Assignment: Probability Distributions in R

Nagchou Lor

1/15/15

Continuous

Probability

```
#This code computes the probability that a normally distributed random number will be less than 1.25.
pnorm(1.25)
```

```
## [1] 0.8943502
```

```
#This code simulates the normal probability that will be less than 2.8, using the mean =2 and the standard deviation =3.
pnorm(2.8,2,3)
```

```
## [1] 0.6051371
```

```
#This code gives the p-value associated with X-squared stat 13.9 when the degree of freedom is 25.
pchisq(13.9,25)
```

```
## [1] 0.03655238
```

```
#This code returns the cumulative distribution function of the exponential distribution when the parameter is 10 and the observation P(X >
 4).
1 - pexp(4,10)
```

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

```
#Or lower.tail if false, the lower tail of the distribution is not considered.
#lower.tail if true, the lower tail of the distribution is considered.
#The output is nearly 0 or 1.
pexp(5,25,lower.tail=FALSE)
```

```
## [1] 5.166421e-55
```

```
#This outputs the probability that T, t-distribution, is less than or equal to 3.9 with 7 degrees of freedom.
pt(3.9,7) #pt(t-value, d.f)
```

```
## [1] 0.9970506
```

Quantiles

```
#This is a normally distributed random variable with mean zero and standard deviation one N(0,1), then if you give the function a probabili
ty it returns the associated Z-score.
#For this example to find the 25th percentile, let q = .25 such that P(X<=q).
qnorm(.25)
```

```
## [1] -0.6744898
```

```
#This code calculates the .75 quantile for N(2,3)
qnorm(.75,2,3)
```

```
## [1] 4.023469
```

```
#This code finds the critical value under the t distribution at 13 degrees of freedom with left-tail probability 0.975 can be found by givi
ng the command:
qt(.975,13)
```

