25102023_Statistical_Learning

Mattia G.

2023-10-25

R Markdown

for a better experience I suggest to input the following code chunks into RStudio.

```
#######################################
#    PROBABILITY DISTRIBUTIONS
# -- discrete Random Variables --
#######################################

# sample from a URN of uppercase letters

urn <- LETTERS

sample(urn, 1)
```

```
## [1] "S"
```

```
sample(urn, 5)
```

```
## [1] "U" "M" "P" "Y" "V"
```

```
urn <- LETTERS[1:5]
sample(urn, 5)
```

```
## [1] "D" "B" "E" "C" "A"
```

```
sample(urn, 5, replace=TRUE)
```

```
## [1] "E" "C" "E" "B" "C"
```

```
# Random seed: state of the random number generator in R
# set.seed(): function to specify  seeds

set.seed(123)
sample(urn, 1)
```

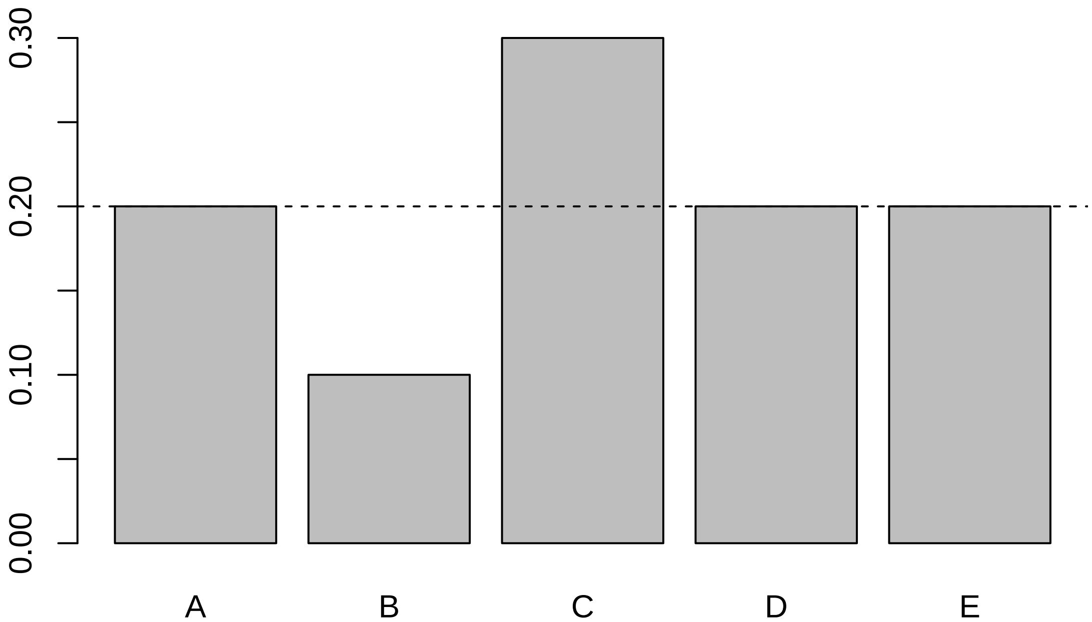
```
## [1] "C"
```

```
sample(urn, 5)
```

```
## [1] "C" "B" "D" "E" "A"
```

```
# compare empirical behavior with expected behavior

n <- 10 # increase up to 10^6
my.sample <- sample(urn, n, replace=TRUE)
```

```r
barplot(table(my.sample)/n)
abline(h=0.20, lty=2)
```




```r
###################################
# the Bernoulli random variable
###################################

urn <- c(rep(1, 7), rep(0,3))
urn
```

