R Objects, workflows, and functions

Create a vector

```
set.seed(42) ##ensures same random numbers selected
my_unif <- runif(30)
is.vector(my_unif) ## = True
```

```
[1] TRUE
```

```
my_unif[1:10]
```

```
 [1] 0.9148060 0.9370754 0.2861395 0.8304476 0.6417455 0.5190959 0.7365883
 [8] 0.1346666 0.6569923 0.7050648
```

```
my_unif[c(1:3, 15:17)]
```

```
[1] 0.9148060 0.9370754 0.2861395 0.4622928 0.9400145 0.9782264
```

Sort the vector

```
sort(my_unif)
```

```
 [1] 0.08243756 0.11748736 0.13466660 0.13871017 0.25542882 0.28613953
 [7] 0.39020347 0.44696963 0.45774178 0.46229282 0.47499708 0.51421178
[13] 0.51909595 0.56033275 0.64174552 0.65699229 0.70506478 0.71911225
[19] 0.73658831 0.83044763 0.83600426 0.90403139 0.90573813 0.91480604
[25] 0.93467225 0.93707541 0.94001452 0.94666823 0.97822643 0.98889173
```

Create a vector with strings in it

```r
char_vec <- c('daf', 'Adf', '12h', 'h3')
sort(char_vec) #numbers, then uppercase, then lowercase
```

```
[1] "12h" "Adf" "daf" "h3"
```

Create a matrix

```r
#basic call, default fills in matrix by column (left then right)
my_mat1 <- matrix(c(1, 3, 4, -1, 5, 6),
                  nrow = 3,
                  ncol = 2)
my_mat1
```

```
     [,1] [,2]
[1,]    1   -1
[2,]    3    5
[3,]    4    6
```

```r
#changes default to fill in by row (left to right)
my_mat2 <- matrix(c(1, 3, 4, -1, 5, 6),
                  nrow = 3,
                  ncol = 2,
                  byrow = TRUE)
my_mat2
```

```
     [,1] [,2]
[1,]    1    3
[2,]    4   -1
[3,]    5    6
```

```r
#name the columns
my_mat3 <- matrix(c(runif(10),
                    rnorm(10),
                    rgamma(10, shape = 1, scale = 1)),
                  ncol = 3,
                  dimnames = list(1:10, c("Uniform", "Normal", "Gamma")))
my_mat3
```

```
       Uniform     Normal     Gamma
1  0.737595618 -0.3066386 0.8739106
2  0.811055141 -1.7813084 1.1225942
3  0.388108283 -0.1719174 1.4997885
4  0.685169729  1.2146747 0.1621219
5  0.003948339  1.8951935 0.1259051
6  0.832916080 -0.4304691 0.0991477
7  0.007334147 -0.2572694 0.9951484
8  0.207658973 -1.7631631 0.3007788
9  0.906601408  0.4600974 0.6562719
10 0.611778643 -0.6399949 0.2772845
```

Create an Array

```r
my_array <- array(1:24, dim = c(4, 2, 3))
my_array
```

```
, , 1

     [,1] [,2]
[1,]    1    5
[2,]    2    6
[3,]    3    7
[4,]    4    8

, , 2

     [,1] [,2]
[1,]    9   13
[2,]   10   14
[3,]   11   15
[4,]   12   16

, , 3

     [,1] [,2]
[1,]   17   21
[2,]   18   22
[3,]   19   23
[4,]   20   24
```

```r
#access your array
my_array[1,1,1]
```

```
[1] 1
```

Create a Data Frame

data.frame(..., row.names = NULL, check.rows = FALSE, check.names = TRUE, fix.empty.names = TRUE, stringsAsFactors = FALSE)

```r
x <- c("a", "b", "c", "d", "e", "f")
y <- c(1, 3, 4, -1, 5, 6)
z <- 10:15
my_df <- data.frame(char = x, data1 = y, data2 =  z)
my_df
```

```
  char data1 data2
1    a     1    10
2    b     3    11
3    c     4    12
4    d    -1    13
5    e     5    14
6    f     6    15
```

```r
data.frame(number = 1:5, letter = c("a", "b", "c", "d", "e"))
```

```
  number letter
1      1      a
2      2      b
3      3      c
4      4      d
5      5      e
```

```r
head(iris)
```

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
```

```
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

```r
iris[1:4, 2:4] #only 3 selected columns and rows 1-4
```

```
  Sepal.Width Petal.Length Petal.Width
1         3.5          1.4         0.2
2         3.0          1.4         0.2
3         3.2          1.3         0.2
4         3.1          1.5         0.2
```

```r
df1 <- iris[1, ]
str(df1)
```

```
'data.frame':   1 obs. of  5 variables:
 $ Sepal.Length: num 5.1
 $ Sepal.Width : num 3.5
 $ Petal.Length: num 1.4
 $ Petal.Width : num 0.2
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1
```

```r
df2 <- iris[1, , drop = FALSE]
str(df2)
```

```
'data.frame':   1 obs. of  5 variables:
 $ Sepal.Length: num 5.1
 $ Sepal.Width : num 3.5
 $ Petal.Length: num 1.4
 $ Petal.Width : num 0.2
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1
```

```r
iris$Sepal.Length #access single column
```

```
  [1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9 5.4 4.8 4.8 4.3 5.8 5.7 5.4 5.1
 [19] 5.7 5.1 5.4 5.1 4.6 5.1 4.8 5.0 5.0 5.2 5.2 4.7 4.8 5.4 5.2 5.5 4.9 5.0
 [37] 5.5 4.9 4.4 5.1 5.0 4.5 4.4 5.0 5.1 4.8 5.1 4.6 5.3 5.0 7.0 6.4 6.9 5.5
 [55] 6.5 5.7 6.3 4.9 6.6 5.2 5.0 5.9 6.0 6.1 5.6 6.7 5.6 5.8 6.2 5.6 5.9 6.1
 [73] 6.3 6.1 6.4 6.6 6.8 6.7 6.0 5.7 5.5 5.5 5.8 6.0 5.4 6.0 6.7 6.3 5.6 5.5
```

```
 [91] 5.5 6.1 5.8 5.0 5.6 5.7 5.7 6.2 5.1 5.7 6.3 5.8 7.1 6.3 6.5 7.6 4.9 7.3
[109] 6.7 7.2 6.5 6.4 6.8 5.7 5.8 6.4 6.5 7.7 7.7 6.0 6.9 5.6 7.7 6.3 6.7 7.2
[127] 6.2 6.1 6.4 7.2 7.4 7.9 6.4 6.3 6.1 7.7 6.3 6.4 6.0 6.9 6.7 6.9 5.8 6.8
[145] 6.7 6.7 6.3 6.5 6.2 5.9
```

