimators of μ

- Say we are interested to compare three estimators for the mean μ of a population distribution based on a random sample $Y_1, Y_2, ..., Y_n$. The 3 estimators are:
  - Sample mean, T^1
  - Sample median, T^2
  - Sample 10% trimmed mean, T^3
- When making comparison, we can base on some criteria like

Examples: Estimators of μ

$\bar{X} \rightarrow \mu$ $E(\bar{X}) - \mu = 0$

- Say we are interested to compare three estimators for the mean μ of a population distribution based on a random sample $Y_1, Y_2, ..., Y_n$. The 3 estimators are:
  - Sample mean, T^1
  - Sample median, T^2
  - Sample 10% trimmed mean, T^3
- When making comparison, we can base on some criteria like
  - Bias$(T^k) = E(T^k) - \mu, \quad k = 1, 2, 3.$ Estimator with smaller bias is better.

Examples: Estimators of μ

- Say we are interested to compare three estimators for the mean μ of a population distribution based on a random sample $Y_1, Y_2, ..., Y_n$. The 3 estimators are:
  - Sample mean, T^1
  - Sample median, T^2
  - Sample 10% trimmed mean, T^3
- When making comparison, we can base on some criteria like
  - Bias(T^k) $= E(T^k) - \mu, \quad k = 1, 2, 3.$ Estimator with smaller bias is better.
  - SD(T^k) $= \sqrt{Var(T^k)}, \quad k = 1, 2, 3.$ Estimator with smaller SD is better.

Examples: Estimators of μ

- Say we are interested to compare three estimators for the mean μ of a population distribution based on a random sample $Y_1, Y_2, ..., Y_n$. The 3 estimators are:
  - Sample mean, T^1
  - Sample median, T^2
  - Sample 10% trimmed mean, T^3
- When making comparison, we can base on some criteria like
  - $\text{Bias}(T^k) = E(T^k) - \mu, \quad k = 1, 2, 3.$ Estimator with smaller bias is better.
  - $\text{SD}(T^k) = \sqrt{Var(T^k)}, \quad k = 1, 2, 3.$ Estimator with smaller SD is better.

Most common used criteria

  - $\text{MSE}(T^k) = \text{Bias}(T^k)^2 + \left(\text{SD}(T^k)\right)^2, \quad k = 1, 2, 3.$ Smaller MSE is better.

Using MSE, compare $\bar{X}$, sample median, trimmed mean

Estimators of μ: Simulation Procedure

For a particular choice of underlying distribution, μ and sample size n.

- Step 1. Generate independent draws $Y_1, Y_2, ..., Y_n$ from the distribution.

Estimators of μ: Simulation Procedure

For a particular choice of underlying distribution, μ and sample size n.

- Step 1. Generate independent draws $Y_1, Y_2, ..., Y_n$ from the distribution.
- Step 2. Compute the value of T^1, T^2, T^3 ~~form~~ from the sample.

Estimators of μ: Simulation Procedure

For a particular choice of underlying distribution, μ and sample size n.

- Step 1. Generate independent draws $Y_1, Y_2, ..., Y_n$ from the distribution.
- Step 2. Compute the value of T^1, T^2, T^3 form the sample.
- Repeat {Step 1, Step 2} M times. Hence, we have M copies of each estimator:

NSE

sample mean $T^1 : \quad T^1_1, T^1_2, ..., T^1_M$: M values ⇒ mean; var

sample median $T^2 : \quad T^2_1, T^2_2, ..., T^2_M$: M values ⇒ mean; var

sample trimmed mean $T^3 : \quad T^3_1, T^3_2, ..., T^3_M$: M values. ⇒ mean ; var

Estimators of μ: Simulation Procedure

For a particular choice of underlying distribution, μ and sample size n.

- Step 1. Generate independent draws $Y_1, Y_2, ..., Y_n$ from the distribution.
- Step 2. Compute the value of T^1, T^2, T^3 form the sample.
- Repeat {Step 1, Step 2} M times. Hence, we have M copies of each estimator:

$$T^1: \quad T_1^1, T_2^1, ..., T_M^1$$

$$T^2: \quad T_1^2, T_2^2, ..., T_M^2$$

$$T^3: \quad T_1^3, T_2^3, ..., T_M^3$$

- Calculate the value for criteria (Bias, SD and MSE) for each estimator to make the comparison between the 3 estimators.

Estimators of μ: Comparison

For each $k = 1, 2, 3$, we calculate

- Mean (of M sample means): $\widehat{\text{mean}} = \frac{1}{M}\sum_{m=1}^{M} T_m^k = \bar{T}^k$

- $\widehat{\text{Bias}(T^k)} = \bar{T}^k - \mu$

- $\widehat{\text{SD}(T^k)} = \sqrt{\frac{1}{M-1}\sum_{m=1}^{M}\left(T_m^k - \bar{T}^k\right)^2}$

- $\widehat{\text{MSE}(T^k)} = \widehat{\text{Bias}(T^k)}^{2} + \widehat{\text{SD}(T^k)}^2$.

