MATLAB Session II

171030 TA Session 5

Yeongmi Jeong

Contents

1. Review: Exercise

- $X_1, X_2, \ldots, X_n \sim$ i.i.d. Bernoulli(p). Let $\hat{p} = \frac{1}{n}\sum X_i$.
- Pointwise asymptotic $1 - \alpha$ confidence interval for p is

$$\left(\hat{p}_n - z_{\frac{\alpha}{2}}\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}, \hat{p}_n + z_{\frac{\alpha}{2}}\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}\right).$$

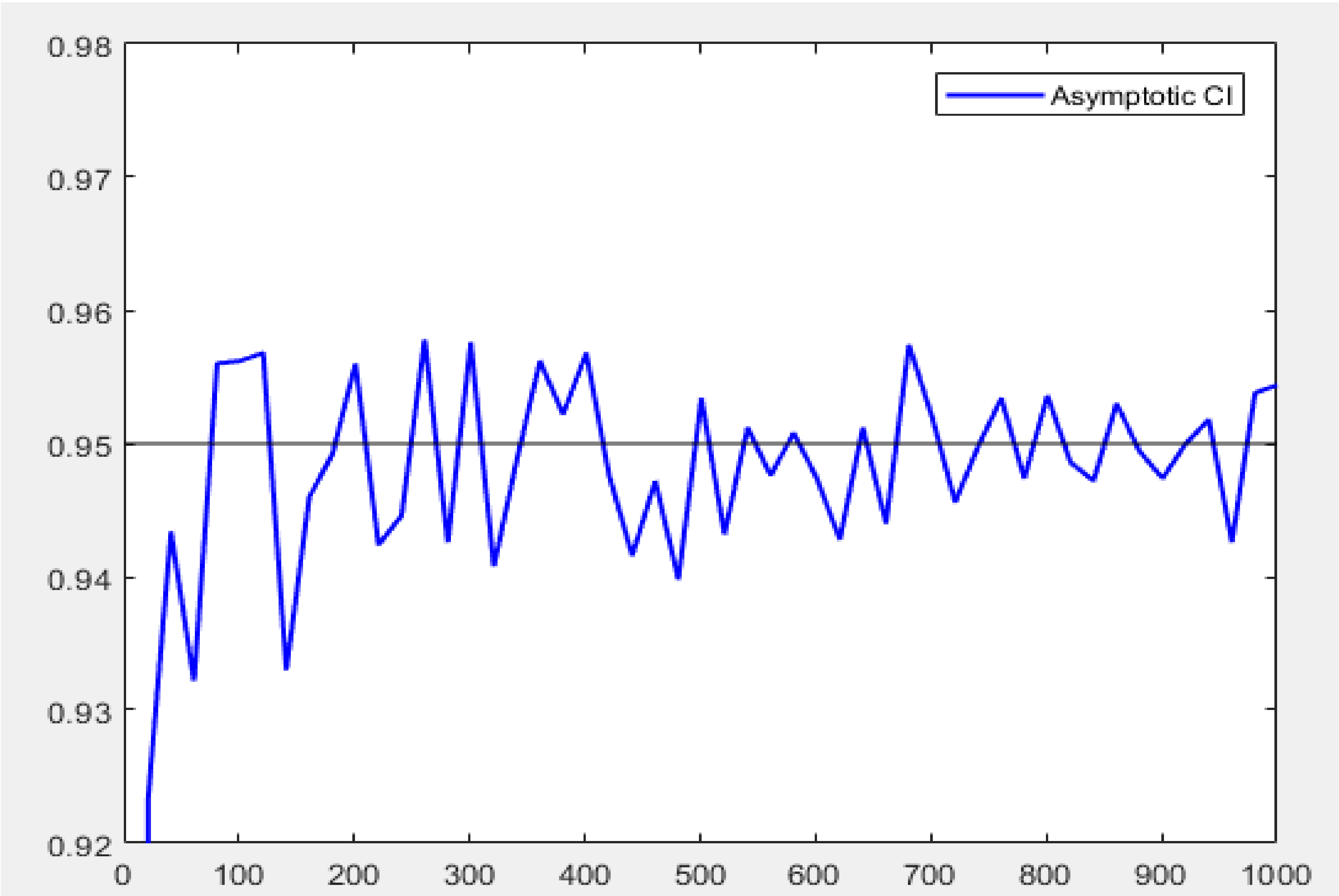
- We want to see how fast the coverage converges to 0.95 as the size of sample n increases.

