Ch2. Vectors

Jongrak Jeong

2023-01-05

1. more on vectors

Scalar: the vector whose length is 1

```
# define new scalar
x <- 1
is.vector(x)
```

```
## [1] TRUE
```

```
# built-in scalar
pi
```

```
## [1] 3.141593
```

```
print(pi, 16)
```

```
## [1] 3.141592653589793
```

Scalar example 1: Fibonacci sequence

```
# define the function
f <- vector(length = 10)
f[1] <- 1
f[2] <- 1

for (i in 3:10) f[i] <- f[i - 1] + f[i - 2]

f
```

```
##  [1]  1  1  2  3  5  8 13 21 34 55
```

Matrix: 2nd dimension vector

```
# matrix(a vector, the length of row, the length of column, by row or not)
# example: convert a vector with length 12 to a 4 x 3 matrix
# default is 'column-wise' or 'by column'
M <- matrix(1:12, nrow = 4, ncol = 3)

M
```

```
##      [,1] [,2] [,3]
## [1,]    1    5    9
## [2,]    2    6   10
```

```
## [3,]    3    7   11
## [4,]    4    8   12
```

```
# 'row-wise' or 'by row'
M.1 <- matrix(1:12, nrow = 4, ncol = 3, byrow = T)

M.1
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
## [4,]   10   11   12
```

```
# recycling (when the length of source vector is smaller than the length that a matrix needs)
M.2 <- matrix(1:10, nrow = 4, ncol = 3)
```

```
## Warning in matrix(1:10, nrow = 4, ncol = 3): data length [10] is not a sub-
## multiple or multiple of the number of rows [4]
```

```
M.2
```

```
##      [,1] [,2] [,3]
## [1,]    1    5    9
## [2,]    2    6   10
## [3,]    3    7    1
## [4,]    4    8    2
```

```
M.3 <- matrix(c(1:10, 1, 2), nrow = 4, ncol = 3)

M.3
```

```
##      [,1] [,2] [,3]
## [1,]    1    5    9
## [2,]    2    6   10
## [3,]    3    7    1
## [4,]    4    8    2
```

Indexing

```
f
```

```
##  [1]  1  1  2  3  5  8 13 21 34 55
```

```
f[10]
```

```
## [1] 55
```

```
f[3:10]
```

```
## [1]  2  3  5  8 13 21 34 55
```

```
# except the first and second elements
f[-c(1,2)]
```

```
## [1]  2  3  5  8 13 21 34 55
```

```
# call specific elements
f[c(2, 4, 6, 8, 10)]
```

```
## [1]  1  3  8 21 55
```

2. Special function, special value, and filtering

seq()

```
# seq(from, to, by = n)
seq(0, 10, by = 2.5)
```

```
## [1]  0.0  2.5  5.0  7.5 10.0
```

```
# seq(from, to, length = n)
seq(0, 10, length = 10)
```

```
##  [1]  0.000000  1.111111  2.222222  3.333333  4.444444  5.555556  6.666667
##  [8]  7.777778  8.888889 10.000000
```

rep()

```
# rep(x, times = k) x can be a scalar or vector
rep(0, times = 10)
```

```
##  [1] 0 0 0 0 0 0 0 0 0 0
```

```
rep(NA, 10)
```

```
##  [1] NA NA NA NA NA NA NA NA NA NA
```

```
rep(c(1, 2, 3), 2)
```

```
## [1] 1 2 3 1 2 3
```

```
# k also can be a vector
rep(c(1, 2, 3), c(1, 2, 3)) # 1 is repeated 1 time, 2 is repeated 2 times, and 3 is repeated 3 times
```

```
## [1] 1 2 2 3 3 3
```

NULL: not exist / NA: not available

```
# NULL is 'nothing'
x <- c(1, 2, NULL, 6) # put 4 things

x
```

```
## [1] 1 2 6
```

```
mean(x)
```

```
## [1] 3
```

```
length(x) # but the length is 3 in that there's one NULL out of foul elements
```

```
## [1] 3
```

```
# NA is not 'nothing' but just 'not available' - missing
x <- c(1, 2, NA, 6) # put 4 things

x
```

```
## [1]  1  2 NA  6
```

```r
mean(x) # the average value can not be calculated
```

```
## [1] NA
```

```r
length(x) # and the length is 4
```

```
## [1] 4
```

```r
mean(x, na.rm = T) # omit NA and calculate the average value
```

```
## [1] 3
```