Depend on what criteria you choose to compare the estimator, you can make conclusion on which estimator is the best to estimate μ.

Estimators of μ: Implementation

- In order to estimate the population mean μ, we consider the 3 estimators: sample mean, sample median and sample 10% trimmed mean.

Estimators of μ: Implementation

- In order to estimate the population mean μ, we consider the 3 estimators: sample mean, sample median and sample 10% trimmed mean.

- Aim: to compare the performance of these 3 estimators.

Estimators of μ: Implementation

- In order to estimate the population mean μ, we consider the 3 estimators: sample mean, sample median and sample 10% trimmed mean.

- Aim: to compare the performance of these 3 estimators.

- Population distribution is $N(170, 10)$ (the true μ is 170).

Estimators of μ: Implementation

- In order to estimate the population mean μ, we consider the 3 estimators: sample mean, sample median and sample 10% trimmed mean.
- Aim: to compare the performance of these 3 estimators.
- Population distribution is $N(170, 10^2)$ (the true μ is 170).

  sd = 10.

- Sample size $n = 20$.

Estimators of μ: Implementation

- In order to estimate the population mean μ, we consider the 3 estimators: sample mean, sample median and sample 10% trimmed mean.
- Aim: to compare the performance of these 3 estimators.
- Population distribution is $N(170, 10)$ (the true μ is 170).
- Sample size $n = 20$.
- Simulation size: $M = \underline{10000}$

Estimators of μ: R code (1)

```
> # To compare 3 estimators for location, mu, through a simulation
> # study, the 3 estimators are sample mean, sample median,
> # and 10% trimmed mean.
>
> rm(list= ls())
> M <- 10000 # simulation size (No. of samples)
> n <- 20 # sample size
> mu <- 170 # Mean of the underlying normal distribution
> sd <- 10 # Standard deviation of the underlying normal distributio
> meanx <- numeric(M) # A vector of all sample means
> medx <- numeric(M) # A vector of all sample medians
> trmx <- numeric(M) # A vector of all sample 10% trimmed means
> stdx <- numeric(M) # A vector of all sample standard deviations
> set.seed(1234)
> # Using the same seed number gives same result every time
```

Estimators of μ: R code (2)

```
> for (i in 1:M){
+ x <- rnorm(n,mu,sd) # Generate a random sample of size n
+ meanx[i] <- mean(x) # Compute mean of the i-th sample: T^1_i
+ medx[i] <- median(x) #Compute median of the i-th sample: T^2_i
+ trmx[i] <- mean(x, trim=0.1)
+ # Compute the 10% trimmed mean for the i-th sample, i.e. T^3_i
+ stdx[i] <- sd(x)
+ # Compute the standard deviation for the i-th sample.
+ # It is used for computing confidence interval.
+ }
```

170 10

⇒ vector of M sample means: ⇐ get mean, sd

______ medians ⇐ " " "

______ trimmed mean ⇐

Estimators of μ: R code (3)

```
> # Compute mean of M sample means, medians, and trimmed means
> simumean <- apply(cbind(meanx,medx,trmx),2,mean)
> # "2" in the second argument asks the R to find "means" for
> # all the columns in the matrix given in argument 1.
> # Hence "simumean" is a 1x3 vector consists of
> # the average of the "ns" means, medians, and the 10% trimmed mean
>
> # Compute sd of the M sample means, medians and 10% trimmed means
> simustd <- apply(cbind(meanx,medx,trmx),2,sd)
> #Compute the bias
> simubias <- simumean - rep(mu,3)
> # Compute the MSE (Mean Square Error)
> simumse <- simubias^2 + simustd^2
```

mean($\bar{x}$) 170.
mean(med) 170.
mean(trim) 170.

Estimators of μ: R code (4)

```
> estimators <- c("Sample Mean", "Sample Median",
+                 "Sample 10% trimmed mean")
> # column heading in the output
> names <- c("True value", "No. of simu", "MC Mean",
+ "MC Std Deviation","MC Bias","MC MSE")
> # row heading in the output
> sumdat <- rbind(c(mu,mu,mu), M, simumean, simustd, simubias,
+                 simumse)
> dimnames(sumdat) <- list(names,estimators)
> round(sumdat,4)
```

mean of M sample means

mean of M sample median

	Sample Mean	Sample Median	Sample 10% trimmed mean
True value	170.0000	170.0000	170.0000
No. of simu	10000.0000	10000.0000	10000.0000
MC Mean	170.0307	170.0314	170.0243
MC Std Deviation	2.2143	2.6935	2.2757
MC Bias	0.0307	0.0314	0.0243
MC MSE	4.9042	7.2559	5.1796

smallest MSE

The sample mean is a better estimator of population mean μ in term of MSE.

