R컴퓨팅

14강 시뮬레이션

정보통계학과 장영재 교수

1. 몬테카를로 시뮬레이션 (MC 시뮬레이션)
2. 중심극한 정리 (Central Limit Theorem)
3. 수치적분 (Numerical Integration)
4. 최적화 기법 (Optimization Technique)

1 몬테카를로 시뮬레이션 (MC 시뮬레이션)

1 몬테카를로 시뮬레이션 (MC 시뮬레이션)

- 시뮬레이션을 이용하는 이유는 수리적으로 계산이 불가능하거나 너무 어려운 경우에 시뮬레이션을 이용하면 비교적 쉽게 계산이 가능하기 때문
- 연속형 변수의 평균과 분산은 정의에 의해서 다음과 같이 계산

$$E(h(X)) = \int_{-\infty}^{+\infty} h(x)\phi(x)dx,$$

$$Var(h(X)) = \int_{-\infty}^{+\infty} (h(x)-\mu)^2\phi(x)dx.$$

$\phi(X)$는 $N(10,5^2)$의 확률밀도함수(probability density function, pdf)이고, μ는 $E(h(X))$

- 적분을 통해서 평균과 분산을 계산할 수 있으나 위의 적분은 수리적으로 계산이 불가능하므로 MC 시뮬레이션 이용

1 몬테카를로 시뮬레이션 (MC 시뮬레이션)

- 어떤 랜덤 변수 X 가 확률밀도함수 f(x)를 가진다고 가정할 때, X의 함수 h(X)의 평균 E(h(x))를 계산하려 한다면 아래와 같은 적분을 계산해야 함

$$E(h(X)) = \int_{-\infty}^{+\infty} h(x) f(x) dx$$

1 몬테카를로 시뮬레이션 (MC 시뮬레이션)

MC 시뮬레이션

- 우리가 확률밀도함수 f(x)를 가지는 iid 랜덤 변수를 n개 $(X1,\cdots,Xn)$ 발생시킬 때 E(h(X))에 대한 MC 추정치 $\widehat{\mu^{MC}}$ 는

$$\widehat{\mu^{MC}} = \frac{1}{n}\sum_{i=1}^{n} h(X_i)$$

이 추정치는 대수의 법칙에 의해 (Law of large number) 참값과 아주 가까워질 수 있음

1 몬테카를로 시뮬레이션 (MC 시뮬레이션)

※ 정규분포에 대한 4가지 함수

dnorm(x,mean,sd)	d는 density를 의미. 평균이 mean, 표준편차가 sd인 정규분포의 확률밀도함수(pdf)
pnorm(x,mean,sd)	p는 probability를 의미. 평균이 mean, 표준편차가 sd인 정규분포의 분포함수(cdf)
qnorm(p,mean,sd)	q는 quantile을 의미. 평균이 mean, 표준편차가 sd인 정규분포의 백분위수 함수. 분포함수의 역함수
rnorm(n,mean,sd)	r은 random을 의미. 평균이 mean, 표준편차가 sd인 정규분포에서 랜덤 변수를 n개 발생

1 몬테카를로 시뮬레이션 (MC 시뮬레이션)

- 어떤 랜덤 변수 X 가 확률밀도함수 f(x)를 가진다고 가정할 때, X의 함수 h(X)의 평균 E(h(x))를 계산하려 한다면 아래와 같은 적분을 계산해야 함

```
> x<-rnorm(100,mean=0,sd=1)
> summary(x)
Min. 1st Qu. Median Mean 3rd Qu. Max.
-2.62100 -0.57980 0.07694 -0.06936 0.53600 2.61200
> sd(x)
[1] 1.032852
```

- 100개의 자료가 표준정규분포에서 발생된 것이지만 표본평균이 0에 가깝긴 하지만 0과 완전히 일치하는 것은 아님
- MC시뮬레이션에서 좀 더 높은 정확성이 필요하다면 이렇게 생성되는 표본의 수를 늘리는 것이 필요