Data frame video

```
data(trees)
trees
```

```
   Girth Height Volume
1    8.3     70   10.3
2    8.6     65   10.3
3    8.8     63   10.2
4   10.5     72   16.4
5   10.7     81   18.8
6   10.8     83   19.7
7   11.0     66   15.6
8   11.0     75   18.2
9   11.1     80   22.6
10  11.2     75   19.9
11  11.3     79   24.2
12  11.4     76   21.0
13  11.4     76   21.4
14  11.7     69   21.3
15  12.0     75   19.1
16  12.9     74   22.2
17  12.9     85   33.8
18  13.3     86   27.4
19  13.7     71   25.7
20  13.8     64   24.9
21  14.0     78   34.5
22  14.2     80   31.7
23  14.5     74   36.3
24  16.0     72   38.3
25  16.3     77   42.6
26  17.3     81   55.4
27  17.5     82   55.7
28  17.9     80   58.3
29  18.0     80   51.5
30  18.0     80   51.0
31  20.6     87   77.0
```

```
str(trees)
```

```
'data.frame':   31 obs. of  3 variables:
 $ Girth : num  8.3 8.6 8.8 10.5 10.7 10.8 11 11 11.1 11.2 ...
 $ Height: num  70 65 63 72 81 83 66 75 80 75 ...
 $ Volume: num  10.3 10.3 10.2 16.4 18.8 19.7 15.6 18.2 22.6 19.9 ...
```

subset a column

```
trees$Height
```

```
 [1] 70 65 63 72 81 83 66 75 80 75 79 76 76 69 75 74 85 86 71 64 78 80 74 72 77
[26] 81 82 80 80 80 87
```

Get attributes from the data frame

```
attributes(trees)
```

```
$names
[1] "Girth"  "Height" "Volume"

$class
[1] "data.frame"

$row.names
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
[26] 26 27 28 29 30 31
```

```
names(trees) #should be same output as colnames
```

```
[1] "Girth"  "Height" "Volume"
```

```
colnames(trees)
```

```
[1] "Girth"  "Height" "Volume"
```

```
colnames(trees)[2:3]
```

```
[1] "Height" "Volume"
```

##Lists

Investigating data frame from before

```
is.list(trees)
```

```
[1] TRUE
```

```
is.data.frame(trees)
```

```
[1] TRUE
```

Since this is a list, we can subset it as a list

```
trees[1] #list
```

```
   Girth
1    8.3
2    8.6
3    8.8
4   10.5
5   10.7
6   10.8
7   11.0
8   11.0
9   11.1
10  11.2
11  11.3
12  11.4
13  11.4
14  11.7
15  12.0
16  12.9
17  12.9
18  13.3
19  13.7
```

```
20  13.8
21  14.0
22  14.2
23  14.5
24  16.0
25  16.3
26  17.3
27  17.5
28  17.9
29  18.0
30  18.0
31  20.6
```

```
trees[[1]] #turns in a vector
```

```
 [1]  8.3  8.6  8.8 10.5 10.7 10.8 11.0 11.0 11.1 11.2 11.3 11.4 11.4 11.7 12.0
[16] 12.9 12.9 13.3 13.7 13.8 14.0 14.2 14.5 16.0 16.3 17.3 17.5 17.9 18.0 18.0
[31] 20.6
```

Look at linear model fit

```
fit <- lm(Volume ~ Height + Girth, data = trees)
```

Look at structure but restrict info:

```
str(fit, max.level = 1) #first level of structure
```

```
List of 12
 $ coefficients : Named num [1:3] -57.988 0.339 4.708
  ..- attr(*, "names")= chr [1:3] "(Intercept)" "Height" "Girth"
 $ residuals    : Named num [1:31] 5.462 5.746 5.383 0.526 -1.069 ...
  ..- attr(*, "names")= chr [1:31] "1" "2" "3" "4" ...
 $ effects      : Named num [1:31] -167.985 53.863 69.159 -0.884 -2.007 ...
  ..- attr(*, "names")= chr [1:31] "(Intercept)" "Height" "Girth" "" ...
 $ rank         : int 3
 $ fitted.values: Named num [1:31] 4.84 4.55 4.82 15.87 19.87 ...
  ..- attr(*, "names")= chr [1:31] "1" "2" "3" "4" ...
 $ assign       : int [1:3] 0 1 2
 $ qr           :List of 5
  ..- attr(*, "class")= chr "qr"
```

```
 $ df.residual  : int 28
 $ xlevels      : Named list()
 $ call         : language lm(formula = Volume ~ Height + Girth, data = trees)
 $ terms        :Classes 'terms', 'formula'  language Volume ~ Height + Girth
  .. ..- attr(*, "variables")= language list(Volume, Height, Girth)
  .. ..- attr(*, "factors")= int [1:3, 1:2] 0 1 0 0 0 1
  .. .. ..- attr(*, "dimnames")=List of 2
  .. ..- attr(*, "term.labels")= chr [1:2] "Height" "Girth"
  .. ..- attr(*, "order")= int [1:2] 1 1
  .. ..- attr(*, "intercept")= int 1
  .. ..- attr(*, "response")= int 1
  .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
  .. ..- attr(*, "predvars")= language list(Volume, Height, Girth)
  .. ..- attr(*, "dataClasses")= Named chr [1:3] "numeric" "numeric" "numeric"
  .. .. ..- attr(*, "names")= chr [1:3] "Volume" "Height" "Girth"
 $ model        :'data.frame':  31 obs. of  3 variables:
  ..- attr(*, "terms")=Classes 'terms', 'formula'  language Volume ~ Height + Girth
  .. .. ..- attr(*, "variables")= language list(Volume, Height, Girth)
  .. .. ..- attr(*, "factors")= int [1:3, 1:2] 0 1 0 0 0 1
  .. .. .. ..- attr(*, "dimnames")=List of 2
  .. .. ..- attr(*, "term.labels")= chr [1:2] "Height" "Girth"
  .. .. ..- attr(*, "order")= int [1:2] 1 1
  .. .. ..- attr(*, "intercept")= int 1
  .. .. ..- attr(*, "response")= int 1
  .. .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
  .. .. ..- attr(*, "predvars")= language list(Volume, Height, Girth)
  .. .. ..- attr(*, "dataClasses")= Named chr [1:3] "numeric" "numeric" "numeric"
  .. .. .. ..- attr(*, "names")= chr [1:3] "Volume" "Height" "Girth"
 - attr(*, "class")= chr "lm"
```