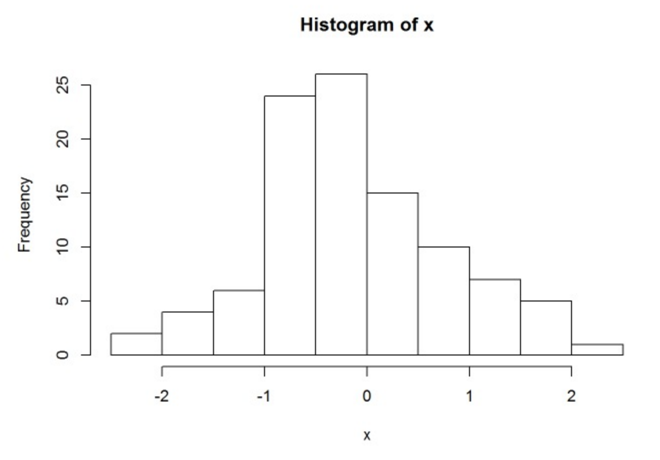
```
## [1] 2.160369
```

Random Numbers

```
#This code generates n random numbers from the normal distribution N(0,1).
#For example let n=100 with N(0,1)
rnorm(100)
```

```
##   [1] -1.67221932 -0.42323975  1.08278111 -0.02491014  1.25744279
##   [6] -0.23194406  1.80580279 -0.17718869 -0.73944584 -0.98018819
##  [11]  1.37961768  0.17236934  0.30109029 -0.03845408  0.22141663
##  [16] -1.78927940  0.92365002 -0.91059748 -1.63483276  0.59075236
##  [21]  1.32662189 -0.15241280 -0.43007180 -0.84581758  0.94754317
##  [26]  0.53745743 -0.28558601  0.48897451 -0.79346917 -0.49301202
##  [31]  2.28757924  2.14058101  1.04801828  0.73830347  0.64629552
##  [36] -0.15471602 -0.76994872  0.24090785  0.77596108  0.93102864
##  [41]  1.55724928  0.71428644  0.90939927  0.38987439  0.58290240
##  [46]  0.13529807  0.05991876  0.85754841  0.94690908 -0.43574949
##  [51] -1.34558494  1.75160958  0.12880792 -0.52923394 -0.27480624
##  [56] -0.63206333  0.08776229 -0.69750999 -0.55742969 -1.79631217
##  [61]  0.64650249 -1.00158256 -2.35680704  0.22491638  1.72565120
##  [66] -0.63675441  0.09547242  0.78841993 -0.78020753  0.20770777
##  [71] -1.26757148  0.02199952  0.29272454  1.98330460  0.67510180
##  [76] -0.86633303 -0.74887895  0.92356200  0.88363921  1.46099905
##  [81] -0.36627286  1.23213313 -0.71431721  1.31483799 -0.42909029
##  [86]  0.27947418 -0.33290579 -0.73898748 -1.28602707  0.31275965
##  [91] -0.64724552 -0.97324253  0.65044454 -1.66903293  0.70363829
##  [96] -1.23743986  0.35608724  0.28731764 -0.92678965 -1.49807633
```

```
x <- rnorm(100)
#This generates the histogram of the random numbers
hist(x)
```

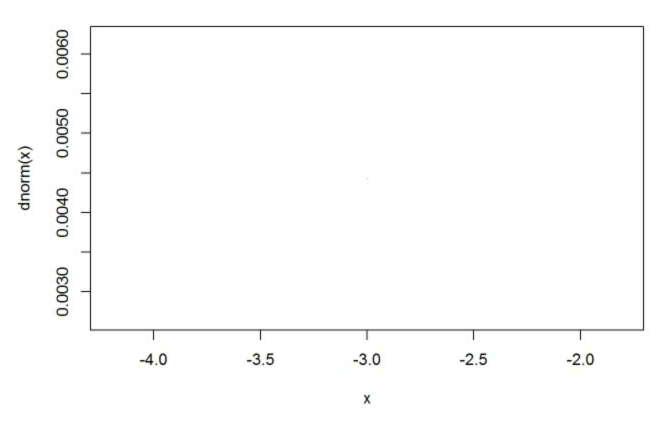



```
#To output random numbers from the chi-square distribution input rchisq(x,df). With x being the number of random numbers and df is the degr
ees of freedom.
rchisq(10,23)
```

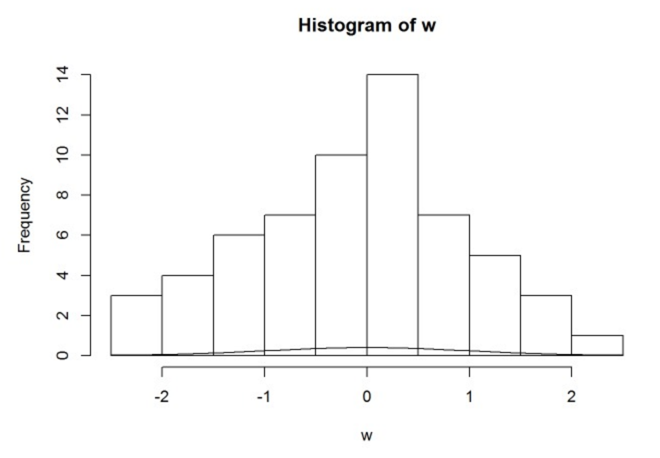
```
##  [1] 31.74429 20.92692 22.10960 16.66151 13.63256 22.14828 30.00349
##  [8] 10.99823 20.03150 14.06583
```

Plotting the Density Curve (pdf)