1. Introduction

2. Random Number Generator

3. Simulation Studies in Statistics
   - Simulation: Comparing Estimators of Mean
   - Simulation: Coverage Probability of Confidence Intervals
   - Simulation: Properties of Hypothesis Tests
   - END

Estimation of μ: Interval Estimate

$\bar{X} \sim N(\mu; \sigma^2/n)$

$SD(\bar{X}) = \sigma/\sqrt{n}$

$SE(\bar{Y}) = s/\sqrt{n}$

bcz of 95%.

- The usual 95% confidence interval for μ based on the sample mean from one sample is built as

$$\bar{Y} \pm t_{n-1}(0.025) \times \underbrace{s/\sqrt{n}}_{\text{se of } \bar{X}}$$

where s is the standard deviation of that sample.

Estimation of μ: Interval Estimate

- The usual 95% confidence interval for μ based on the sample mean from one sample is built as

$$\bar{Y} \pm t_{n-1}(0.025) \times s/\sqrt{n}$$

where s is the standard deviation of that sample.

- The usual interpretation for 95% CI is: 95% of the confidence intervals constructed as above will cover the true parameter. Is this statement reliable? We'll use simulation to check this statement.

100 → 95 ←
1000 → 950 ←
946
952

Estimation of μ: Interval Estimate

- The usual 95% confidence interval for μ based on the sample mean from one sample is built as
$$\bar{Y} \pm t_{n-1}(0.025) \times s/\sqrt{n}$$
where s is the standard deviation of that sample.

- The usual interpretation for 95% CI is: 95% of the confidence intervals constructed as above will cover the true parameter. Is this statement reliable? We'll use simulation to check this statement.

- For each of the M samples, we'll form a 95% CI for μ. We expect 95% of the M CIs will cover the true mean $\mu = 170$.

Estimation of μ: Interval Estimate

- The usual 95% confidence interval for μ based on the sample mean from one sample is built as
$$\bar{Y} \pm t_{n-1}(0.025) \times s/\sqrt{n}$$
where s is the standard deviation of that sample.

- The usual interpretation for 95% CI is: 95% of the confidence intervals constructed as above will cover the true parameter. Is this statement reliable? We'll use simulation to check this statement.

- For each of the M samples, we'll form a 95% CI for μ. We expect 95% of the M CIs will cover the true mean $\mu = 170$.

- Write R code to get the results of coverage (which should be close to 0.95).

CIs of μ: R code

$\bar{X} \pm t * S/\sqrt{n}$; n=20

got M values of sample means ; sd for each sample (S) $\in$ $\bar{X}^*$

→ M CIs for M samples.

```
> # Check the coverage probability of Confidence interval
> # We make use of the data obtained from above simulation study
> # Get t_{0.025}(0.025)
> t05 <- qt(0.975,n-1)
> d <- 0
> for (i in 1:M) {
+ # check if the i-th CI covers mu or not
+ cover <- (meanx[i]-t05*stdx[i]/sqrt(n)<= mu)&
+           (mu<= meanx[i]+t05*stdx[i]/sqrt(n))
+ d <- d + cover
+ }
> coverage <- d/M; coverage # this value should close to 0.95
[1] 0.9538
```

true value of μ = 170 (slide 42)

set.seed(1234)

The value of coverage as 0.9538. So the coverage is good.

1. Introduction

2. Random Number Generator

3. Simulation Studies in Statistics
   - Simulation: Comparing Estimators of Mean
   - Simulation: Coverage Probability of Confidence Intervals
   - **Simulation: Properties of Hypothesis Tests** ←
   - END

$H_0: \mu = \mu_0$ vs $H_1: \mu \neq \mu_0$.

Type 1 = Prob (reject H_0 when it is true.)

Type 2 = Prob (do not reject H_0 when H_1 is true)

power of a test = 1 − Type 2
= Prob (reject H_0 when H_1 is true)

t-Test for the Mean

$IQ \sim N(100; \sigma = 10) \Leftarrow$ $\quad H_0 : \mu = 100$.

$\bar{X}, S \Rightarrow t \rightarrow \boxed{\text{small p-value}}$

Consider the usual t-test for the population mean μ at significance level α:

$$H_0 : \mu = \mu_0 \quad \text{vs} \quad H_1 : \mu \neq \mu_0$$

Note that, α is the probability of ~~not~~ rejecting H_0 when it is true.

Power of a test is the probability of rejecting H_0 when H_1 is true.