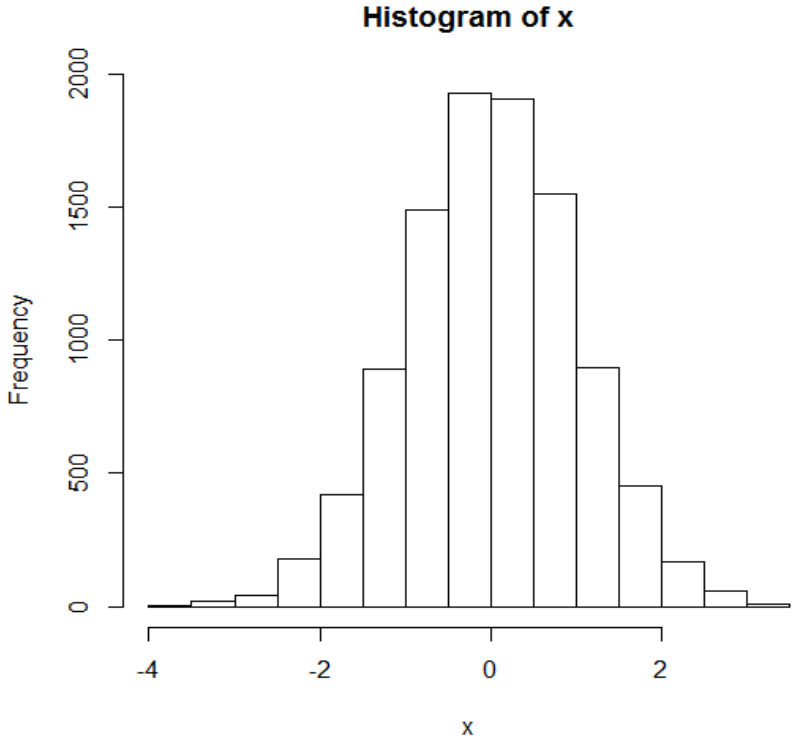
1 몬테카를로 시뮬레이션 (MC 시뮬레이션)

- 보기 14-1: 위의 예에서 100개의 자료 대신 10000개의 자료를 생성해서 자료의 분포를 살펴보기

```
> x<-rnorm(10000,0,1)
> summary(x)
Min. 1st Qu. Median Mean 3rd Qu. Max.
-3.967000 -0.651700 0.007021 0.006971 0.673000 3.824000
> sd(x)
[1] 0.9957861
> hist(x)
```

1 몬테카를로 시뮬레이션 (MC 시뮬레이션)

※ 표준정규분포에서 발생된 10000개의 랜덤변수를 이용한 히스토그램



1 몬테카를로 시뮬레이션 (MC 시뮬레이션)

› 보기 14-2: 랜덤 변수 X가 평균이 10이고 표준편차가 5인 정규분포를 따 를 때, $h(X)=(X^2+2X+\log(x^2+3))/\sqrt{5|X|+\exp(x)}$ 의 평균 $(E(h(X)))$과 분산 $(Var(h(X))$을 계산하기

- MC 시뮬레이션을 이용해서 계산하려면 $N(10,5\text{^}2)$ 분포에서 많은 수의 랜덤변수를 발생시킨 다음에 h(x)값을 계산한 후, 그 평균값을 취함

```
〉 x〈−rnorm(10000,mean=10,sd=5)
〉 hx〈−(x^2+2*x+log(x^2+3))/sqrt(5*abs(x)+exp(x))
〉 mean(hx)
[1] 1.128817
〉 var(hx)
[1] 1.034547
```

1 몬테카를로 시뮬레이션 (MC 시뮬레이션)

- 표본수 n을 증가하면 MC 추정치의 분산이 줄어들고 좀 더 정확한 추정이 가능하며 많은 경우에 이런 MC 추정치 계산은 한번만 하지 않고 여러 번 반복

```
> sim.n<-1000
> myhx<-rep(0,sim.n)
> for (i in 1:sim.n){
+     x<-rnorm(10000,mean=10,sd=5)
+     hx<-(x^2+2*x+log(x^2+3))/sqrt(5*abs(x)+exp(x))
+     myhx[i]<-mean(hx)
+ }
```

1 몬테카를로 시뮬레이션 (MC 시뮬레이션)

- 위의 코드는 앞에서 수행한 코드를 sim.n 번 만큼 (이 경우에는 1000번) 반복해서 sim.n 번 만큼의 MC 추정치를 계산한 다음에 이를 myhx 변수에 저장하는 코드로 저장된 myhx의 평균과 분산은

```
> mean(myhx)
[1] 1.129586
> var(myhx)
[1] 0.0001081087
```

2 중심극한 정리 (Central Limit Theorem)

2 중심극한 정리 (Central Limit Theorem)

- 통계학에서 가장 중요한 정리 중에 하나가 중심극한 정리

- 표본의 수가 커지면 표본평균의 분포가 모집단의 분포에 상관없이 정규분포로 근사한다는 것

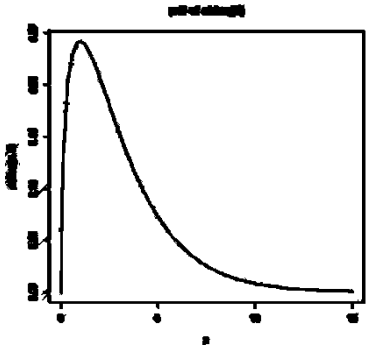
  - 시뮬레이션을 통해서 모집단의 분포가 정규분포가 아니더라도 표본평균의 분포가 정말 정규분포로 근사되는지 확인