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

```r
sample(urn, 1)
```

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

```r
# (population) parameters

mu <- 0.7
sigma2 <- 0.7*0.3
sigma2
```

```
## [1] 0.21
```

```r
sigma <- sqrt(sigma2)
sigma
```

```
## [1] 0.4582576
```

```r
# sample

n <- 5
n <- 100
n <- 100000
out <- sample(urn, n, replace = TRUE)

# sample statistics
```

```r
x.bar <- mean(out)
x.bar
```

```
## [1] 0.69823
```

```r
mu-x.bar
```

```
## [1] 0.00177
```

```r
s2 <- var(out)
s2
```

```
## [1] 0.210707
```

```r
sigma2-s2
```

```
## [1] -0.0007069742
```

```r
####################################
# the binomial distribution
####################################

n <- 10
out <- sample(urn, n, replace=TRUE)
x <- sum(out)
x
```

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

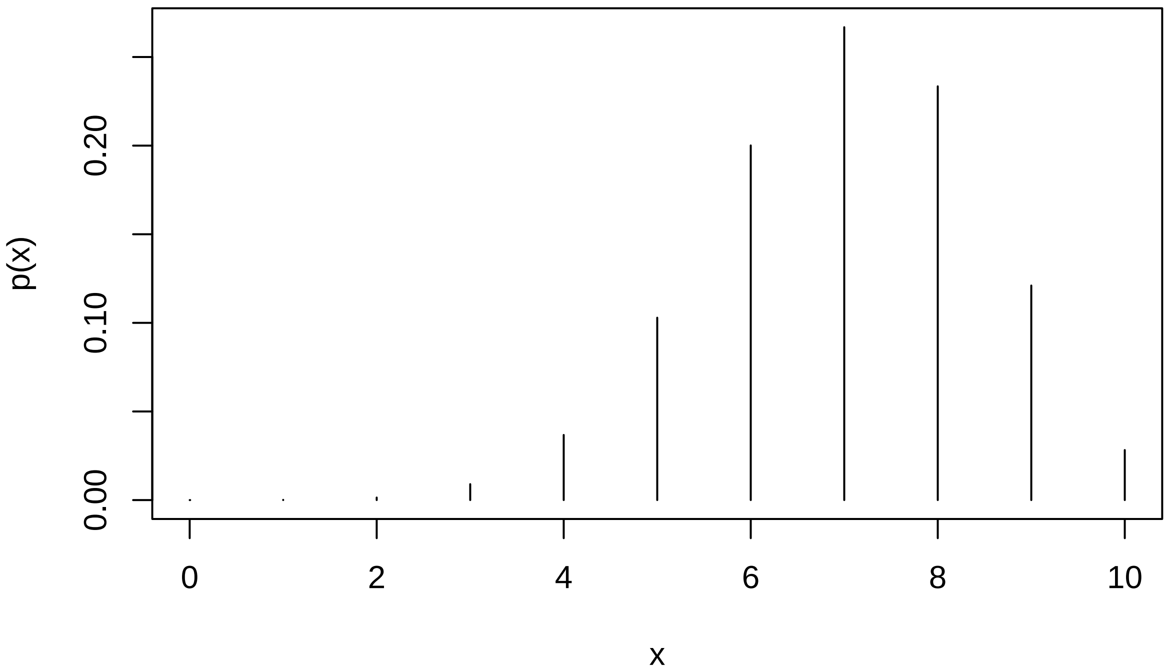
```r
# rbinom() function

# binomial
rbinom(4, size=10, prob=0.7)
```

```
## [1] 6 8 6 5
```