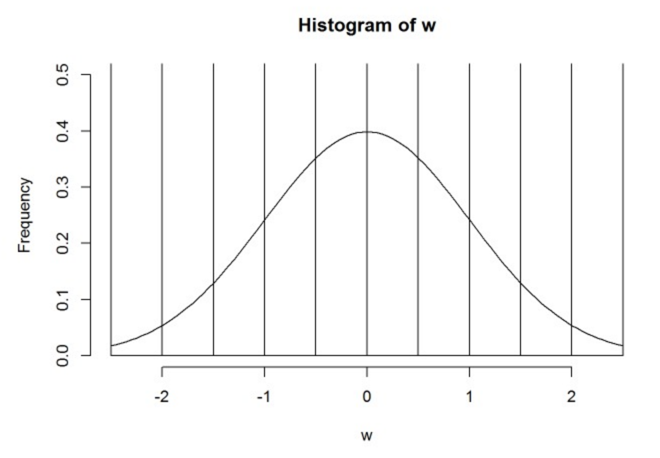
```
#The probability density function (pdf) of a continuous random variable, is a function that describes the relative likelihood for this rand
om variable to take on a given value.
#The probability of the random number falling within a particular range, for this example is between -3 to 3.
curve(dnorm(x),from = -3, to = -3)
```




```
w <- rnorm(60) #this codes the random sample from N(0,1)
hist(w,freq = TRUE) #A histogram representation of frequencies where the area equal one.
curve(dnorm(x),add = TRUE) #The curve impose normal density
```




```
hist(w,freq = TRUE, ylim = c(0,.5)) #Set the range of the y-axis.
curve(dnorm(x), add = TRUE)
```




```
#To plot the pdf curve for the chi-square with the vector x and with 14 degrees of freedom and set with parameters from 0 to 20.
?curve(dchisq(x,14),from = 0, to = 20)
```

```
## starting httpd help server ... done
```

Discrete

Example–Binomial

```
#Suppose you have a biased coin that has a probability of 0.8 of coming up heads. The probability of getting 5 heads in 16 tosses of this c
oin is
dbinom(5,16,.8) #Gives the density (x,size,probability)
```

```
## [1] 2.931315e-05
```

```
#To check the math for dbinom
choose(16,5)*.8^5*.2^11
```

```
## [1] 2.931315e-05
```

```
#This gives the probability of getting at most 5 heads in 16 tosses is
pbinom(5,16,.8)
```

```
## [1] 3.26145e-05
```

```
#pbinom(5,16,.8) is the same as computing dbinom 5 times of getting heads on a toss: dbinom(0,16,.8)+dbinom(1,16,.8)+dbinom(2,16,.8)+dbinom
(3,16,.8)+dbinom(4,16,.8)+dbinom(5,16,.8)

#This code calculates the 0.25 quantile of the 16 tosses and the probability of 0.8 success. Furthermore, it calculates the smallest number
 of successes that the probability of successes is greater than or equal to .25.
qbinom(.25,16,.8)
```

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

```
#To check qbinom use these codes:
pbinom(11,16,.8) #This is less than .25, so it cannot be the smallest number of successes.
```

```
## [1] 0.2017546
```

```
pbinom(12,16,.8) #This .40, which is greater than or equal to .25 and is the smallest number of success.
```

```
## [1] 0.4018657
```

Example–Geometric

```
#In R the geometric distribution is the number of failures before a success, not the number of trials including the success.

#So if we want to find the probability that the first head is on the fourth toss, then the number of failure will be 3. Then from the previ
ous example, the probability of tossing a head is 0.8.
#The arguments to geom are geom(# of failures,p).
dgeom(3,.8)
```

```
## [1] 0.0064
```

```
#What if we wanted to calculate the probability that the first head occurs on one of the first four tosses (that is, on the first, second,
 third or fourth toss)?
#To calculate this probability, use pgeom, the geometric distribution models the number of failures until the first head occurs on one of t
he first four tosses.
pgeom(3,.8)
```

```
## [1] 0.9984
```

Example–Poisson

```
#Poisson distribution is expressing the probability of a given number of events occurring in a fixed interval of time. Lambada is mean of o
ccurrences.
#Suppose a certain region of California experiences about 5 earthquakes a year. Assume occurrences follow a Poisson distribution. What is t
he probability of 3 earthquakes in a given year?
#We want to know the probability of 3 earthquakes occurring in one year. When we know that the average that California experiences about 5
 a year.
?dpois(3,5) #dpois(x, lambda(mean))

5^3*exp(-5)/(3*2) #The equation to check dpois()
```

```
## [1] 0.1403739
```