Filtering

```r
# define a vector
z <- c(1, 8, 9, 2, 7, 10, 5, 6, 4, 3)

index <- (z %% 2) == 1
index
```

```
##  [1]  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE FALSE  TRUE
```

```r
z[index]
```

```
## [1] 1 9 7 5 3
```

```r
z[(z %% 2) == 1]
```

```
## [1] 1 9 7 5 3
```

```r
subset(z, (z %% 2) == 1)
```

```
## [1] 1 9 7 5 3
```

3. ifelse(), identical(), all(), names()

ifelse(): yes or no

```r
# ifelse(test, yes, no) / test, yes, and no are all vectors

# example: if x > 0, then print x. Otherwise, print 2x
x <- c(-0.6, 0.2, -0.8, 1.6, 0.3, -0.8, 0.5, 0.7, 0.6, -0.3)

x.1 <- ifelse(x > 0, x, 2 * x)
x.1
```

```
##  [1] -1.2  0.2 -1.6  1.6  0.3 -1.6  0.5  0.7  0.6 -0.6
```

```r
rbind(x, x.1)
```

```
##     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## x   -0.6  0.2 -0.8  1.6  0.3 -0.8  0.5  0.7  0.6  -0.3
## x.1 -1.2  0.2 -1.6  1.6  0.3 -1.6  0.5  0.7  0.6  -0.6
```

```r
# ifelse() help us to make simpler code

# basic code with 'for'
x.1 <- x
for (i in 1:length(x)) {
```

```
  if (x[i] > 0) x.1[i] <- x[i] else x.1[i] <- 2 * x[i]
}

rbind(x, x.1)
```

```
##     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## x   -0.6  0.2 -0.8  1.6  0.3 -0.8  0.5  0.7  0.6  -0.3
## x.1 -1.2  0.2 -1.6  1.6  0.3 -1.6  0.5  0.7  0.6  -0.6
```

identical()

```
# k1:k3 and c(k1, k2, k3) are different
x <- 1:3 # integer
y <- c(1, 2, 3) # numeric

identical(x, y)
```

```
## [1] FALSE
```

```
# c(k1, k2, k3) and seq(k1, k2, 1) are same
z <- seq(1, 3, 1) # numeric

identical(y, z)
```

```
## [1] TRUE
```

all(): Do all the things in specific vector satisfy the condition? Yes or no

```
# it returns TRUE only when all elements satisfy the condition
all(x == y)
```

```
## [1] TRUE
```

```
# x is integer vector but it is converted to numeric so can be compare with y.
```

names() <- : assign specific names to each element

```
era <- c(5, 4, 3, 4, 5, 6)
names(era) # NULL
```

```
## NULL
```

```
# assign names
# paste is combine two character strings paste(c, c, sep = "seperator")
names(era) <- paste("y", 2001:2006, sep = ".")

era
```

```
## y.2001 y.2002 y.2003 y.2004 y.2005 y.2006
##      5      4      3      4      5      6
```

```
era[2] # indexting with the order
```

```
## y.2002
##      4
```

```
era["y.2002"] # indexing with the name
```

```
## y.2002
##      4
```

4. arithmetic and relation operators

application: run lengths

```
# define the binomial sample
set.seed(1)
z <- rbinom(20, 1, 0.5)

z
```

```
##  [1] 0 0 1 1 0 1 1 1 1 0 0 0 1 0 1 0 1 1 0 1
```

```
# If the difference between two elements are same, run. otherwise, run is over.
n <- length(z)
d <- z[2:n] - z[1:(n-1)]
d
```

```
##  [1]  0  1  0 -1  1  0  0  0 -1  0  0  1 -1  1 -1  1  0 -1  1
```

```
sum(d != 0) + 1 # how many 'run's are there?
```

```
## [1] 12
```

```
index <- c(which(d != 0), length(z))
index
```

```
##  [1]  2  4  5  9 12 13 14 15 16 18 19 20
```