some helper functions exist

```
fit$coefficients
```

```
(Intercept)      Height       Girth
-57.9876589   0.3392512   4.7081605
```

```
coef(fit)
```

```
(Intercept)      Height       Girth
-57.9876589   0.3392512   4.7081605
```

```
fit$residuals
```

```
         1          2          3          4          5          6
 5.46234035  5.74614837  5.38301873  0.52588477 -1.06900844 -1.31832696
         7          8          9         10         11         12
-0.59268807 -1.04594918  1.18697860 -0.28758128  2.18459773 -0.46846462
        13         14         15         16         17         18
-0.06846462  0.79384587 -4.85410969 -5.65220290  2.21603352 -6.40648192
        19         20         21         22         23         24
-4.90097760 -3.79703501  0.11181561 -4.30831896  0.91474029 -3.46899800
        25         26         27         28         29         30
-2.27770232  4.45713224  3.47624891  4.87148717 -2.39932888 -2.89932888
        31
 8.48469518
```

```
#no helper for rank: rank(fit)
```

##Logical Statements

```
iris[iris$Species == "setosa" & iris$Petal.Width == 0.2, ]
```

```
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1           5.1         3.5          1.4         0.2  setosa
2           4.9         3.0          1.4         0.2  setosa
3           4.7         3.2          1.3         0.2  setosa
4           4.6         3.1          1.5         0.2  setosa
5           5.0         3.6          1.4         0.2  setosa
8           5.0         3.4          1.5         0.2  setosa
9           4.4         2.9          1.4         0.2  setosa
11          5.4         3.7          1.5         0.2  setosa
12          4.8         3.4          1.6         0.2  setosa
15          5.8         4.0          1.2         0.2  setosa
21          5.4         3.4          1.7         0.2  setosa
23          4.6         3.6          1.0         0.2  setosa
25          4.8         3.4          1.9         0.2  setosa
26          5.0         3.0          1.6         0.2  setosa
28          5.2         3.5          1.5         0.2  setosa
29          5.2         3.4          1.4         0.2  setosa
30          4.7         3.2          1.6         0.2  setosa
31          4.8         3.1          1.6         0.2  setosa
```

```
34          5.5         4.2          1.4         0.2  setosa
35          4.9         3.1          1.5         0.2  setosa
36          5.0         3.2          1.2         0.2  setosa
37          5.5         3.5          1.3         0.2  setosa
39          4.4         3.0          1.3         0.2  setosa
40          5.1         3.4          1.5         0.2  setosa
43          4.4         3.2          1.3         0.2  setosa
47          5.1         3.8          1.6         0.2  setosa
48          4.6         3.2          1.4         0.2  setosa
49          5.3         3.7          1.5         0.2  setosa
50          5.0         3.3          1.4         0.2  setosa
```

#If/then/else

Fizz buzz challenge - take in a number - if it is divisible by 3 return fizz - if it divisiabile by 5 return buzz - if it divisible by 15 return fizz buzz

```
number <- 2
if((number %% 15) == 0) {
  print("fizz buzz")
} else if ((number %% 5) == 0) {
  print("buzz")
} else if ((number %% 3) == 0) {
  print("fizz")
} else {
  print("whoops?")
}
```

```
[1] "whoops?"
```

##Loops

Wrap the fizz buzz code into a loop to check for multiple values. And add it into a data frame

```
summary_fb <- data.frame(num =  -1:30)
summary_fb
```

```
   num
1   -1
2    0
3    1
```

```
4    2
5    3
6    4
7    5
8    6
9    7
10   8
11   9
12  10
13  11
14  12
15  13
16  14
17  15
18  16
19  17
20  18
21  19
22  20
23  21
24  22
25  23
26  24
27  25
28  26
29  27
30  28
31  29
32  30
```

```
for(i in -1:30){
             if((i %% 15) == 0) {
              summary_fb[i,2] <- "fizz buzz"
             } else if ((i %% 5) == 0) {
               summary_fb[i,2] <- "buzz"
             } else if ((i %% 3) == 0) {
               summary_fb[i,2] <- "fizz"
             } else {
               summary_fb[i,2] <- "whoops?"
  }
}
summary_fb
```

```
   num        V2
1   -1   whoops?
2    0   whoops?
3    1      fizz
4    2   whoops?
5    3      buzz
6    4      fizz
7    5   whoops?
8    6   whoops?
9    7      fizz
10   8      buzz
11   9   whoops?
12  10      fizz
13  11   whoops?
14  12   whoops?
15  13 fizz buzz
16  14   whoops?
17  15   whoops?
18  16      fizz
19  17   whoops?
20  18      buzz
21  19      fizz
22  20   whoops?
23  21   whoops?
24  22      fizz
25  23      buzz
26  24   whoops?
27  25      fizz
28  26   whoops?
29  27   whoops?
30  28 fizz buzz
31  29   whoops?
32  30   whoops?
```

##vectorized functions

```
ifelse(airquality$Wind >= 15, "HighWind",
          ifelse(airquality$Wind >= 10, "Windy",
                 ifelse(airquality$Wind >= 6, "LightWind",
                        ifelse(airquality$Wind >= 0, "Calm", "Error"))))
```

```
  [1] "LightWind" "LightWind" "Windy"     "Windy"     "Windy"     "Windy"
```

```
  [7] "LightWind" "Windy"     "HighWind"  "LightWind" "LightWind" "LightWind"
 [13] "LightWind" "Windy"     "Windy"     "Windy"     "Windy"     "HighWind"
 [19] "Windy"     "LightWind" "LightWind" "HighWind"  "LightWind" "Windy"
 [25] "HighWind"  "Windy"     "LightWind" "Windy"     "Windy"     "Calm"
 [31] "LightWind" "LightWind" "LightWind" "HighWind"  "LightWind" "LightWind"
 [37] "Windy"     "LightWind" "LightWind" "Windy"     "Windy"     "Windy"
 [43] "LightWind" "LightWind" "Windy"     "Windy"     "Windy"     "HighWind"
 [49] "LightWind" "Windy"     "Windy"     "LightWind" "Calm"      "Calm"
 [55] "LightWind" "LightWind" "LightWind" "Windy"     "Windy"     "Windy"
 [61] "LightWind" "Calm"      "LightWind" "LightWind" "Windy"     "Calm"
 [67] "Windy"     "Calm"      "LightWind" "Calm"      "LightWind" "LightWind"
 [73] "Windy"     "Windy"     "Windy"     "Windy"     "LightWind" "Windy"
 [79] "LightWind" "Calm"      "Windy"     "LightWind" "LightWind" "Windy"
 [85] "LightWind" "LightWind" "LightWind" "Windy"     "LightWind" "LightWind"
 [91] "LightWind" "LightWind" "LightWind" "Windy"     "LightWind" "LightWind"
 [97] "LightWind" "Calm"      "Calm"      "Windy"     "LightWind" "LightWind"
[103] "Windy"     "Windy"     "Windy"     "LightWind" "Windy"     "Windy"
[109] "LightWind" "LightWind" "Windy"     "Windy"     "HighWind"  "Windy"
[115] "Windy"     "LightWind" "Calm"      "LightWind" "Calm"      "LightWind"
[121] "Calm"      "LightWind" "LightWind" "LightWind" "Calm"      "Calm"
[127] "Calm"      "LightWind" "HighWind"  "Windy"     "Windy"     "Windy"
[133] "LightWind" "Windy"     "HighWind"  "LightWind" "Windy"     "Windy"
[139] "LightWind" "Windy"     "Windy"     "Windy"     "LightWind" "Windy"
[145] "LightWind" "Windy"     "Windy"     "HighWind"  "LightWind" "Windy"
[151] "Windy"     "LightWind" "Windy"
```