- To evaluate whether the **size**/**level** of test can achieve the advertised α:

t-Test for the Mean

Consider the usual t-test for the population mean μ at significance level α:

$$H_0 : \mu = \mu_0 \quad \text{vs} \quad H_1 : \mu \neq \mu_0$$

Note that, α is the probability of not rejecting H_0 when it is true.

Power of a test is the probability of rejecting H_0 when H_1 is true.

- To evaluate whether the **size**/**level** of test can achieve the advertised α:
    - Generate data under $H_0 : \mu = \mu_0$ and calculate the proportion of rejections of H_0.

t-Test for the Mean

Consider the usual t-test for the population mean μ at significance level α:

$$H_0 : \mu = \mu_0 \quad \text{vs} \quad H_1 : \mu \neq \mu_0$$

Note that, α is the probability of not rejecting H_0 when it is true.

Power of a test is the probability of rejecting H_0 when H_1 is true.

- To evaluate whether the **size**/**level** of test can achieve the advertised α:
  - Generate data under $H_0 : \mu = \mu_0$ and calculate the proportion of rejections of H_0.
  - Get the true probability of rejecting H_0 when it is true (the proportion should be close to α).

t-Test for the Mean

Consider the usual t-test for the population mean μ at significance level α:

$$H_0 : \mu = \mu_0 \quad \text{vs} \quad H_1 : \mu \neq \mu_0$$

Note that, α is the probability of not rejecting H_0 when it is true.

Power of a test is the probability of rejecting H_0 when H_1 is true.

- To evaluate whether the **size**/**level** of test can achieve the advertised α:
  - Generate data under $H_0 : \mu = \mu_0$ and calculate the proportion of rejections of H_0.
  - Get the true probability of rejecting H_0 when it is true (the proportion should be close to α).

- To estimate the power of the test:

t-Test for the Mean

Consider the usual t-test for the population mean μ at significance level α:

$$H_0 : \mu = \mu_0 \quad \text{vs} \quad H_1 : \mu \neq \mu_0$$

Note that, α is the probability of not rejecting H_0 when it is true.

Power of a test is the probability of rejecting H_0 when H_1 is true.

- To evaluate whether the **size**/**level** of test can achieve the advertised α:
  - Generate data under $H_0 : \mu = \mu_0$ and calculate the proportion of rejections of H_0.
  - Get the true probability of rejecting H_0 when it is true (the proportion should be close to α).

- To estimate the power of the test:
  - Generate data under some alternative $H_1 : \mu \neq \mu_0$, say $\mu = \mu_1 \neq \mu_0$ and calculate the proportion of rejections of H_0.

t-Test for the Mean

Consider the usual t-test for the population mean μ at significance level α:

$$H_0 : \mu = \mu_0 \quad \text{vs} \quad H_1 : \mu \neq \mu_0$$

Note that, α is the probability of not rejecting H_0 when it is true.

Power of a test is the probability of rejecting H_0 when H_1 is true.

- To evaluate whether the **size**/**level** of test can achieve the advertised α:
  - Generate data under $H_0 : \mu = \mu_0$ and calculate the proportion of rejections of H_0.
  - Get the true probability of rejecting H_0 when it is true (the proportion should be close to α).

$H_0: \mu = 160$ $H_1: \mu > 160$

$\mu = 170$

- To estimate the power of the test:
  - Generate data under some alternative $H_1 : \mu \neq \mu_0$, say $\mu = \mu_1 \neq \mu_0$ and calculate the proportion of rejections of H_0.
  - Approximate the true probability of rejecting H_0 when H_1 is true.

Checking the Size of t-Test: R code

α

$H_0 : \mu = 170$.

data $\sim N(170 ; \sigma = 10)$.

```
> # Check the size of t-test
> # We make use of the data obtained from the above simulation study
> ttests <- (meanx-mu)/(stdx/sqrt(n))
> size <- sum(abs(ttests) > t05)/M
> size
[1] 0.0462
```

0.0538 999 0.05 seed(1234)

1% 5%

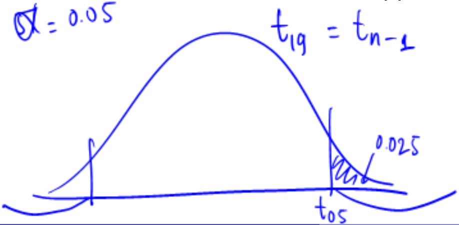
test statistic: $t = \dfrac{\bar{x} - 170}{s/\sqrt{n}} > t_{05} \Rightarrow$ reject H_0

prob of rejection $\approx 5\%$

$\to \approx 1\%$.

We see that the size of 0.05 is approximately being achieved.

$\alpha = 0.05$



1. Introduction

2. Random Number Generator

3. Simulation Studies in Statistics
   - Simulation: Comparing Estimators of Mean
   - Simulation: Coverage Probability of Confidence Intervals
   - Simulation: Properties of Hypothesis Tests
   - END