Note 13. Statistical Programming

Introduction to Statistical Programming

Chanmin Kim

Department of Statistics
Sungkyunkwan University

2022 Spring

Basic Mathematical Functions

- `exp()`: Exponential function with base e.
- `log()`: Natural logarithm.
- `log10()`: Logarithm with base 10.
- `sqrt()`: Square root.
- `abs()`: Absolute value.
- `sin()`, `cos()`, `tan()`: Triangular functions.
- `min()`, `max()`: Minimum / maximum value within a vector.
- `which.min()`, `which.max()`: Index of the minimal / maximal element of a vector.

Basic Mathematical Functions

- `sum()`, `prod()`: Sum / Product of the elements of a vector.
- `cumsum()`, `cumprod()`: Cumulative sum / product of the elements of a vector.
- `round()`, `floor()`, `ceiling()`: Round to the closest integer / to the closest integer below / to the closest integer above.
- `factorial()`: Factorial function.

Example

```
> x <- c(6,3,4,1,5)
> min(x)
[1] 1
> which.min(x)
[1] 4
> which.max(x)
[1] 1

> sum(x)
[1] 19
> prod(x)
[1] 360

> cumsum(x)
[1]  6  9 13 14 19
> cumprod(x)
[1]   6  18  72  72 360

> factorial(x)
[1] 720   6  24   1 120
```

Calculus

- Differentiation:
  - `D(`*expression(f), 'x'*`)`: Derivative of $f(x)$ w.r.t x.
  - `eval(`*D object*`)`: Derivative values at x.
- Integration:
  - `integrate(`*f, a, b*`)`: $\int_a^b f(x)dx$.

```
> y <- D(expression(exp(x^2)),'x'); y
exp(x^2) * (2 * x)
> x <- c(1, 1.2)
> eval(y)
[1]  5.436564 10.129670

> f <- function(x) 2*x^2
> integrate(f,0,1)
0.6666667 with absolute error < 7.4e-15
```

Functions for Statistical Distributions

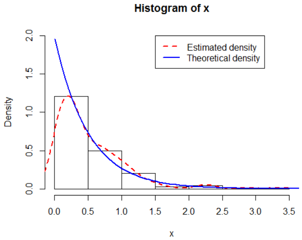
Functions for Statistical Distributions

	pdf/pmf	cdf	quantile	random #
Uniform	`dunif()`	`punif()`	`qunif()`	`runif()`
Normal	`dnorm()`	`pnorm()`	`qnorm()`	`rnorm()`
Exponential	`dexp()`	`pexp()`	`qexp()`	`rexp()`
Binomial	`dbinom()`	`pbinom()`	`qbinom()`	`rbinom()`
t	`dt()`	`pt()`	`qt()`	`rt()`
Chi-square	`dchisq()`	`pchisq()`	`qchisq()`	`rchisq()`

Example

```
# Exponential distribution
> x <- rexp(200,rate=2);   mean(x)
[1] 0.4749908

> hist(x,freq=F,ylim=c(0,2))
> lines(density(x),col='red',lty=2,lwd=2)
> x <- seq(0.01,5,0.05)
> y <- dexp(x,rate=2)
> lines(x,y,type='l',col='blue',lwd=2)
> legend(1.5,2,c('Estimated density','Theoretical density'),
+        col=c('red','blue'),lty=c(2,1),lwd=c(2,2))
```



Example

```
> # P(X < 1), where X ~ Exp(rate)
> pexp(1,rate=2)
[1] 0.8646647
> # Quantile: x value satisfying P(X < x) = p (e.g., p = 0.5)
> qexp(0.5,rate = 2)
[1] 0.3465736

> # Binomial distribution
> x <- rbinom(500,10,0.3)  # X ~ binomial(n=10,p=0.3)
> mean(x)
[1] 3.034
> var(x)
[1] 2.089022

> dbinom(2,10,0.3)     # P(X=2)
[1] 0.2334744
> pbinom(2,10,0.3)     # P(X <= 2)=p(0)+p(1)+p(2)
[1] 0.3827828
> qbinom(0.5,10,0.3)  # x satisfying P(X < x) <= p or P(X <= x) >= p
[1] 3
```

Set Operations

- `union(x,y)`: Union of the sets `x` & `y`.
- `intersect(x,y)`: Intersection of the sets `x` & `y`.
- `setdiff(x,y)`: Set difference between `x` & `y`, consisting of all elements of `x` that are not in `y`.
- `setequal(x,y)`: Test for equality between `x` & `y`.
- `c %in% x`: Membership, testing whether `c` is an element of the set `y`.