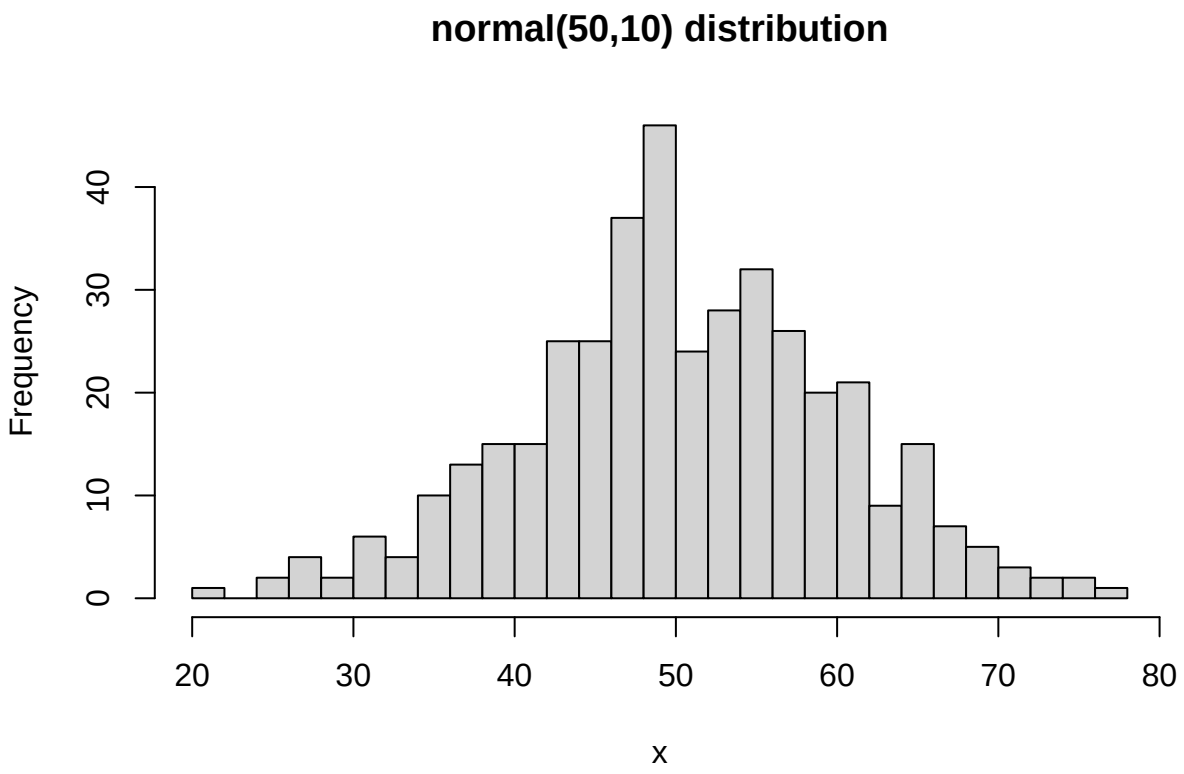
```
c(index[1], diff(index))
```

```
##  [1] 2 2 1 4 3 1 1 1 1 2 1 1
```