Writing Functions

generic syntax

nameOfFunction <- function(input1, input2, ...) { #code #return something with return() #or returns last value }

```
standardize <- function(vector){
  return((vector - mean(vector))/ sd(vector))
}
```

Create a dataset and apply the function

```
set.seed(10)
data <- runif(15)
data
```

```
 [1] 0.50747820 0.30676851 0.42690767 0.69310208 0.08513597 0.22543662
 [7] 0.27453052 0.27230507 0.61582931 0.42967153 0.65165567 0.56773775
[13] 0.11350898 0.59592531 0.35804998
```

```
result <- standardize(data)
result
```

```
 [1]  0.51053294 -0.52232963  0.09591275  1.46576309 -1.66286222 -0.94086777
 [7] -0.68822797 -0.69968029  1.06811337  0.11013572  1.25247769  0.82063172
[13] -1.51685322  0.96568634 -0.25843252
```

```
#check for mean 0 and sd = 1
round(mean(result), digits = 0)
```

```
[1] 0
```

```
sd(result)
```

```
[1] 1
```

update function to automatically return mean & SD and default values

```
standardize <- function(vector, center = TRUE, scale = TRUE){
  mean <- round(mean(vector))
  stdev <- sd(vector)
  if (center) {
    vector <- vector - mean
    }
  if (scale) {
    vector <- vector / stdev
    }
  return(list(result = vector, mean = mean, sd = stdev))
  }
```

Apply it

```
result <- standardize(data)
result
```

```
$result
 [1] 2.6115093 1.5786467 2.1968891 3.5667395 0.4381141 1.1601086 1.4127484
 [8] 1.4012961 3.1690897 2.2111121 3.3534540 2.9216081 0.5841231 3.0666627
[15] 1.8425438

$mean
[1] 0

$sd
[1] 0.1943237
```

```
result[[2]] # only mean
```

```
[1] 0
```