1. Review: Exercise

```
p=0.5; alpha=0.05; % declare constants
z=icdf('normal',1-alpha/2,0,1); % compute quantiles

x=1:20:1001; n=size(x,2); %sample size
sim_num=5000;

coverage=zeros(n,1);

parfor i=1:n
    sample=random('binomial',1,p,x(i),sim_num); % B(1,p)=Bernoulli(p)
    phat=mean(sample); % 1 by sim_num row vector
    cover=(p<=phat+z.*sqrt(phat.*(1-phat)./x(i)))&(p>=phat-z.*sqrt(phat.*(1-phat)./x(i)));
    coverage(i)=sum(cover)/sim_num;
end

plot(x,coverage,'b','Linewidth',1.5)
hold on
plot(x,ones(n,1)*0.95,'k')
legend('Asymptotic CI'); ylim([0.92,0.98]); xlim([0,1000]);
```

alpha	0.0500
cover	1x5000 logical
coverage	51x1 double
i	2
n	51
p	0.5000
phat	1x5000 double
sample	21x5000 double
sim_num	5000
x	1x51 double
z	1.9600

1. Review: Exercise



Contents

2. Efficient Code

- 'for' executes statements specified number of times.

```
for i=1:100
    commands
end
```

✓ n=n+1; is implicitly embedded in the end of the commands.

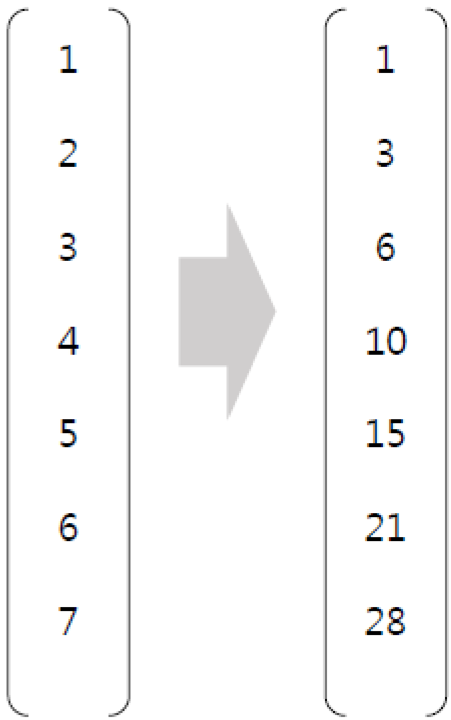
✓ Loop variable is scalar within the command block. Loop variable can be defined by a vector.
  - Eg. a=0:2:10
  - Eg. A=[0 2 4 6 8 10]

➢ **We can make codes more efficiently by using built-in functions or vectorization!**

2. Efficient Code: built-in function

- **Example**

✓ Cumulative sum vector



1. Loop

```
a=1:7; b=zeros(1,7);
for i=1:7
    if i==1
        b(i)=a(i);
    else
        b(i)=a(i)+b(i-1);
    end
end
```

2. Built-in Function

```
b=cumsum(a);
```

Search before coding!

2. Efficient Code: Vectorization

- **Example**

✓ <u>Consecutive sum vector</u>

$$\begin{pmatrix}1\\2\\3\\4\\5\\6\\7\end{pmatrix} \Rightarrow \begin{pmatrix}1\\3\\5\\7\\9\\11\\13\end{pmatrix}$$

1. **Loop**

```
a=1:7; b=zeros(1,7);
for i=1:7
    if i==1
        b(i)=a(i);
    else
        b(i)=a(i)+a(i-1);
    end
end
```

