Stat 292 - Recitation 12

Probability Distributions & Exceptions and Timings

Orçun Oltulu

16 / 6 / 2021

Probability Distributions in R

Every distribution that R handles has four functions. There is a root name, for example, the root name for the normal distribution is norm. This root is prefixed by one of the letters;

- **p** for "probability", the cumulative distribution function (cdf)
- **q** for "quantile", the inverse cdf.
- **d** for "density", the density function (pmf or pdf)
- **r** for "random", a random variable having the specified distribution

For the normal distribution, these functions are pnorm, qnorm, dnorm, and rnorm. For the binomial distribution, these functions are pbinom, qbinom, dbinom, and rbinom. And so forth.

Exercise 1:

Bob makes 60% of his free-throw attempts. If he shoots 12 free throws,

Part A:

What is the probability that he makes exactly 10? P(X = 10) = ?

```
dbinom(10, size = 12, prob = .6)
```

```
## [1] 0.06385228
```

Part B:

What is the probability that he makes more than 9? P(X > 9) = ?

```
1 - pbinom(9, size = 12, prob = .6)
```

```
## [1] 0.08344332
```

```r
# alternatively,
# pbinom(9, size = 12, prob = .6, lower.tail = F)
```

Part C:

Find the 10^{th} quantile of Bob's free-throw attempts.

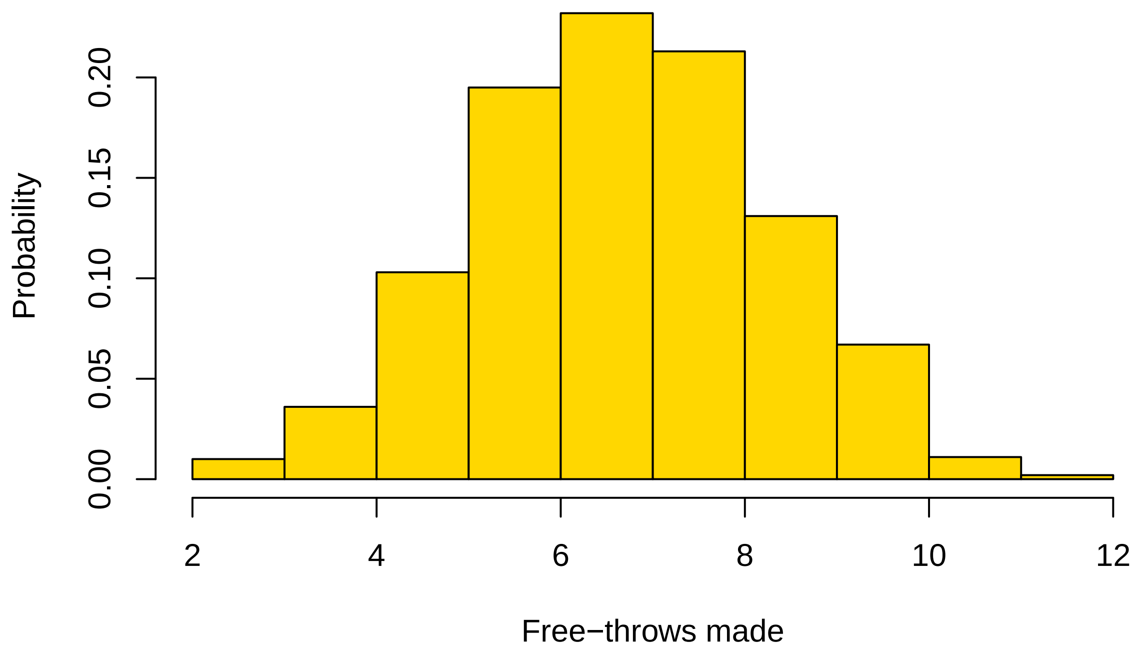
```r
qbinom(p = 0.1, size = 12, prob = .6)
```

```
## [1] 5
```

Part D:

Plot the distributional histogram for the Bob's free-throw attempts by simulating this scenario.

```r
x <- rbinom(n = 1000, size = 12, prob = .6)
hist(x, probability = TRUE, main = "",
     xlab = "Free-throws made",
     ylab = "Probability", col = "Gold")
```