2 중심극한 정리 (Central Limit Theorem)

- 보기 14-3: rchisq() 함수를 이용하여 카이제곱분포를 따르는 pdf 및 표본수 증가에 따른 표본평균의 분포의 변화

- 카이제곱분포는 1개의 모수를 가지는데 자유도라고 부르며 자유도 3인 카이제곱분포의 pdf는 아래와 같음

```
> x<-seq(0,15,by=0.1)
> plot(x,dchisq(x,3),type="l",main="pdf of chisq(3)")
```



2 중심극한 정리 (Central Limit Theorem)

- 표본크기가 5, 10, 20, 50 인 4가지 경우에 표본평균의 분포가 어떻게 되는 지 비교하기

```
>sim.n<-1000
>sam.n<-c(5,10,20,50)
>x.mean<-matrix(0,sim.n,4)
>for(j in 1:4){
>for (i in 1:sim.n){
+     x<-rchisq(sam.n[j],3)
+     x.mean[i,j]<-mean(x)
+}
+}
> par(mfrow=c(2,2))
```

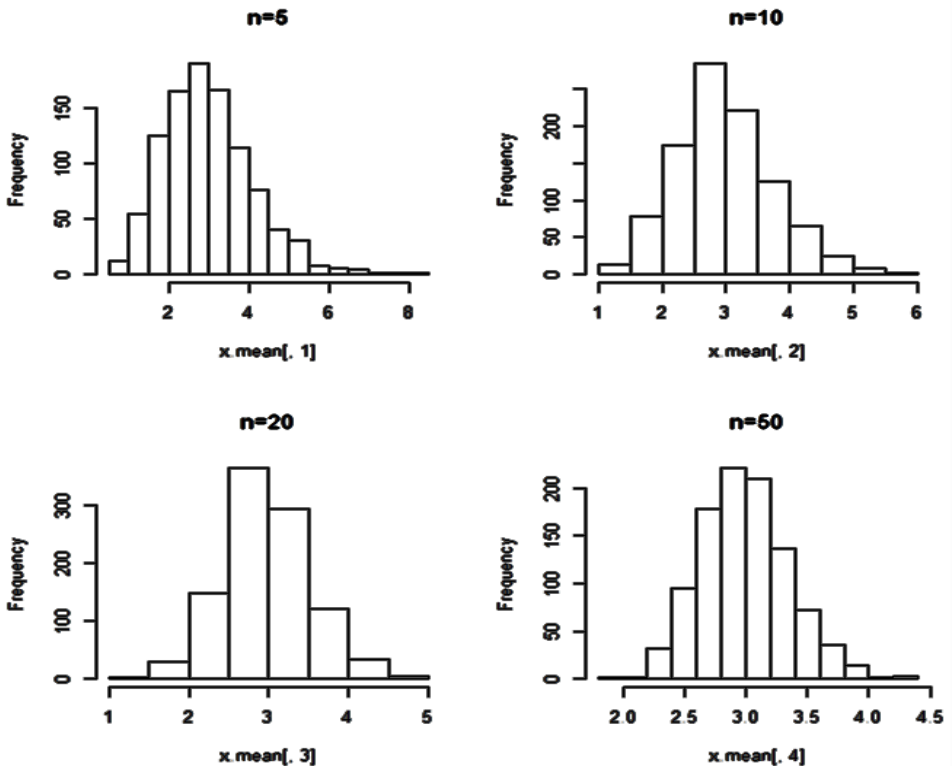
2 중심극한 정리 (Central Limit Theorem)

```
> hist(x.mean[,1],main="n=5")
> hist(x.mean[,2],main="n=10")
> hist(x.mean[,3],main="n=20")
> hist(x.mean[,4],main="n=50“)
```

- 표본수(n)가 증가함에 따라 표본평균의 분포는 점점 더 정규분포에 가까워지고 또한 분산도 줄어들고 있음

2 중심극한 정리 (Central Limit Theorem)

※ 다른 표본 수에 따른 표본 평균의 분포 (카이제곱 d.f. 3)



3 수치적분 (Numerical Integration)

3 수치적분 (Numerical Integration)

- 통계학에서는 연속형 분포의 평균이나 분산 계산에서 적분을 많이 이용

- 수치적분에는 여러 가지 알고리즘이 있으나 기본적인 아이디어는 적분 구간을 나눈 다음에 구간 내에서 적분해야 하는 함수를 다항식 함수(polynomial function)으로 근사한 다음에 적분하는 것