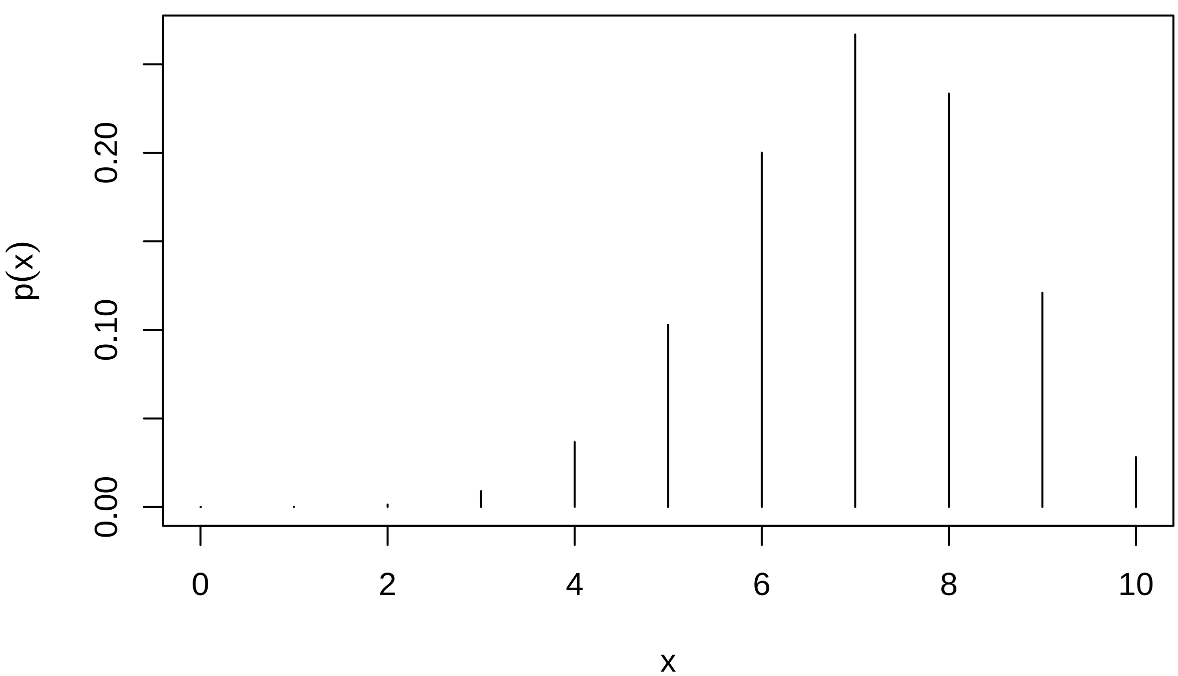
```r
# Bernoulli as a special case of the binomial distribution
rbinom(1, size=1, prob=0.7)
```

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

```r
# binomial pdf and cdf

p <- 0.7
n <- 10
x <- 0:n


# probability mass function
pdf <- dbinom(x, n, p)

# a plot of the probability mass function

plot(x, pdf, xlab = "x", ylab = "p(x)", type = "h")
```




```
# function expression() to improve the quality of
# axis label (xlab and ylab)

plot(x, pdf, xlab = expression(x), ylab = expression(p(x)), type = "h")
```




```
# rules for using the expression() function
# demo(plotmath)


# cumulative distribution function
cdf <- pbinom(x, n, p)
```