- `choose(n,k)`: Number of possible subsets of size `k` chosen from a set of size `n`.
- `combn(x,k)`: All subsets with size `k` of the set `x`.

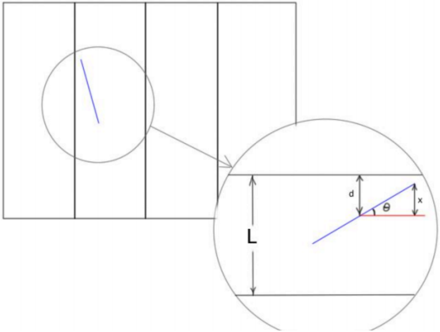
Example

```
> x <- c(1,2,7);    y <- c(5,1,6,7)

> union(x,y)
[1] 1 2 7 5 6
> intersect(x,y)
[1] 1 7
> setdiff(x,y)
[1] 2
> setequal(x,y)
[1] FALSE
> setequal(x,c(7,2,1))
[1] TRUE
> 2 %in% x
[1] TRUE
> choose(5,3)
[1] 10
> combn(x,2)
     [,1] [,2] [,3]
[1,]    1    1    2
[2,]    2    7    7
```

Buffon's Needle Problem (Simulation)

Buffon's needle problem



- Buffon, a French naturalist in the 18th centaury introduced a probability question. The problem asks to find the probability that a needle of length L will land on a line, given a floor with equally spaced parallel lines at distance L apart.

Buffon's Needle Problem (Simulation)

- L: The length of the needle and the distance between the lines on the floor.
- Suppose we throw the needle at an angle θ considered to be between 0 and π by symmetry (i.e., $0 \leq \theta < \pi$).
- d: the distance from the center of the needle to the closest line ($0 \leq d \leq L/2$).
- If $d < x$, the needle hits the line.
- $x = \frac{L}{2} \sin \theta$.
- Probability that the needle hits the line: $P(d \leq \frac{L}{2} \sin \theta)$.

Buffon's Needle Problem (Simulation)

- Solution: Geometrically, consider the graph $d = \frac{L}{2}\sin\theta$.

$$P(d \le \frac{L}{2}\sin\theta) = \frac{\int_0^{\pi} \frac{L}{2}\sin\theta d\theta}{\frac{L}{2}\pi} = \frac{\frac{L}{2}[-cos\theta]_0^{\pi}}{\frac{L}{2}\pi} = \frac{2}{\pi}.$$

- Simulation:
  - $d \sim Uniform(0, \frac{L}{2})$ & $\theta \sim Uniform(0, \pi)$.
  - d and θ are independent.

```
> buffon <- function(n,L)
+ # n: # of replications
+ # L: The length of the needle & the distance between lines
+ {
+   d <- runif(n,0,L/2)
+   theta <- runif(n,0,pi)
+   x <- (L/2)*sin(theta)
+   p <- sum(d <= x) / n
+   return(p)
+ }
```

Buffon's Needle Problem (Simulation)

```
> 2/pi    # True value
[1] 0.6366198

> buffon(100,3)
[1] 0.59

> buffon(1000,3)
[1] 0.649

> buffon(10000,3)
[1] 0.6362

> buffon(50000,3)
[1] 0.63642
```

Confidence Interval

- The meaning of 95% confidence interval: If we draw 100 random samples from the population and construct 95% confidence intervals for each sample, then about 95 confidence intervals include the parameter value.