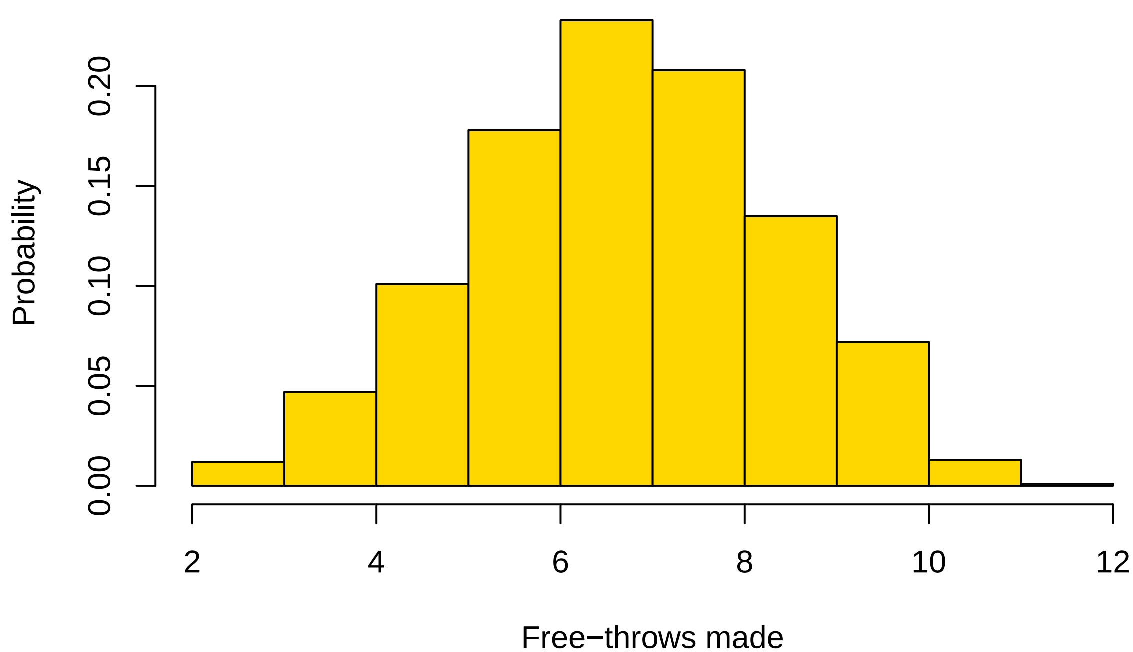


Part D:

Can you create this simulation design without using rbinom() function, instead sample() command.

```
x_sim <- numeric()
for(i in 1:1000){
 x_sim[i] <- sum(sample(0:1, size = 12, prob = c(0.4,0.6), replace = TRUE))
}

hist(x_sim, probability = TRUE, main = "",
     xlab = "Free-throws made",
     ylab = "Probability", col = "Gold")
```



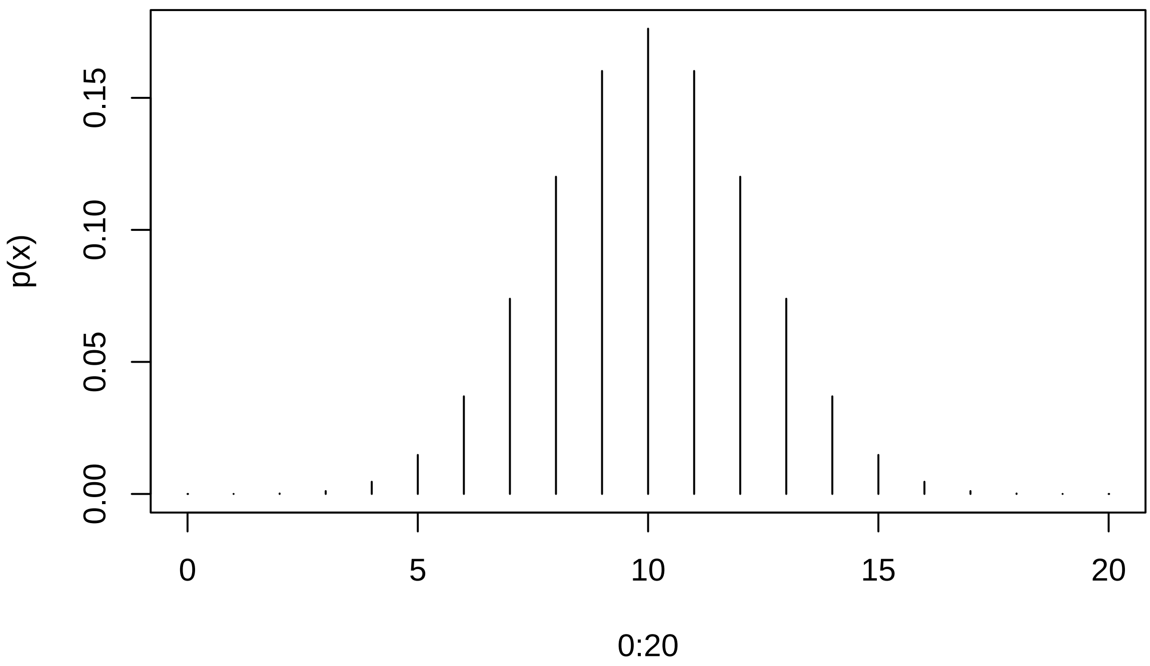
Exercise 2:

Dr. Horrible decided to give his Econ 1 students a pop quiz on Advanced Quantum Mechanics. Since he isn't completely unreasonable he made the quiz True-or-False. Since they don't know any Quantum Physics, Dr. Horrible's students guess randomly on each of the 20 questions.

Part A:

An individual student's score on this quiz can be modeled as the realization of random variable. What random variable? Plot its pmf.

```
plot(0:20, dbinom(0:20, size = 20, prob = 0.5), type = 'h', ylab = 'p(x)')
```



Part B:

Suppose that a passing grade on the quiz is a 60%. What is the probability that a given student passes?

```
pbinom(11, size=20, prob=.5, lower.tail=FALSE)
```

```
## [1] 0.2517223
```

```
# Alternatively,
# sum(dbinom(12:20, size = 20, prob = 0.5))
```

Part C:

Suppose that anything over 90% is an A. What is the probability that an individual student gets an A?

```r
pbinom(17, size = 20, prob = .5, lower.tail = FALSE)
```

```
## [1] 0.0002012253
```

```r
# Alternatively,
# sum(dbinom(18:20, size = 20, prob = 0.5))
```

Part D:

If the class has 250 students, approximately how many will pass the quiz? Approximately how many will get an A?

```r
pbinom(11, size=20, prob=.5, lower.tail=FALSE) * 250
```

```
## [1] 62.93058
```

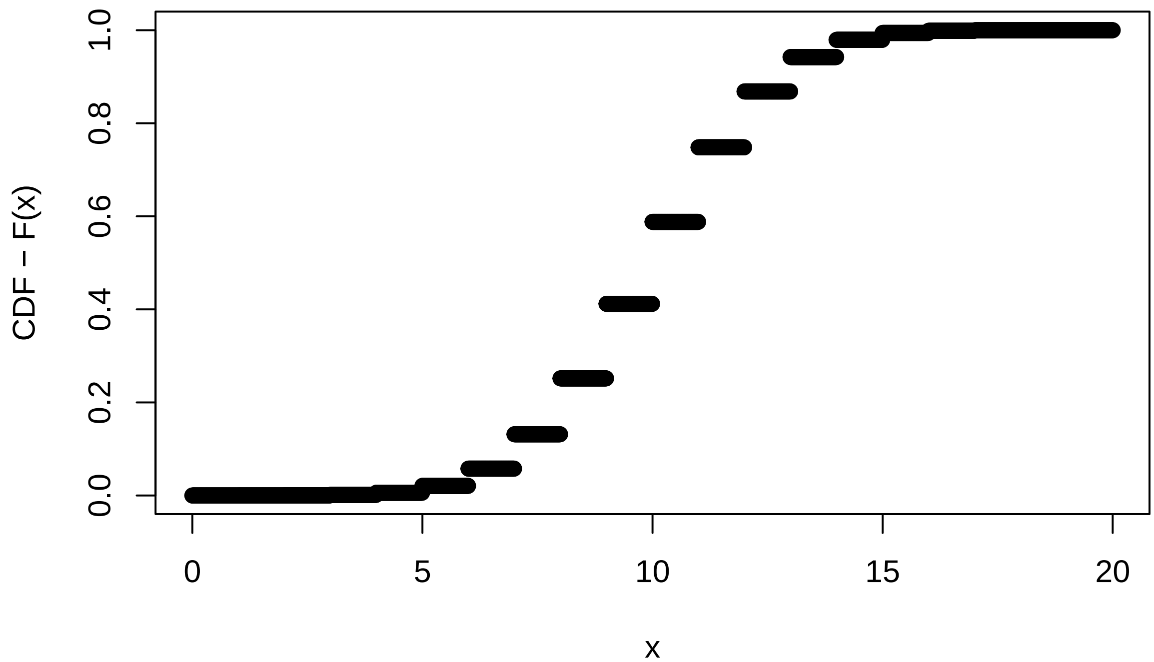
```r
pbinom(17, size = 20, prob = .5, lower.tail = FALSE) * 250
```

```
## [1] 0.05030632
```