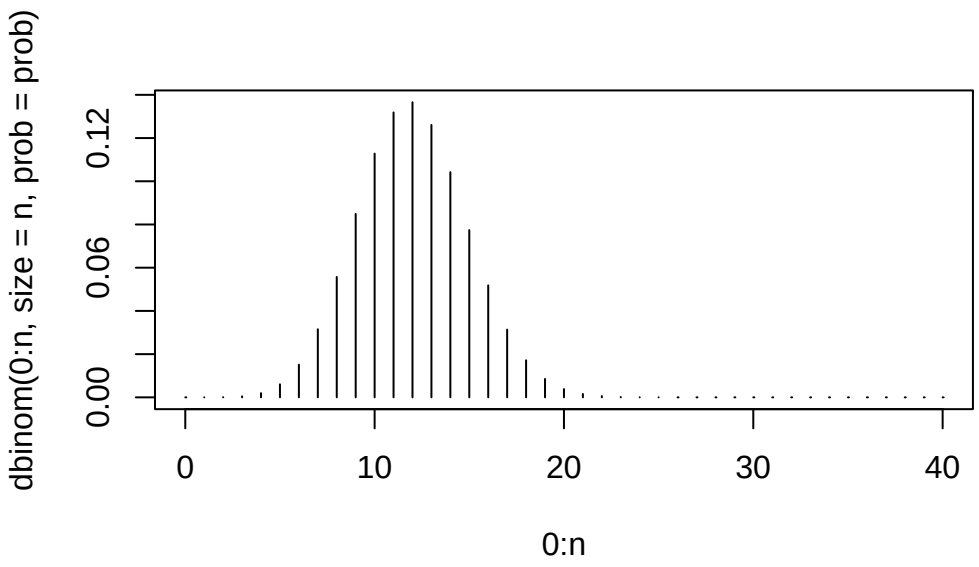
Writing R video

Normal approximation LO the binomial.

```
n <-40
prob <-0.3
#probabilities from a binomial RV
dbinom(0:n, size = n, prob = prob)
```

```
 [1] 6.366806e-07 1.091452e-05 9.121424e-05 4.951630e-04 1.962968e-03
 [6] 6.057157e-03 1.514289e-02 3.152194e-02 5.572629e-02 8.491625e-02
[11] 1.128173e-01 1.318644e-01 1.365738e-01 1.260681e-01 1.041992e-01
[16] 7.740510e-02 5.183378e-02 3.136161e-02 1.717422e-02 8.522543e-03
[21] 3.835144e-03 1.565365e-03 5.793884e-04 1.943290e-04 5.899274e-05
[26] 1.618087e-05 4.000763e-06 8.890585e-07 1.769045e-07 3.137223e-08
[31] 4.929921e-09 6.815560e-10 8.215184e-11 8.535256e-12 7.531108e-13
[36] 5.533059e-14 3.293487e-15 1.525940e-16 5.162955e-18 1.134715e-19
[41] 1.215767e-21
```

```
#plot
plot(0:n, dbinom(0:n, size = n, prob = prob),
     type = "h") #type h for histogram
```




```
#plot normal distribution
norm_x <- seq(from = 0, to = n, length = 1000)
dnorm(norm_x, mean = n*prob, sd = sqrt(n*prob*(1-prob)))
```

```
  [1] 2.607632e-05 2.760874e-05 2.922563e-05 3.093131e-05 3.273029e-05
  [6] 3.462729e-05 3.662725e-05 3.873533e-05 4.095691e-05 4.329765e-05
 [11] 4.576343e-05 4.836040e-05 5.109499e-05 5.397391e-05 5.700417e-05
 [16] 6.019305e-05 6.354820e-05 6.707756e-05 7.078943e-05 7.469244e-05
 [21] 7.879560e-05 8.310831e-05 8.764033e-05 9.240185e-05 9.740348e-05
 [26] 1.026562e-04 1.081716e-04 1.139616e-04 1.200386e-04 1.264154e-04
 [31] 1.331057e-04 1.401232e-04 1.474826e-04 1.551989e-04 1.632877e-04
 [36] 1.717653e-04 1.806486e-04 1.899551e-04 1.997028e-04 2.099108e-04
 [41] 2.205984e-04 2.317859e-04 2.434943e-04 2.557453e-04 2.685615e-04
 [46] 2.819661e-04 2.959832e-04 3.106379e-04 3.259560e-04 3.419641e-04
 [51] 3.586899e-04 3.761621e-04 3.944100e-04 4.134642e-04 4.333563e-04
 [56] 4.541186e-04 4.757849e-04 4.983898e-04 5.219690e-04 5.465595e-04
 [61] 5.721992e-04 5.989274e-04 6.267844e-04 6.558119e-04 6.860528e-04
 [66] 7.175513e-04 7.503526e-04 7.845037e-04 8.200526e-04 8.570487e-04
 [71] 8.955430e-04 9.355876e-04 9.772363e-04 1.020544e-03 1.065568e-03
 [76] 1.112366e-03 1.160998e-03 1.211524e-03 1.264008e-03 1.318514e-03
 [81] 1.375108e-03 1.433857e-03 1.494831e-03 1.558101e-03 1.623738e-03
 [86] 1.691818e-03 1.762416e-03 1.835609e-03 1.911477e-03 1.990101e-03
```

```
 [91] 2.071564e-03 2.155949e-03 2.243344e-03 2.333836e-03 2.427516e-03
 [96] 2.524473e-03 2.624802e-03 2.728598e-03 2.835956e-03 2.946977e-03
[101] 3.061759e-03 3.180404e-03 3.303017e-03 3.429702e-03 3.560567e-03
[106] 3.695719e-03 3.835270e-03 3.979330e-03 4.128014e-03 4.281435e-03
[111] 4.439712e-03 4.602961e-03 4.771302e-03 4.944855e-03 5.123744e-03
[116] 5.308091e-03 5.498021e-03 5.693661e-03 5.895136e-03 6.102577e-03
[121] 6.316111e-03 6.535869e-03 6.761983e-03 6.994584e-03 7.233806e-03
[126] 7.479781e-03 7.732645e-03 7.992531e-03 8.259576e-03 8.533914e-03
[131] 8.815681e-03 9.105013e-03 9.402047e-03 9.706918e-03 1.001976e-02
[136] 1.034072e-02 1.066991e-02 1.100749e-02 1.135358e-02 1.170832e-02
[141] 1.207183e-02 1.244426e-02 1.282573e-02 1.321636e-02 1.361630e-02
[146] 1.402567e-02 1.444458e-02 1.487316e-02 1.531154e-02 1.575984e-02
[151] 1.621816e-02 1.668663e-02 1.716535e-02 1.765443e-02 1.815399e-02
[156] 1.866412e-02 1.918492e-02 1.971649e-02 2.025892e-02 2.081231e-02
[161] 2.137673e-02 2.195226e-02 2.253899e-02 2.313698e-02 2.374631e-02
[166] 2.436703e-02 2.499921e-02 2.564289e-02 2.629813e-02 2.696496e-02
[171] 2.764343e-02 2.833356e-02 2.903537e-02 2.974889e-02 3.047413e-02
[176] 3.121109e-02 3.195978e-02 3.272017e-02 3.349227e-02 3.427604e-02
[181] 3.507146e-02 3.587849e-02 3.669708e-02 3.752719e-02 3.836875e-02
[186] 3.922170e-02 4.008596e-02 4.096145e-02 4.184806e-02 4.274571e-02
[191] 4.365428e-02 4.457366e-02 4.550371e-02 4.644430e-02 4.739529e-02
[196] 4.835653e-02 4.932784e-02 5.030905e-02 5.130000e-02 5.230048e-02
[201] 5.331029e-02 5.432924e-02 5.535709e-02 5.639363e-02 5.743861e-02
[206] 5.849178e-02 5.955291e-02 6.062171e-02 6.169791e-02 6.278124e-02
[211] 6.387140e-02 6.496809e-02 6.607100e-02 6.717980e-02 6.829418e-02
[216] 6.941380e-02 7.053831e-02 7.166735e-02 7.280057e-02 7.393759e-02
[221] 7.507805e-02 7.622154e-02 7.736769e-02 7.851608e-02 7.966631e-02
[226] 8.081796e-02 8.197062e-02 8.312385e-02 8.427722e-02 8.543029e-02
[231] 8.658260e-02 8.773372e-02 8.888317e-02 9.003049e-02 9.117522e-02
[236] 9.231689e-02 9.345501e-02 9.458911e-02 9.571870e-02 9.684330e-02
[241] 9.796241e-02 9.907554e-02 1.001822e-01 1.012819e-01 1.023741e-01
[246] 1.034584e-01 1.045341e-01 1.056009e-01 1.066583e-01 1.077056e-01
[251] 1.087425e-01 1.097685e-01 1.107829e-01 1.117854e-01 1.127755e-01
[256] 1.137525e-01 1.147162e-01 1.156660e-01 1.166013e-01 1.175218e-01
[261] 1.184270e-01 1.193163e-01 1.201894e-01 1.210458e-01 1.218850e-01
[266] 1.227066e-01 1.235101e-01 1.242952e-01 1.250615e-01 1.258084e-01
[271] 1.265356e-01 1.272428e-01 1.279295e-01 1.285953e-01 1.292400e-01
[276] 1.298631e-01 1.304643e-01 1.310432e-01 1.315997e-01 1.321332e-01
[281] 1.326436e-01 1.331306e-01 1.335938e-01 1.340331e-01 1.344482e-01
[286] 1.348388e-01 1.352047e-01 1.355457e-01 1.358617e-01 1.361525e-01
[291] 1.364178e-01 1.366575e-01 1.368716e-01 1.370598e-01 1.372220e-01
[296] 1.373583e-01 1.374685e-01 1.375524e-01 1.376102e-01 1.376417e-01
[301] 1.376470e-01 1.376260e-01 1.375787e-01 1.375052e-01 1.374055e-01
```

```
[306] 1.372797e-01 1.371278e-01 1.369499e-01 1.367462e-01 1.365167e-01
[311] 1.362616e-01 1.359811e-01 1.356752e-01 1.353441e-01 1.349881e-01
[316] 1.346073e-01 1.342020e-01 1.337724e-01 1.333187e-01 1.328412e-01
[321] 1.323402e-01 1.318158e-01 1.312685e-01 1.306985e-01 1.301062e-01
[326] 1.294918e-01 1.288558e-01 1.281984e-01 1.275200e-01 1.268209e-01
[331] 1.261017e-01 1.253626e-01 1.246040e-01 1.238264e-01 1.230302e-01
[336] 1.222158e-01 1.213835e-01 1.205340e-01 1.196675e-01 1.187846e-01
[341] 1.178857e-01 1.169713e-01 1.160419e-01 1.150978e-01 1.141397e-01
[346] 1.131679e-01 1.121829e-01 1.111854e-01 1.101756e-01 1.091542e-01
[351] 1.081217e-01 1.070784e-01 1.060250e-01 1.049620e-01 1.038897e-01
[356] 1.028088e-01 1.017197e-01 1.006229e-01 9.951901e-02 9.840841e-02
[361] 9.729163e-02 9.616917e-02 9.504152e-02 9.390916e-02 9.277259e-02
[366] 9.163228e-02 9.048872e-02 8.934238e-02 8.819372e-02 8.704322e-02
[371] 8.589133e-02 8.473851e-02 8.358521e-02 8.243187e-02 8.127893e-02
[376] 8.012682e-02 7.897597e-02 7.782680e-02 7.667970e-02 7.553510e-02
[381] 7.439339e-02 7.325494e-02 7.212016e-02 7.098940e-02 6.986303e-02
[386] 6.874142e-02 6.762491e-02 6.651383e-02 6.540852e-02 6.430931e-02
[391] 6.321650e-02 6.213041e-02 6.105132e-02 5.997952e-02 5.891529e-02
[396] 5.785891e-02 5.681062e-02 5.577068e-02 5.473932e-02 5.371679e-02
[401] 5.270330e-02 5.169906e-02 5.070427e-02 4.971914e-02 4.874385e-02
[406] 4.777857e-02 4.682346e-02 4.587869e-02 4.494441e-02 4.402075e-02
[411] 4.310784e-02 4.220581e-02 4.131476e-02 4.043481e-02 3.956605e-02
[416] 3.870857e-02 3.786245e-02 3.702775e-02 3.620454e-02 3.539288e-02
[421] 3.459281e-02 3.380438e-02 3.302761e-02 3.226253e-02 3.150916e-02
[426] 3.076751e-02 3.003758e-02 2.931938e-02 2.861288e-02 2.791808e-02
[431] 2.723495e-02 2.656347e-02 2.590360e-02 2.525530e-02 2.461853e-02
[436] 2.399323e-02 2.337935e-02 2.277683e-02 2.218560e-02 2.160560e-02