2. **Vectorization**

```
b=[0 a(1:6)]+a;
```

2. Efficient Code: Vectorization

- **Example**

```
p=0.5; alpha=0.05; z=icdf('normal',1-alpha/2,0,1); % declare constants & compute quantiles
n=100                                              % sample size
sim_num=5000;                                      % simulation number

cover=zeros(1,n);

for i=1:sim_num
    sample=random('binomial',1,p,n,1);             % B(1,p)=Bernoulli(p)
    phat=mean(sample);
    cover(i)=(p<=phat+z*sqrt(phat*(1-phat)/n))&(p>=phat-z*sqrt(phat*(1-phat)/n));
end

coverage=sum(cover)/sim_num;
```

```
p=0.5; alpha=0.05; z=icdf('normal',1-alpha/2,0,1); % declare constants & compute quantiles
n=100                                              % sample size
sim_num=5000;                                      % simulation number

cover=zeros(1,n);

    sample=random('binomial',1,p,n,sim_num);
    phat=mean(sample);                             % 1 by sim_num row vector
    cover=(p<=phat+z*sqrt(phat.*(1-phat)/n))&(p>=phat-z*sqrt(phat.*(1-phat)/n)); % 1 by 5000

coverage=sum(cover)/sim_num;
```

Contents

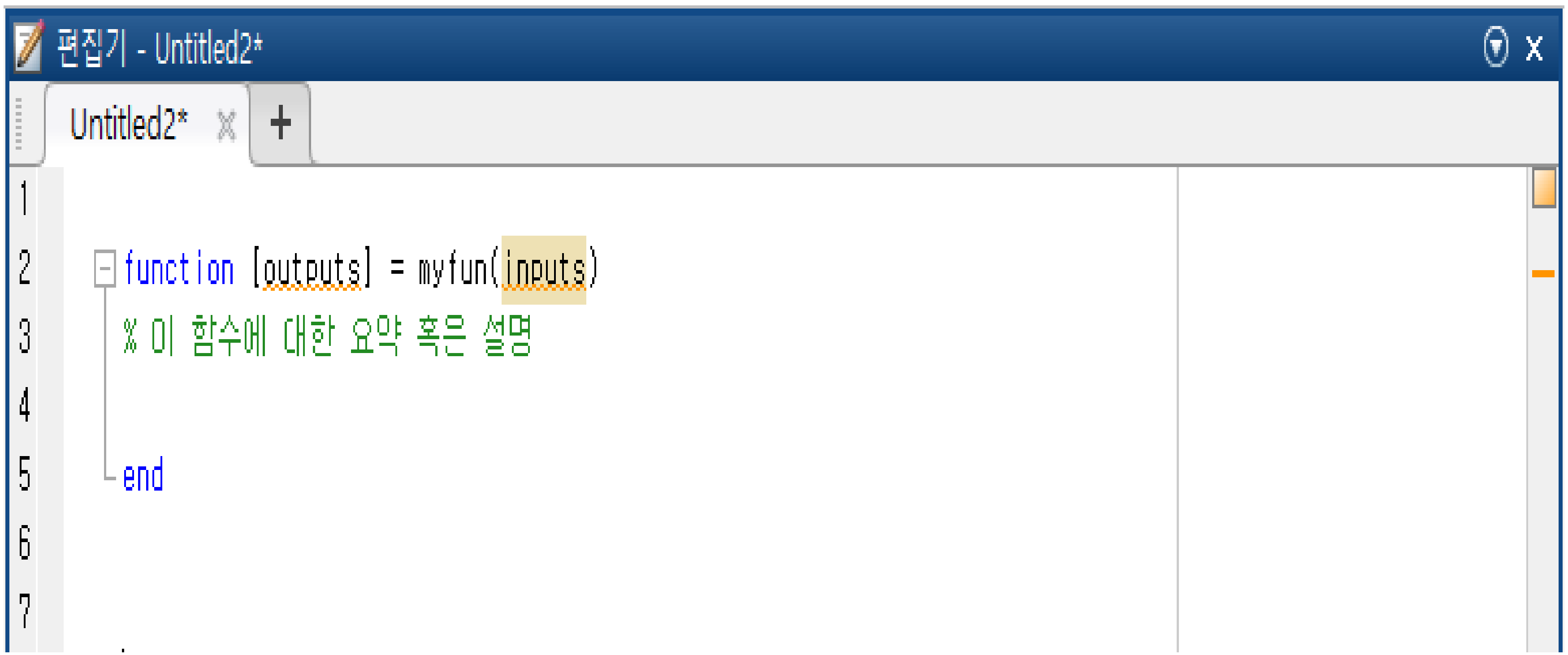
2. User-defined function

- **Function [y1, y2, …, ym] = myfun(x1, …, xn)**

M outputs

Function Name

N inputs



2. User-defined function

- **Function [y1, y2, …, ym] = myfun(x1, …, xn) ………. End**

M outputs ↑   Function Name ↑   N inputs ↑

✓ Function name
- Must begin with an alphabetic character.
- Must match the name of the file.