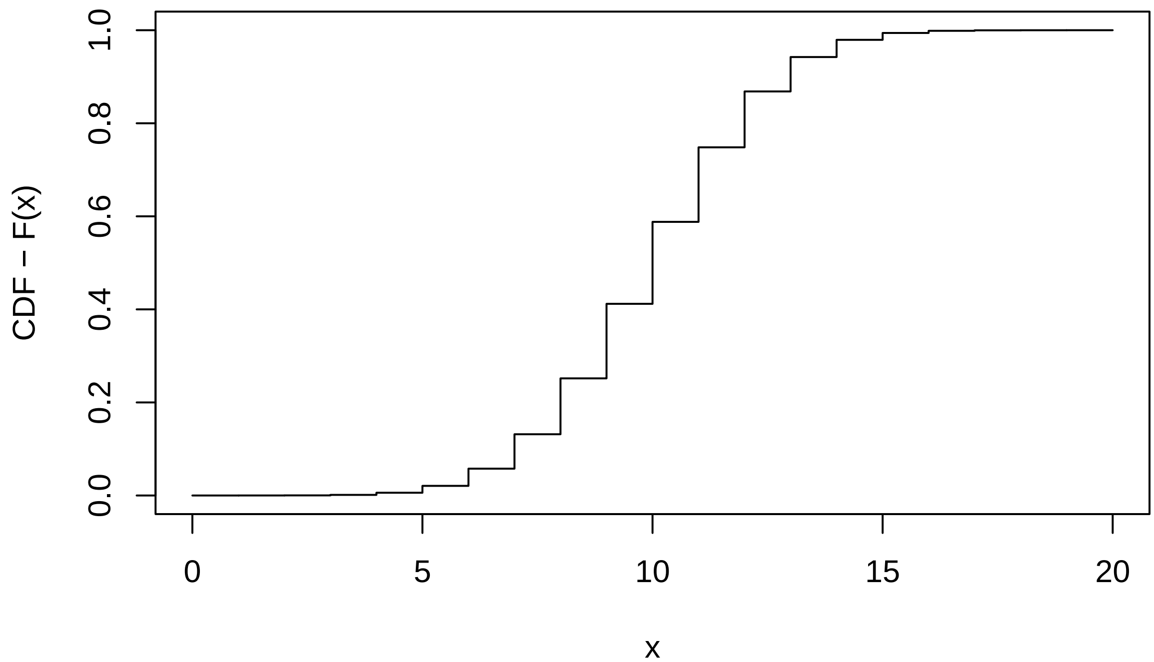
Part E:

Obtain a CDF plot for this distribution.

```r
x <- seq(from = 0, to = 20, by = 0.01)
y <- pbinom(x, size = 20, prob = .5)
plot(x, y, ylab = 'CDF - F(x)')
```




```
plot(x, y, ylab = 'CDF - F(x)', type = "s")
```



Exercise 3:

Suppose that the data concerning the first-year salaries of METU graduates is normally distributed with the population mean $\mu = 60.000$ TL and the population standard deviation $\sigma = 15.000$ TL.