5. random number generation

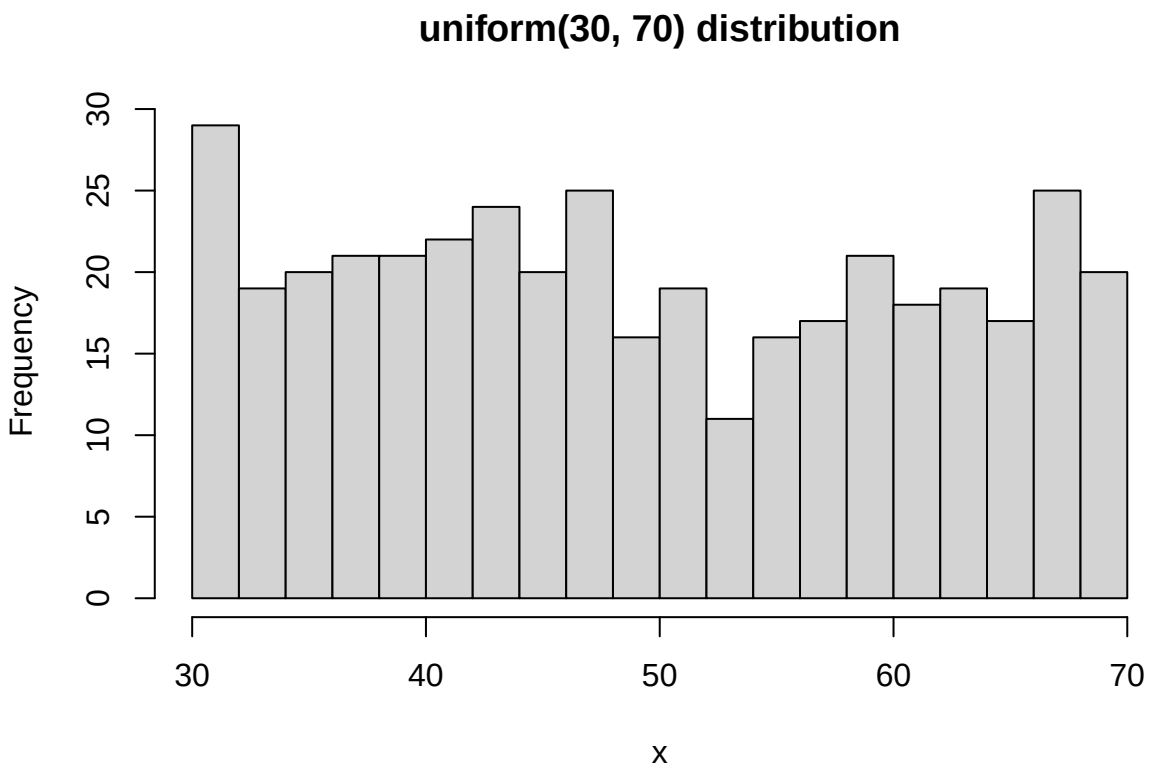
normal distribution

```
n <- 400
set.seed(1)
x <- rnorm(n, 50, 10)

hist(x, nclass = 20, main = "normal(50,10) distribution")
```