[441] 2.103675e-02 2.047896e-02 1.993216e-02 1.939625e-02 1.887115e-02
[446] 1.835677e-02 1.785299e-02 1.735973e-02 1.687688e-02 1.640432e-02
[451] 1.594196e-02 1.548967e-02 1.504733e-02 1.461485e-02 1.419208e-02
[456] 1.377891e-02 1.337522e-02 1.298087e-02 1.259575e-02 1.221972e-02
[461] 1.185266e-02 1.149443e-02 1.114490e-02 1.080393e-02 1.047140e-02
[466] 1.014716e-02 9.831092e-03 9.523048e-03 9.222895e-03 8.930499e-03
[471] 8.645722e-03 8.368428e-03 8.098483e-03 7.835749e-03 7.580093e-03
[476] 7.331378e-03 7.089471e-03 6.854238e-03 6.625545e-03 6.403260e-03
[481] 6.187252e-03 5.977389e-03 5.773543e-03 5.575585e-03 5.383386e-03
[486] 5.196821e-03 5.015764e-03 4.840091e-03 4.669679e-03 4.504408e-03
[491] 4.344157e-03 4.188807e-03 4.038242e-03 3.892346e-03 3.751005e-03
[496] 3.614107e-03 3.481541e-03 3.353197e-03 3.228967e-03 3.108747e-03
[501] 2.992432e-03 2.879919e-03 2.771108e-03 2.665899e-03 2.564195e-03
[506] 2.465900e-03 2.370921e-03 2.279164e-03 2.190541e-03 2.104962e-03
[511] 2.022341e-03 1.942591e-03 1.865631e-03 1.791377e-03 1.719751e-03
[516] 1.650673e-03 1.584068e-03 1.519860e-03 1.457976e-03 1.398345e-03
```

```
[521] 1.340897e-03 1.285564e-03 1.232279e-03 1.180977e-03 1.131595e-03
[526] 1.084071e-03 1.038345e-03 9.943573e-04 9.520516e-04 9.113718e-04
[531] 8.722637e-04 8.346745e-04 7.985528e-04 7.638484e-04 7.305129e-04
[536] 6.984988e-04 6.677603e-04 6.382526e-04 6.099324e-04 5.827576e-04
[541] 5.566873e-04 5.316818e-04 5.077025e-04 4.847123e-04 4.626748e-04
[546] 4.415549e-04 4.213187e-04 4.019332e-04 3.833664e-04 3.655876e-04
[551] 3.485667e-04 3.322749e-04 3.166840e-04 3.017672e-04 2.874980e-04
[556] 2.738514e-04 2.608027e-04 2.483283e-04 2.364055e-04 2.250122e-04
[561] 2.141271e-04 2.037297e-04 1.938002e-04 1.843194e-04 1.752690e-04
[566] 1.666312e-04 1.583888e-04 1.505254e-04 1.430251e-04 1.358726e-04
[571] 1.290532e-04 1.225526e-04 1.163572e-04 1.104540e-04 1.048302e-04
[576] 9.947382e-05 9.437308e-05 8.951682e-05 8.489424e-05 8.049500e-05
[581] 7.630917e-05 7.232720e-05 6.853993e-05 6.493859e-05 6.151472e-05
[586] 5.826026e-05 5.516745e-05 5.222886e-05 4.943736e-05 4.678612e-05
[591] 4.426862e-05 4.187859e-05 3.961003e-05 3.745721e-05 3.541464e-05
[596] 3.347706e-05 3.163945e-05 2.989700e-05 2.824512e-05 2.667942e-05
[601] 2.519570e-05 2.378995e-05 2.245835e-05 2.119723e-05 2.000312e-05
[606] 1.887267e-05 1.780270e-05 1.679020e-05 1.583225e-05 1.492611e-05
[611] 1.406915e-05 1.325886e-05 1.249285e-05 1.176885e-05 1.108469e-05
[616] 1.043831e-05 9.827753e-06 9.251138e-06 8.706692e-06 8.192725e-06
[621] 7.707626e-06 7.249867e-06 6.817992e-06 6.410621e-06 6.026440e-06
[626] 5.664201e-06 5.322720e-06 5.000871e-06 4.697587e-06 4.411854e-06
[631] 4.142710e-06 3.889242e-06 3.650586e-06 3.425921e-06 3.214468e-06
[636] 3.015492e-06 2.828292e-06 2.652207e-06 2.486610e-06 2.330907e-06
[641] 2.184538e-06 2.046968e-06 1.917696e-06 1.796246e-06 1.682165e-06
[646] 1.575030e-06 1.474436e-06 1.380003e-06 1.291373e-06 1.208203e-06
[651] 1.130175e-06 1.056984e-06 9.883443e-07 9.239856e-07 8.636530e-07
[656] 8.071058e-07 7.541170e-07 7.044727e-07 6.579709e-07 6.144213e-07
[661] 5.736447e-07 5.354721e-07 4.997443e-07 4.663113e-07 4.350319e-07
[666] 4.057732e-07 3.784101e-07 3.528249e-07 3.289068e-07 3.065516e-07
[671] 2.856614e-07 2.661439e-07 2.479126e-07 2.308861e-07 2.149879e-07
[676] 2.001462e-07 1.862936e-07 1.733667e-07 1.613059e-07 1.500556e-07
[681] 1.395633e-07 1.297798e-07 1.206592e-07 1.121581e-07 1.042361e-07
[686] 9.685513e-08 8.997965e-08 8.357628e-08 7.761379e-08 7.206292e-08
[691] 6.689627e-08 6.208821e-08 5.761472e-08 5.345334e-08 4.958307e-08
[696] 4.598424e-08 4.263848e-08 3.952862e-08 3.663858e-08 3.395335e-08
[701] 3.145893e-08 2.914219e-08 2.699092e-08 2.499368e-08 2.313981e-08
[706] 2.141936e-08 1.982305e-08 1.834220e-08 1.696874e-08 1.569512e-08
[711] 1.451433e-08 1.341981e-08 1.240546e-08 1.146559e-08 1.059491e-08
[716] 9.788479e-09 9.041702e-09 8.350304e-09 7.710304e-09 7.117997e-09
[721] 6.569938e-09 6.062919e-09 5.593961e-09 5.160291e-09 4.759333e-09
[726] 4.388691e-09 4.046142e-09 3.729618e-09 3.437199e-09 3.167102e-09
[731] 2.917673e-09 2.687375e-09 2.474783e-09 2.278573e-09 2.097519e-09
```

```
[736] 1.930483e-09 1.776410e-09 1.634322e-09 1.503312e-09 1.382540e-09
[741] 1.271227e-09 1.168654e-09 1.074152e-09 9.871039e-10 9.069367e-10
[746] 8.331212e-10 7.651675e-10 7.026223e-10 6.450665e-10 5.921124e-10
[751] 5.434016e-10 4.986029e-10 4.574102e-10 4.195405e-10 3.847327e-10
[756] 3.527455e-10 3.233560e-10 2.963586e-10 2.715633e-10 2.487952e-10
[761] 2.278924e-10 2.087059e-10 1.910983e-10 1.749428e-10 1.601225e-10
[766] 1.465298e-10 1.340653e-10 1.226377e-10 1.121628e-10 1.025630e-10
[771] 9.376696e-11 8.570891e-11 7.832839e-11 7.156975e-11 6.538182e-11
[776] 5.971749e-11 5.453348e-11 4.978999e-11 4.545042e-11 4.148116e-11
[781] 3.785132e-11 3.453252e-11 3.149870e-11 2.872593e-11 2.619224e-11
[786] 2.387747e-11 2.176312e-11 1.983220e-11 1.806916e-11 1.645971e-11
[791] 1.499075e-11 1.365028e-11 1.242731e-11 1.131175e-11 1.029436e-11
[796] 9.366692e-12 8.520991e-12 7.750169e-12 7.047730e-12 6.407735e-12
[801] 5.824744e-12 5.293785e-12 4.810308e-12 4.370152e-12 3.969514e-12
[806] 3.604917e-12 3.273183e-12 2.971409e-12 2.696943e-12 2.447361e-12
[811] 2.220453e-12 2.014198e-12 1.826753e-12 1.656436e-12 1.501711e-12
[816] 1.361180e-12 1.233564e-12 1.117699e-12 1.012524e-12 9.170705e-13
[821] 8.304573e-13 7.518808e-13 6.806092e-13 6.159759e-13 5.573741e-13
[826] 5.042512e-13 4.561043e-13 4.124758e-13 3.729495e-13 3.371464e-13
[831] 3.047223e-13 2.753640e-13 2.487866e-13 2.247315e-13 2.029636e-13
[836] 1.832692e-13 1.654542e-13 1.493425e-13 1.347739e-13 1.216034e-13
[841] 1.096989e-13 9.894101e-14 8.922105e-14 8.044064e-14 7.251048e-14
[846] 6.534963e-14 5.888472e-14 5.304925e-14 4.778295e-14 4.303123e-14
[851] 3.874464e-14 3.487841e-14 3.139199e-14 2.824868e-14 2.541526e-14
[856] 2.286167e-14 2.056073e-14 1.848784e-14 1.662077e-14 1.493940e-14
[861] 1.342555e-14 1.206280e-14 1.083631e-14 9.732668e-15 8.739757e-15
[866] 7.846645e-15 7.043454e-15 6.321273e-15 5.672055e-15 5.088544e-15
[871] 4.564189e-15 4.093087e-15 3.669909e-15 3.289855e-15 2.948597e-15
[876] 2.642233e-15 2.367249e-15 2.120479e-15 1.899070e-15 1.700455e-15
[881] 1.522322e-15 1.362589e-15 1.219383e-15 1.091020e-15 9.759838e-16
[886] 8.729100e-16 7.805728e-16 6.978699e-16 6.238104e-16 5.575039e-16
[891] 4.981502e-16 4.450305e-16 3.974993e-16 3.549769e-16 3.169429e-16
[896] 2.829299e-16 2.525189e-16 2.253337e-16 2.010367e-16 1.793254e-16
[901] 1.599283e-16 1.426021e-16 1.271287e-16 1.133127e-16 1.009789e-16
[906] 8.997037e-17 8.014671e-17 7.138205e-17 6.356373e-17 5.659094e-17
[911] 5.037343e-17 4.483047e-17 3.988982e-17 3.548690e-17 3.156393e-17
[916] 2.806928e-17 2.495678e-17 2.218518e-17 1.971762e-17 1.752117e-17
[921] 1.556642e-17 1.382712e-17 1.227981e-17 1.090357e-17 9.679723e-18
[926] 8.591604e-18 7.624347e-18 6.764694e-18 6.000823e-18 5.322192e-18
[931] 4.719406e-18 4.184093e-18 3.708792e-18 3.286855e-18 2.912366e-18
[936] 2.580051e-18 2.285219e-18 2.023692e-18 1.791753e-18 1.586095e-18
[941] 1.403774e-18 1.242173e-18 1.098966e-18 9.720836e-19 8.596863e-19
[946] 7.601399e-19 6.719921e-19 5.939528e-19 5.248761e-19 4.637445e-19
```