- Random sample from the population with normal distribution.

- $(1-\alpha)100$ % confidence interval for the population mean μ:

$$\bar{x} \pm t_{(n-1,\alpha/2)} \frac{s}{\sqrt{n}},$$

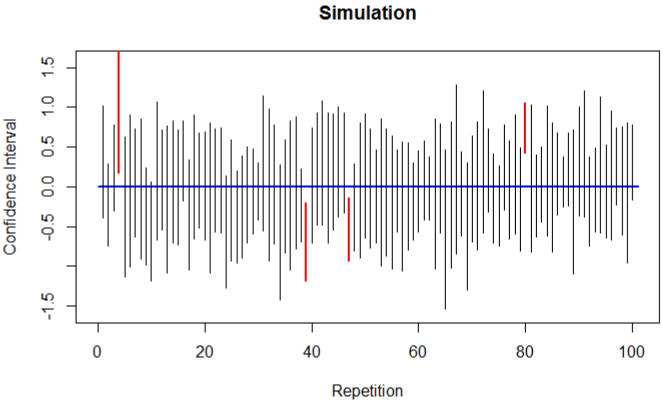
where $\bar{x}$ and s are the sample mean and standard deviation, respectively, and n is the sample size.

Confidence Interval

```
> cfv <- function(n, R=100, mu=0, sd=1, alpha=0.05)
+ {
+   B <- 5 * sd/sqrt(n)
+   K <- 0
+   plot(c(0,R+1), c(mu,mu), type='l', col='blue',
+        lwd=2, ylim=c(mu-B,mu+B), xlab='Repetition',
+        ylab='Confidence Interval', main='Simulation')
+   for (i in 1:R)
+   {
+     x <- rnorm(n,mean=mu,sd=sd)
+     LB <- mean(x) + qt(alpha/2,n-1) * sd(x) / sqrt(n)
+     UB <- mean(x) + qt(1-alpha/2,n-1) * sd(x) / sqrt(n)
+     if (LB > mu | UB < mu) lines(c(i,i),c(LB,UB),col='red',lwd=2)
+     else {
+       lines(c(i,i),c(LB,UB))
+       K = K + 1  }
+     Sys.sleep(0.25)
+   }
+   sprintf('Coverage Probability: %.3f',K/R)
+ }
```

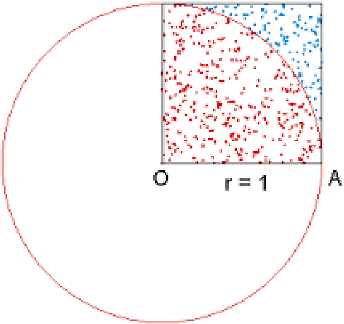
Confidence Interval

```
> cfv(n=10,R=100)
[1] "Coverage Probability: 0.960"
```



Estimating π

- Estimating π.
- Two uniform (0,1) random numbers x and y.
- $x^2 + y^2 \le 1 \Rightarrow \pi/4$.



ESTIMATING π

```
> compute.pi <- function(r)
+ {
+   x <- runif(r)
+   y <- runif(r)
+   z <- (x^2 + y^2) <= 1
+   return(4*sum(z)/r)
+ }
>
> compute.pi(10000)
[1] 3.1416
```

Estimating π

```
> com.pi <- function(r)
+ {
+   plot(c(0,r),c(pi,pi),type='l',xlim=c(1,r),
+        ylim=c(0,5),xlab='R',ylab='pi',col='red')
+   m <- 0
+   old.pi <- 0
+   for (i in 1:r)
+   {
+     x <- runif(2)
+     if((x[1]^2 + x[2]^2) <= 1) m = m + 1
+     new.pi <- 4*m/i
+     lines(c(i-1,i),c(old.pi,new.pi),lwd=2,col='blue')
+     old.pi <- new.pi
+     Sys.sleep(0.05)
+   }
+   return(4*m/r)
+ }
>
> com.pi(1000)
[1] 3.172
```

ESTIMATING π