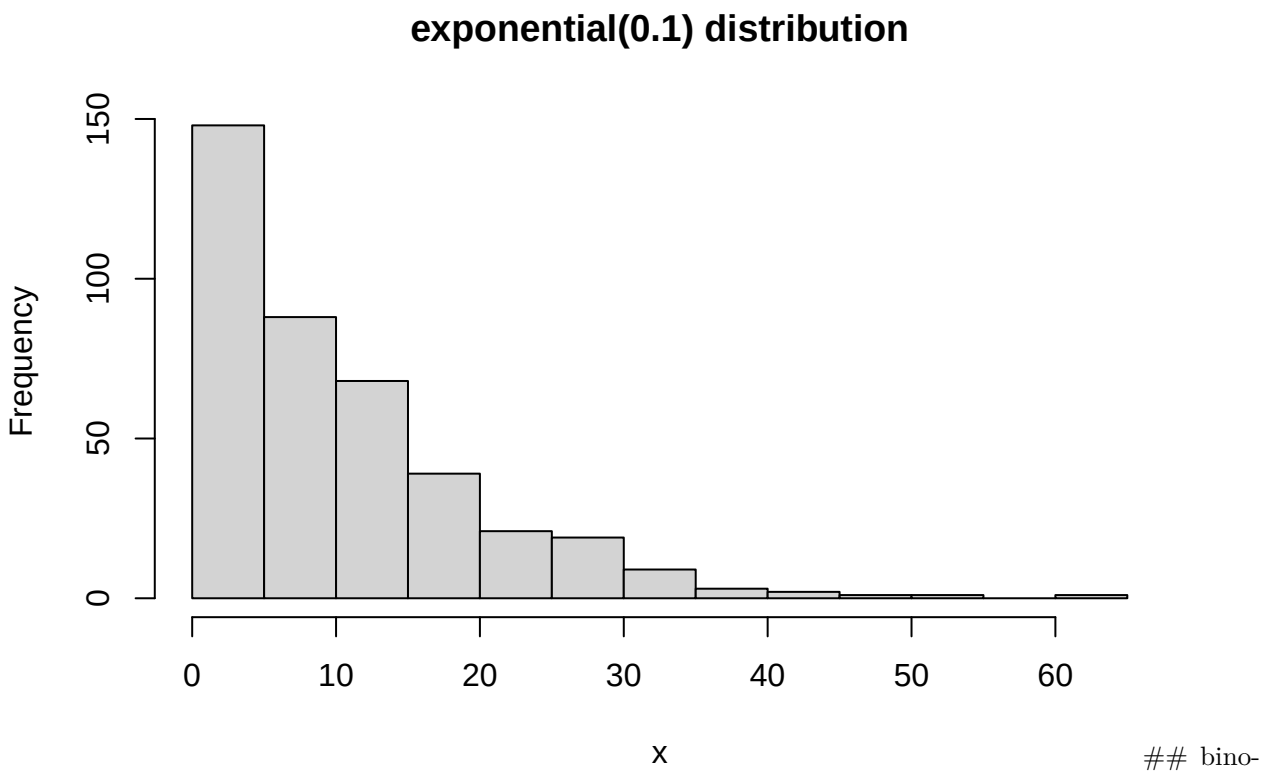
uniform distribution

```
n <- 400
x <- runif(n, 30, 70)

hist(x, nclass = 20, main = "uniform(30, 70) distribution")
```



exponential distribution

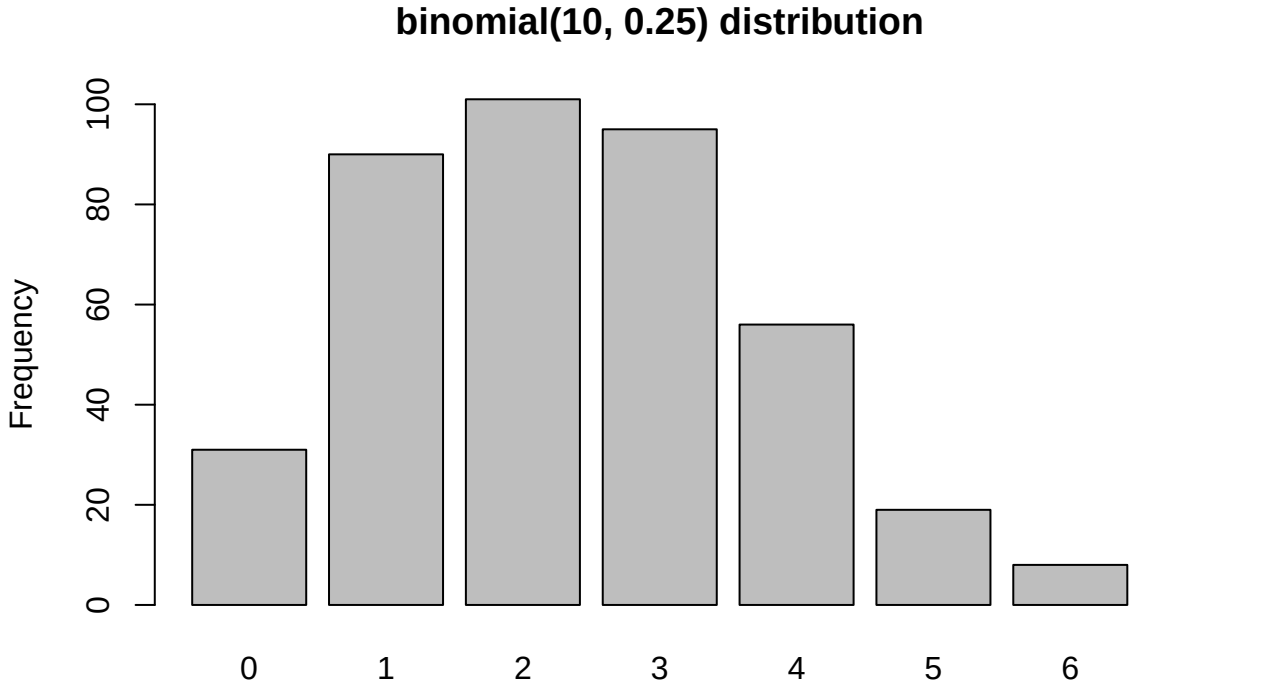
```
n <- 400
x <- rexp(n, 0.1)

hist(x, nclass = 20, main = "exponential(0.1) distribution")
```