  - R에서 구현된 수치적분 알고리즘은 Gaussian Quadrature 방법론을 이용한 것으로 함수 이름은 integrate임

```
〉 integrate(dnorm, -1.96, 1.96)
0.9500042 with absolute error < 1e-11
〉 integrate(dnorm, -Inf, Inf)
1 with absolute error < 9.4e-05
```

3 수치적분 (Numerical Integration)

- 앞 절에서는 랜덤 변수 X가 평균이 10이고 표준편차가 5인 정규분포를 따른다고 가정하고,
  $h(X) = (X^2 + 2X + \log(x^2 + 3))/\sqrt{5|X| + \exp(x)}$ 의 평균 ($E(h(X))$)과 분산 ($Var(h(X))$)을 MC 시뮬레이션을 이용하여 계산한 바 있음

- 수치적분을 이용해서 계산할 수도 있음

$$E(h(X)) = \int_{-\infty}^{+\infty} h(x)\phi(x)dx,$$

$$Var(h(X)) = \int_{-\infty}^{+\infty} (h(x) - \mu)^2 \phi(x)dx.$$

3 수치적분 (Numerical Integration)

› 보기 14-4: $h(X) = (X^2 + 2X + \log(x^2 + 3))/\sqrt{5|X| + \exp(x)}$ 의 평균 ($E(h(X))$)과 분산 ($Var(h(X))$)을 수치적분을 이용해서 계산하기

› integrate 함수 안에 적분에 사용될 함수(integrand)를 정의

- 평균계산식의 적분부분을 보면 $h(x)\varnothing(x)$를 적분해야 하므로 이 함수를 정의한 다음에 integrate 함수를 이용

```
> mytestftn1<-function(x){
+     hx<-(x^2+2*x+log(x^2+3))/sqrt(5*abs(x)+exp(x))
+     res<-hx*dnorm(x,10,5)
+     return(res)
+ }
> integrate(mytestftn1,-Inf,Inf)
1.129548 with absolute error < 7.1e-06
```

: MC 시뮬레이션을 이용한 결과 1.129586과 소숫점 4째자리 까지 일치

3 수치적분 (Numerical Integration)

- 분산의 계산

```
>myintftn2<-function(x){
+     hx<-(x^2+2*x+log(x^2+3))/sqrt(5*abs(x)+exp(x))
+     res<-(hx-1.129548)^2*dnorm(x,10,5)
+     return(res)
+ }
> integrate(myintftn2,-Inf,Inf)
1.025533 with absolute error < 7e-06
```

- 이 값도 앞에서 MC 시뮬레이션을 이용한 결과와 비슷하며 일반적으로 수치적분이 더 정확한 값을 주고 (심지어 error 까지 제공함) 계산도 더 빠른 경우가 많음

4 최적화 기법 (Optimization Technique)

4 최적화 기법 (Optimization Technique)

- 통계학이나 수학에서 최적화란 어떤 함수의 최대값이나 최소값을 찾는 것

$$f(x) = (x-2)^2 + 3$$

- 이는 아주 간단한 함수로 우리는 모두 이 함수가 x=2에서 최소값 3을 갖는다는 것을 알고 있음

- R에서 일변량 함수의 최적화는 optimize 함수를 이용하면 계산할 수 있음
- 위의 함수에 대한 최적화는 다음과 같은 코드를 사용

4 최적화 기법 (Optimization Technique)

```
> myf1<-function(x){    # 함수를 정의
+     res<-(x-2)^2+3
+     return(res)}
> optimize(myf1,c(-10,10))
$minimum
[1] 2
$objective
[1] 3
```

- optimize함수는 첫 번째 인수가 최적화할 함수이고, 두 번째 인수가 범위
- myf1이라는 함수를 주면서 최적화 구간을 (-10,10)으로 지정해 주면 결과산출
- 결과 중 $minimum이 2라는 것은 x=2에서 이 함수가 최소값을 가진다는 것, $object가 3이라는 것은 그때 최소값이 3이라는 뜻

4 최적화 기법 (Optimization Technique)

- 보기 14-5: 우리가 최적화 하려는 함수가 $f(x) = \frac{\log(x)}{1+x}$ 라고 할 때, R의 optimize함수를 이용하여 최대값을 찾아보기

```
> myf2<-function(x){
+     res<-log(x)/(1+x)
+     return(res)
+ }
> optimize(myf2,c(0,20),maximum=T)
$maximum
[1] 3.591121
$objective
[1] 0.2784645
```

4 최적화 기법 (Optimization Technique)

- 앞에서와 다른 점은 optimize 함수 안에 있는 maximum 이라는option에 TRUE 값을 준 것

- 기본적으로 optimize 함수는 최소값을 찾는 데 만약 최대값을 찾고 싶으면 maximum=T 설정

R컴퓨팅