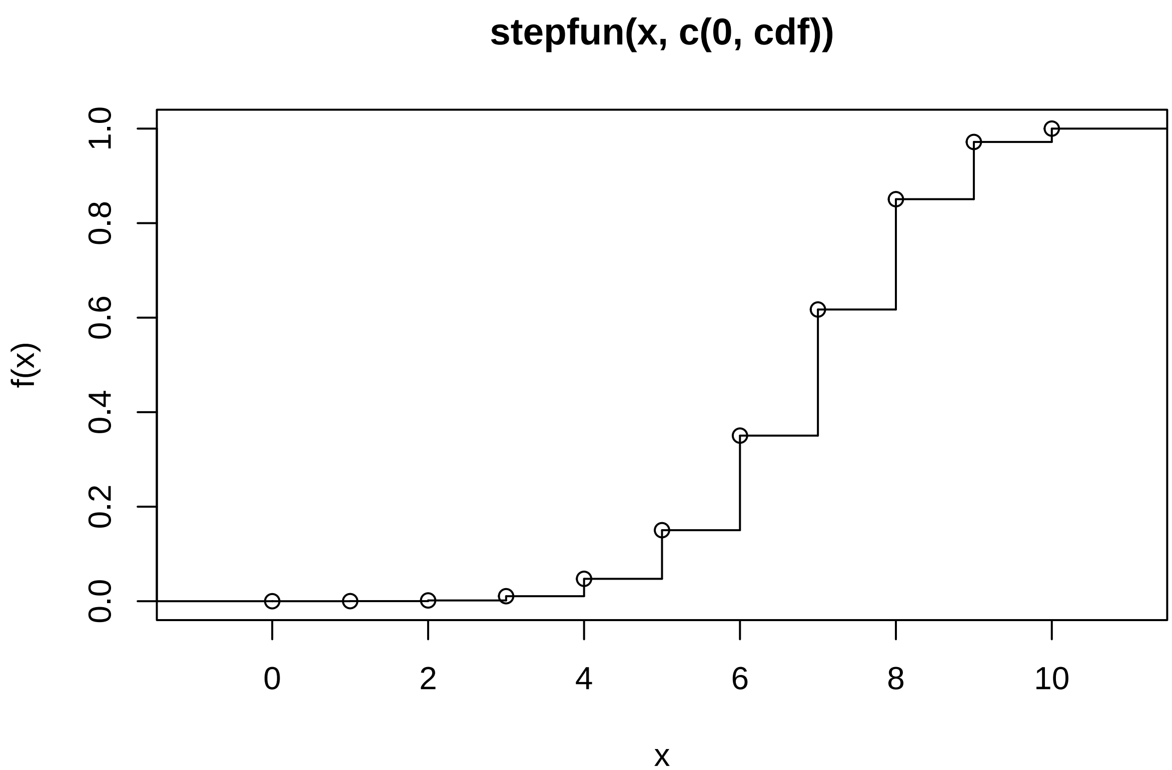
```r
# for discrete distribution compute the cdf as
# cumulative sum of the probability mass function values
cumsum(pdf)
```

```
##  [1] 0.0000059049 0.0001436859 0.0015903864 0.0105920784 0.0473489874
##  [6] 0.1502683326 0.3503892816 0.6172172136 0.8506916541 0.9717524751
## [11] 1.0000000000
```

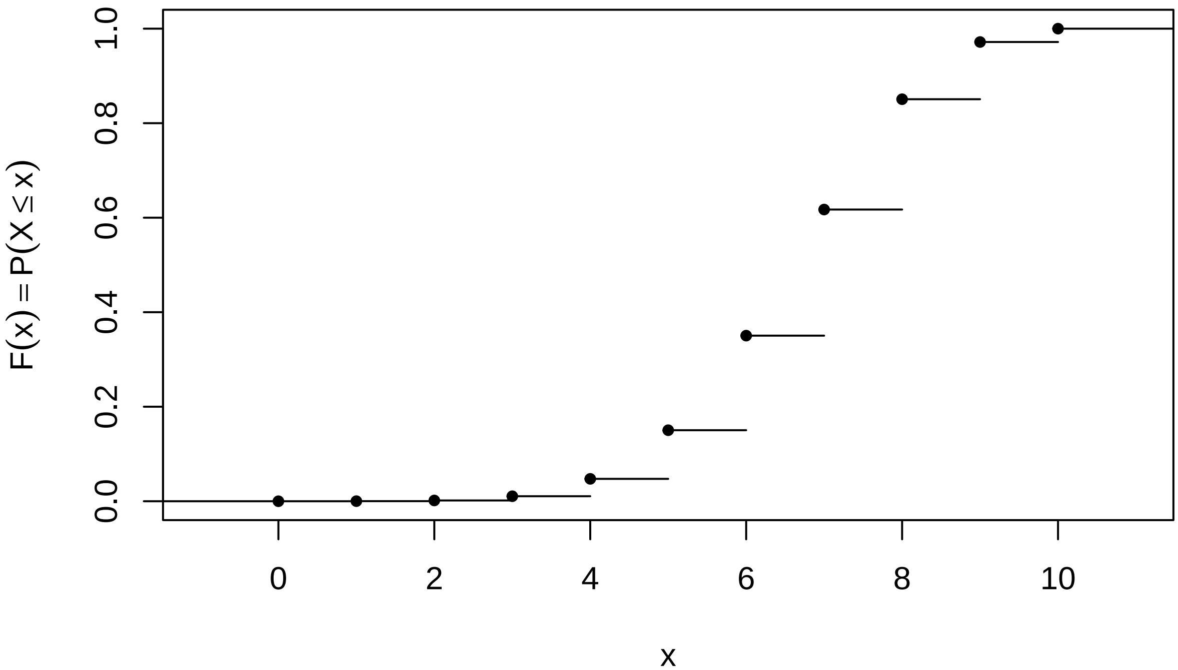
```r
# check that it coincides with the cdf from the function pbinom()
max(abs(cumsum(pdf)-cdf))
```

```
## [1] 2.220446e-16
```

```r
# plot the cumulative distribution function
#

# basic step function
n <- 10
x <- 0:n
cdf <- pbinom(x, n, p)
out.step <- stepfun(x, c(0, cdf))

plot(out.step)
```




```r
# nicer output with more graphical parameters

plot(out.step, pch=20, vertical=FALSE, xlab = expression(x),  ylab = expression(F(x)==P(X<=x)), main =
```




```r
# EXAMPLE: AA Airlines

1 - pbinom(58, 64, 0.8)
```

```
## [1] 0.006730152
```

```r
# user defined functions

my.square <- function(a){
  b <- a*a
  return(b)
}

my.square
```

```
## function(a){
##   b <- a*a
##   return(b)
## }
```

```r
my.square(2)
```