bino-

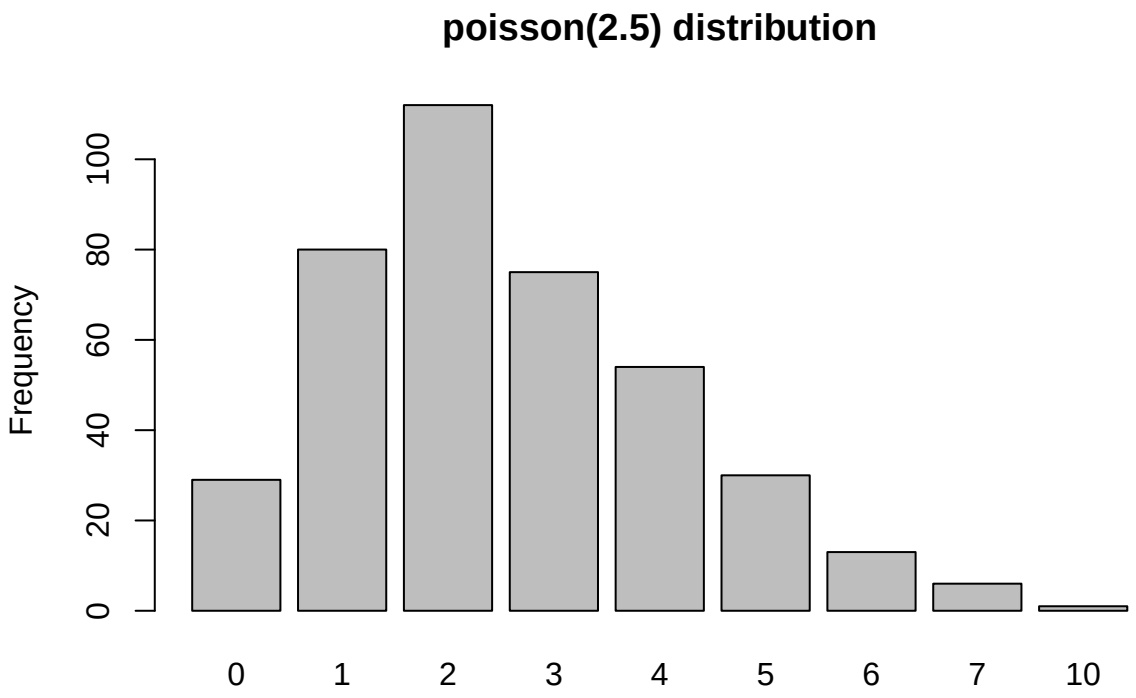
mial distribution

```
n <- 400
x <- rbinom(n, 10, 0.25)

barplot(table(x), ylab = "Frequency", main = "binomial(10, 0.25) distribution")
```



Poisson distribution

```
n <- 400
x <- rpois(n, 2.5)

barplot(table(x), main = "poisson(2.5) distribution", ylab = "Frequency")
```



Random sampling from finite population: sample()

ex

```
# n = 10, without replacement -> random permutatin
sample(LETTERS[1:10])
```

```
##  [1] "I" "H" "J" "B" "G" "C" "E" "A" "F" "D"
```

```
# n = 10, with replacement
sample(LETTERS[1:10], replace = T)
```

```
##  [1] "H" "B" "A" "G" "H" "F" "E" "H" "J" "I"
```