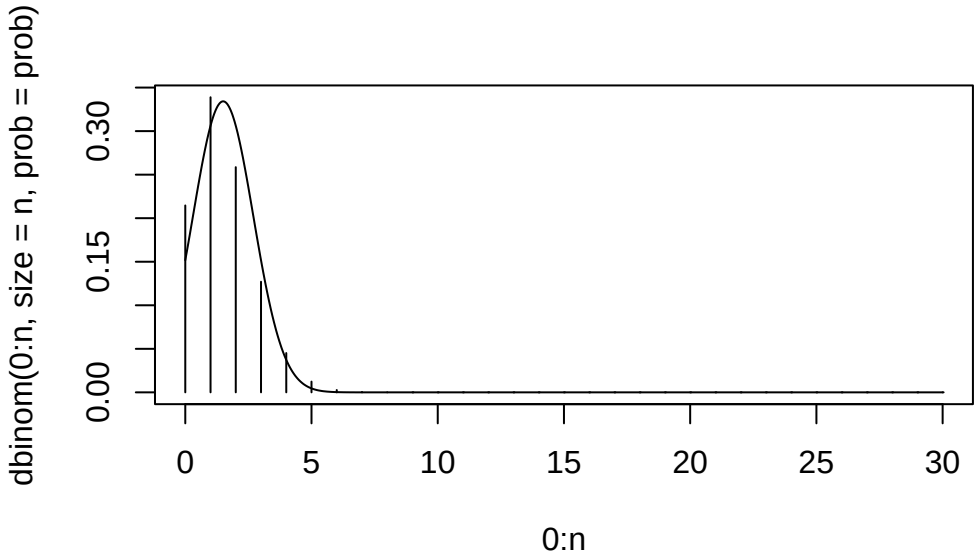
```
 [951] 4.096546e-19 3.618046e-19 3.194827e-19 2.820576e-19 2.489690e-19
 [956] 2.197202e-19 1.938705e-19 1.710293e-19 1.508504e-19 1.330269e-19
 [961] 1.172870e-19 1.033897e-19 9.112165e-20 8.029400e-20 7.073946e-20
 [966] 6.230996e-20 5.487447e-20 4.831704e-20 4.253510e-20 3.743791e-20
 [971] 3.294526e-20 2.898621e-20 2.549805e-20 2.242537e-20 1.971920e-20
 [976] 1.733629e-20 1.523843e-20 1.339187e-20 1.176683e-20 1.033701e-20
 [981] 9.079193e-21 7.972908e-21 7.000087e-21 6.144792e-21 5.392971e-21
 [986] 4.732232e-21 4.151654e-21 3.641609e-21 3.193616e-21 2.800200e-21
 [991] 2.454781e-21 2.151559e-21 1.885433e-21 1.651908e-21 1.447031e-21
 [996] 1.267322e-21 1.109720e-21 9.715309e-22 8.503879e-22 7.442085e-22
```