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

```r
x <- my.square(2)
x
```

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

```r
# "for" loops

# example 1
for(i in 1:5) print(i)
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
```

```r
# example 2
x <- rbinom(5, size=10,  p=0.5)
for (k in x) print(k)
```

```
## [1] 5
## [1] 5
## [1] 6
## [1] 3
## [1] 5
```

```r
# example 3
superheroes <- c("superman", "batman", "spiderman")
for (name in superheroes ) print(name)
```

```
## [1] "superman"
## [1] "batman"
## [1] "spiderman"
```

```r
# for loop in a function

my.power <- function(a, power = 2) {
  b <- 1
  for(i in 1:power) {
    b <- b*a
  }
  return(b)
}

my.power(2)
```

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

```r
my.power(2, 2)
```

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

```r
my.power(2, 3)
```

```
## [1] 8
```

```r
my.power(2, 10)
```

```
## [1] 1024
```

```r
# apply a function to a vector

a <- c(1, 3, 5, 7)

# many (but not all) functions
# work also with vectors

x <- my.square(a)
x
```

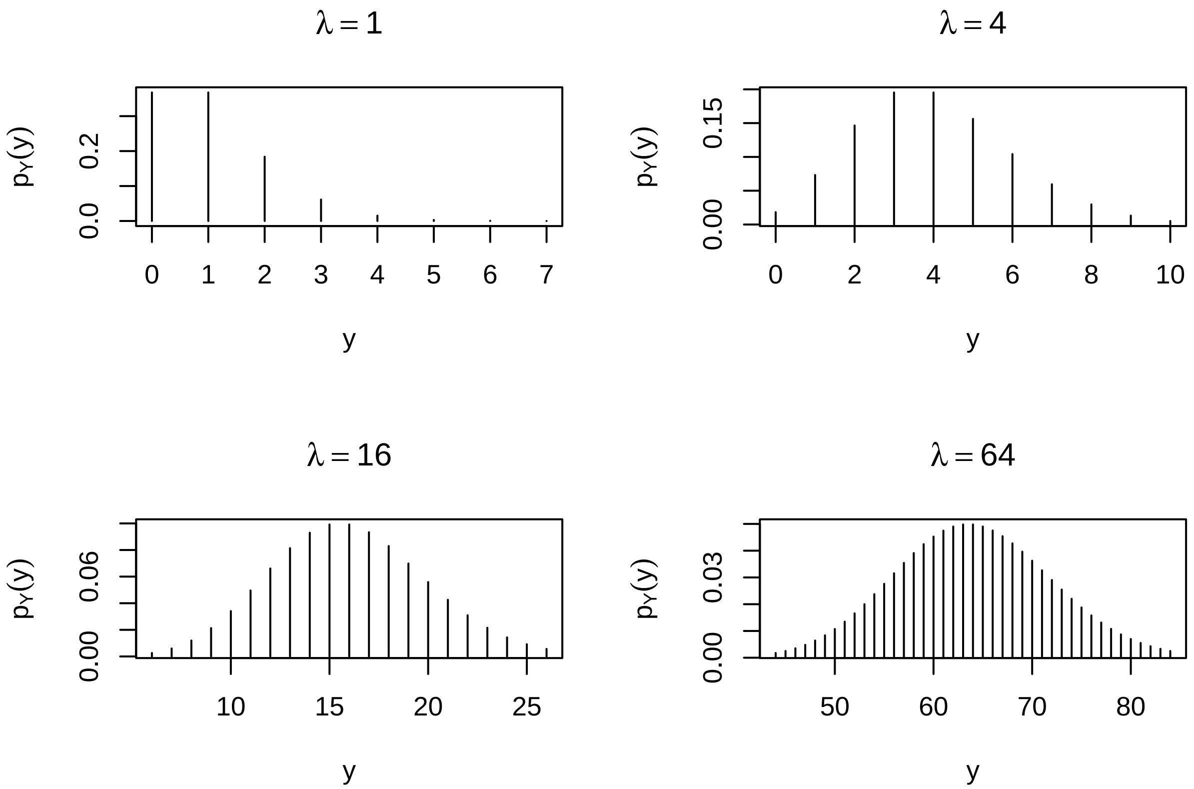
```
## [1]  1  9 25 49
```

```
# in alternative use a cycle
x <- c()
for(i in a) x <- c(x, my.power(i))
x
```

```
## [1]  1  9 25 49
```

```
# "sapply" function: the same as the
# cycle for above but more efficient

x <- sapply(a, FUN = my.power)


# plot the binomial mass functions
# for different values of the
# probability parameter

# p.vec <- seq(0, 1, length=100). #Run this code chunk in RStudio

# for (p in p.vec) {
  #plot(0:10, dbinom(0:10,10,p), ylim=c(0,0.5), type="h")
  #Sys.sleep(0.1)
# }

# function for plot of binomial(10, p)

binom.plot <- function(p) {
  plot(0:10, dbinom(0:10,10,p), ylim=c(0,0.5), type="h")
  Sys.sleep(0.1)
  }

# cycle
#for (p in p.vec) binom.plot(p)

# sapply function
#ignore <- sapply(p.vec, binom.plot)

# behaviour of binomial as n increases
# function for binomial(n, 0.5)

#binom.plot <- function(n) {
 # plot(0:n, dbinom(0:n,n,0.5), type="h")
  #Sys.sleep(0.1)
#}

#ignore <- sapply(1:100, binom.plot)

# Poisson distribution

yl <- expression(p[Y](y))

par(mfrow=c(2,2))
y <- 0:7
```

```
plot (y, dpois(y,1), xlab="y", ylab=yl, main=expression(lambda==1), type="h")
y <- 0:10
plot (y, dpois(y,4), xlab="y", ylab=yl, main=expression(lambda==4), type="h")
y <- 6:26
plot (y, dpois(y,16), xlab="y", ylab=yl,main=expression(lambda==16), type="h")
y <- 44:84
plot (y, dpois(y,64), xlab="y", ylab=yl,main=expression(lambda==64), type="h")
```




```
par(mfrow=c(1,1))

# negative binomial distribution

#plot(dnbinom(0:30, 10, 1/2), names.arg=0:30 + 10,
# xlab="Number of flips before 10 heads", type="h").
# Run this code chunk in RStudio
```