✓ Function file can contain multiple local functions or nested functions.

✓ Any variable created within the function but not returned disappear after the function stops running.

(A function works on a separated workspace from the base workspace.)

2. User-defined function: example 1

Main Code

```
p=0.5; alpha=0.05; z=icdf('normal',1-alpha/2,0,1); % declare constants & compute quantiles
n=100:10:500                                        % sample size
sim_num=5000;                                       % simulation number

prob=zeros(size(n,2),1);

for j=1:size(n,2)
    sample=random('binomial',1,p,n(j),sim_num);     % B(1,p)=Bernoulli(p)
    prob(j)=asyCI(p,sample);
end
```

Function Code

```
function [coverage]=asyCI(parameter, data)
    z=icdf('normal',0.975,0,1);
    [n k]=size(data);
    phat=mean(data);
    cover=(p<=phat+z*sqrt(phat.*(1-phat)/n))&(p>=phat-z*sqrt(phat.*(1-phat)/n));
    coverage=sum(cover)/k;
end
```

2. User-defined function: example 2

- **OLS Estimation Function**

Function Code

OLS_Estimation.m | exercise_ols.m | +

```
function [b, SE, resid, n, k] = OLS_Estimation(y,x)

% OLS Estimation Function
% ***** INPUT  **********
% y: dependent variable, x: independent variable
% ***** OUTPUT **********
% b: estimator/ SE: standard error / resid: residuals
% n: observation/ k: number of regressors (including constants)

X = [ones(length(y),1), x];
Y = y;

[n k]=size(X);

b=(X'*X)₩(X'*Y);
resid=X*b-Y;

RSS=resid'*resid;
sigma2=(RSS/(n-k));
cov_mat=sigma2*inv(X'*X);
SE=sqrt(diag(cov_mat));

end
```

2. User-defined function: example 2

- **Simple way for OLS Estimation**

Using User defined Function

```
% using user defined function
load('dataset.mat')
y=dataset(:,1);
x=dataset(:,3:4);
[b,se,resid,n,k]=OLS_Estimation(y,x)
```

Using Existing function

```
% using exiting function
x1=x(:,1); x2=x(:,2);
tbl=table(y,x1,x2);
lm=fitlm(tbl, 'y~x1+x2')
```

작업 공간

이름	값
b	[10.0052;3.9776;1.0094]
dataset	1000x12 double
k	3
lm	1x1 LinearModel
n	1000
resid	1000x1 double
se	[0.0161;0.0161;0.0161]
tbl	1000x3 table
x	1000x2 double
x1	1000x1 double
x2	1000x1 double
y	1000x1 double

명령 창

```
lm =


선형 회귀 모델:
    y ~ 1 + x1 + x2

추정된 계수:
                   Estimate      SE        tStat    pValue
                   ________   ________    ______    ______

    (Intercept)     10.005    0.016071    622.58      0
    x1              3.9776    0.016098    247.09      0
    x2              1.0094     0.01611    62.659      0


관측값 개수: 1000, 오차 자유도: 997
RMS 오차: 0.508
결정계수: 0.985, 수정된 결정계수 0.985
상수 모델에 대한 F-통계량: 3.21e+04, p-값 = 0
```

2. User-defined function

- **Anonymous function**

✓ <u>myfun = @(input) statement</u>

- A function that is not stored in a program file, and is composed of a single executable statement.
- Accepts multiple inputs and outputs.

명령 창
```
>> sqroot=@(x) x.^2

sqroot =

  다음 값을 갖는 function_handle:

    @(x)x.^2

>> sqroot([1 2 3])

ans =

     1     4     9
```

명령 창
```
>> myfun=@(x,y) x.^2+y.^2+x.*y

myfun =

  다음 값을 갖는 function_handle:

    @(x,y)x.^2+y.^2+x.*y

>> x=[1 2];
>> y=[3 4];
>> z=myfun(x,y)

z =

    13    28
```

Q & A

END