Part A:

Find the probability of a randomly selected METU graduate earning more than 80.000 TL annually.

```r
pnorm(q = 80000, mean = 60000, sd = 15000, lower.tail = FALSE)
```

```
## [1] 0.09121122
```

Part B:

Find the lower bound of annual salaries of the top 15% earners of METU graduates.

```r
qnorm(p = 0.15, mean = 60000, sd = 15000, lower.tail = FALSE)
```

```
## [1] 75546.5
```

Part C:

Find the probability of a randomly selected METU graduate earning between 50.000 and 55.000 TL annually.

```
diff(pnorm(q = c(50000,55000), mean = 60000, sd = 15000))
```

```
## [1] 0.1169488
```

Part D:

Generate a new data set of size 1000, from normal distribution having the same parameters and then obtain a histogram.

```
x <- rnorm(n = 1000, mean = 60000, sd = 15000)
hist(x, probability = TRUE, main = "",
     xlab = "Salary",
     ylab = "Probability", col = "Gold")
```

Exceptions

Exercise 1:

Write an R function which calculates Euclidean distance of two given points. If the dimensions does not match then it stops.

```
distance <- function(u,v) {
  if(length(u) == length(v)){

    d <- sqrt(sum((u-v)^2))
    return(round(d,3))

  } else{

    stop("vectors must have the same length")
  }
}
```

```
distance(u = 1:3, v = 3:9)
```

```
distance(u = c(2,2,-1), v = c(2,-1,5))
```

```
## [1] 6.708
```

Exercise 2:

Write a function that takes a real number (or a vector of numbers) x and a nonnegative integer n, and returns the value of the sum

$$1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^n}{n!} = \sum_{i=0}^{n} \frac{x^i}{i!}$$

If the parameter x is a vector, the function's output should be a vector.

If the parameter n is negative, the function should display warning message "parameter n should not be negative" and takes absolute value of n then uses abs(n) for the calculations.

```
powerseries <- function(x, n){

  if(n < 0){
    warning(paste("Given value n =",n,
                  "is not a positive number. \n
                  abs(n) will be used \n"))
    n <- abs(n)
  }
    out_power_series <- numeric()

    for(i in 1:length(x)){
```

```
      out_power_series[i] <- sum(x[i]^(0:n) / factorial(0:n))
    }

    return(out_power_series)

}
powerseries(x=1, n=7)
powerseries(x=2, n=-1)
```

Timings

Part A:

Generate 1000000 random numbers from standard normal distribution and sort them in increasing order. Calculate the run time.

```
start <- Sys.time()

x <- sort(rnorm(n = 1000000))

end <- Sys.time()

end - start
```

```
## Time difference of 0.09773898 secs
```

```
# alternatively,
system.time({
  sort(rnorm(n = 1000000))
})
```

```
##    user  system elapsed 
##    0.08    0.03    0.11
```

Part B:

Now, create a for loop with 50 iterations where each iteration sleeps system 0.1 seconds (Sys.sleep(0.1)) and have a Progress Bar to watch the progress of the loop.

```
n_iter <- 50 # Number of iterations of the loop

# Initializes the progress bar
pb <- txtProgressBar(min = 0,
                     max = n_iter,
                     style = 3,
```

```
                     width = 50,
                     char = "=")
```

```
##   |                                                          |
```

```
for(i in 1:n_iter) {

    #----------------------
    # Code to be executed
    #----------------------

    Sys.sleep(0.1) # Remove this line and add your code

    #----------------------

    # Sets the progress bar to the current state
    setTxtProgressBar(pb, i)
}
```

```
##   |                                                          |=
```

```
close(pb) # Close the connection
```

```
# alternatively

n_iter <- 50 # Number of iterations

pb <- winProgressBar(title = "Windows progress bar", # Window title
                     label = "Percentage completed", # Window label
                     min = 0,      # Minimum value of the bar
                     max = n_iter, # Maximum value of the bar
                     initial = 0,  # Initial value of the bar
                     width = 300L) # Width of the window

for(i in 1:n_iter) {

    Sys.sleep(0.1) # Remove this line and add your code


    pctg <- paste(round(i/n_iter * 100, 0), "% completed")
    setWinProgressBar(pb, i, label = pctg)

}

beepr::beep(5)
```

```r
close(pb) # Close the connection
```

```
## NULL
```