Random Numbers

```
# This code simulates 25 flips of a biased coin where P(Head) = 0.8 or 80%.
# The output is the total count of heads (or successes) in those 25 flips.
rbinom(1,25,.8)
```

```
## [1] 15
```

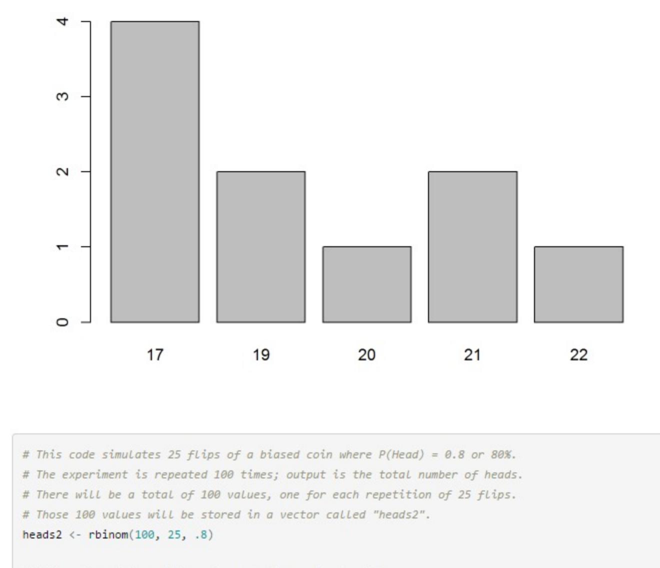
```
# This value seeds the pseudo-random number generator in R. It is optional.
# Anyone using the same seed value will generate identical "random" values.
set.seed(0)

# This code simulates 25 flips of a biased coin where P(Head) = 0.8 or 80%.
# The experiment is repeated 10 times; output is the total number of heads.
# There will be a total of ten values, one for each repetition of 25 flips.
# Those ten values will be stored in a vector called "heads".
heads <- rbinom(10, 25, .8)

# This code prints out the values in the vector heads.
heads
```

```
##  [1] 17 21 21 20 17 22 17 17 19 19
```

```
# This code creates a frequency table of the values in the vector heads.
table(heads)
```

```
## heads
## 17 19 20 21 22
##  4  2  1  2  1
```

```
# This code creates a bar chart of the values in the vector heads.
barplot(table(heads))
```



```
# This code simulates 25 flips of a biased coin where P(Head) = 0.8 or 80%.
# The experiment is repeated 100 times; output is the total number of heads.
# There will be a total of 100 values, one for each repetition of 25 flips.
# Those 100 values will be stored in a vector called "heads2".
heads2 <- rbinom(100, 25, .8)

# This code prints out the values in the vector heads2.
heads2
```

```
##   [1] 23 22 22 19 21 19 20 19 15 21 19 17 22 19 22 21 21 24 21 18 21 20 20
##  [24] 20 22 18 19 18 22 19 21 18 19 18 20 20 18 24 20 19 19 20 18 20 21 23
##  [47] 22 21 20 19 21 17 21 20 21 19 21 20 19 23 18 21 18 21 21 20 17 18 21
##  [70] 19 16 20 19 21 21 19 22 19 22 21 22 21 23 19 18 19 18 20 21 18 20 19
##  [93] 21 21 15 19 22 22 20 17
```

```
# This code creates a frequency table of the values in the vector heads2.
table(heads2)
```

```
## heads2
## 15 16 17 18 19 20 21 22 23 24
##  2  1  4 13 21 17 24 12  4  2
```

```
# This code creates a bar chart of the values in the vector heads.
barplot(table(heads2))
```