lets write a function to make this plot for any n and p we give it.

```r
plot_norm_approx <- function(n, prob){
  plot(0:n, dbinom(0:n, size = n, prob = prob),
     type = "h") #type h for histogram
  #plot normal distribution
  norm_x <- seq(from = 0, to = n, length = 1000)
  lines(norm_x,
        dnorm(norm_x,
              mean = n*prob,
              sd = sqrt(n*prob*(1-prob)))
        )
}
```

Test

```r
plot_norm_approx(30, 0.05)
```

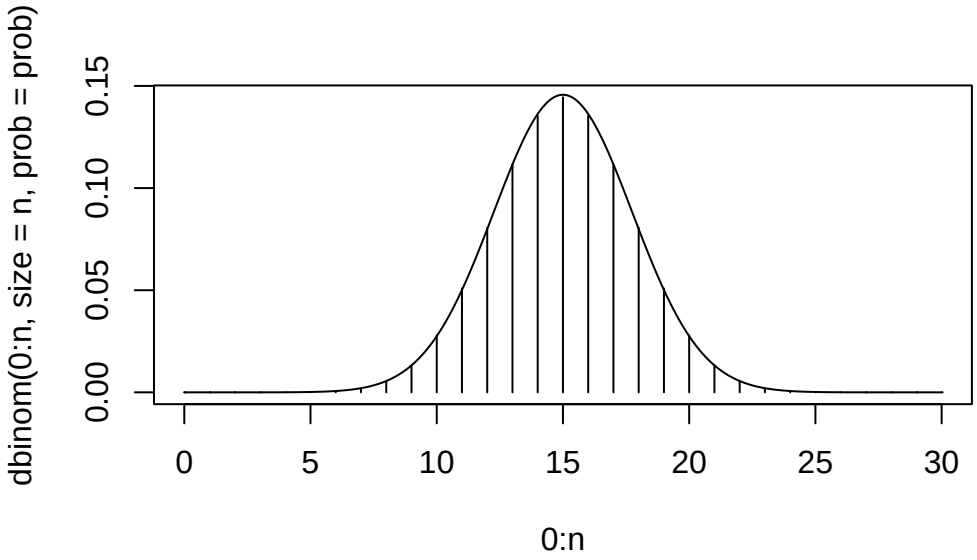


Add some default values

```
plot_norm_approx <- function(n = 30, prob = 0.5){
  plot(0:n, dbinom(0:n, size = n, prob = prob),
     type = "h") #type h for histogram
  #plot normal distribution
  norm_x <- seq(from = 0, to = n, length = 1000)
  lines(norm_x,
        dnorm(norm_x,
              mean = n*prob,
              sd = sqrt(n*prob*(1-prob)))
        )
}
```

Test

```
plot_norm_approx()
```


dbinom(0:n, size = n, prob = prob)
0.00
0.05
0.10
0.15
0
5
10
15
20
25
30
0:n