Distributions in R

Miles D. Williams
Stephen Mullins

8/14/2019

R provides us with a wide array of tools for describing and plotting various distributions, as well as generating random values from these distributions. Here, we'll explore some of these tools, namely `dnorm`, `pnorm`, `rnorm`, and `rbinom`. (Note: If you wish to access the help pages for each of these functions, simply enter, for example, `?dnorm` in your R console.)

Plotting Theoretical Distributions with `dnorm` and `pnorm`

Let's begin by plotting your first normal distribution. You can do this very simply in R. First, create a vector of values ranging from -4 to 4 in units of 0.01 (the smaller the units, the more fine grained our plot will be).

```
x = seq(
  from = -4,
  to = 4,
  by = 0.01
)
head(x) # See the first 6 values
```

```
## [1] -4.00 -3.99 -3.98 -3.97 -3.96 -3.95
```

```
tail(x) # See the last 6 values
```

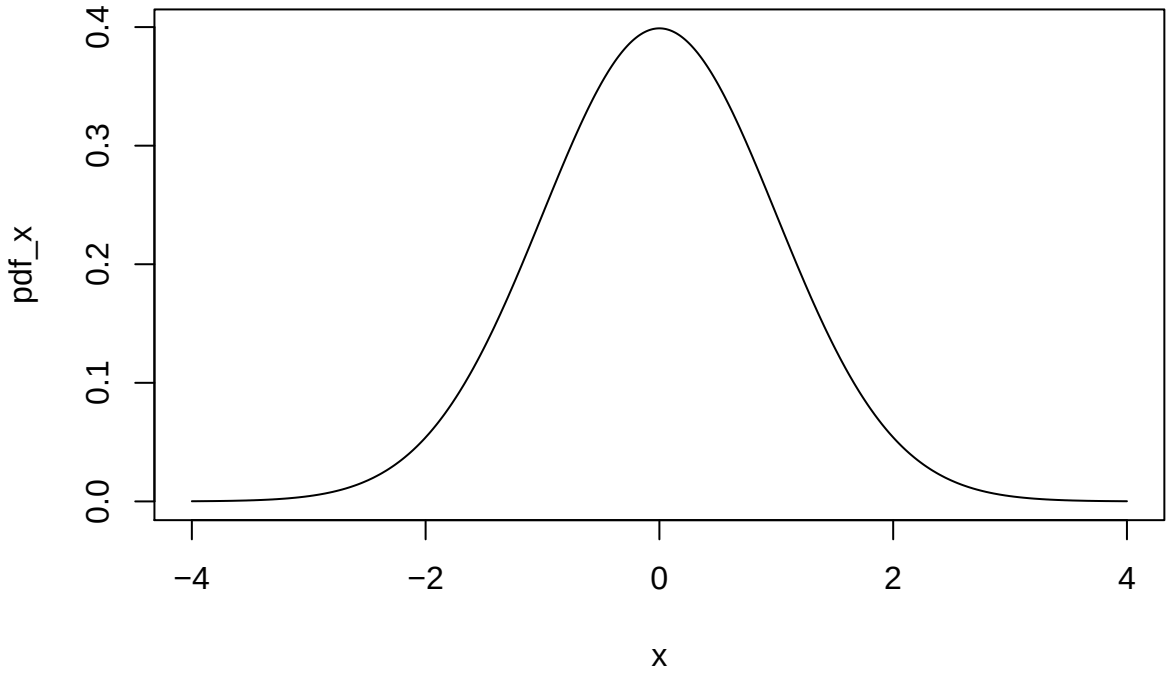
```
## [1] 3.95 3.96 3.97 3.98 3.99 4.00
```

Next, we can input the vector `x` to the `dnorm` function. `dnorm` will return the theoretical probability density over the values of `x`.

```
pdf_x = dnorm(x)
```

We can see this by plotting the object `pdf_x` over `x`:

```
plot(
  x = x,
  y = pdf_x,
  type = "l" # specify type as 'l' for line
)
```

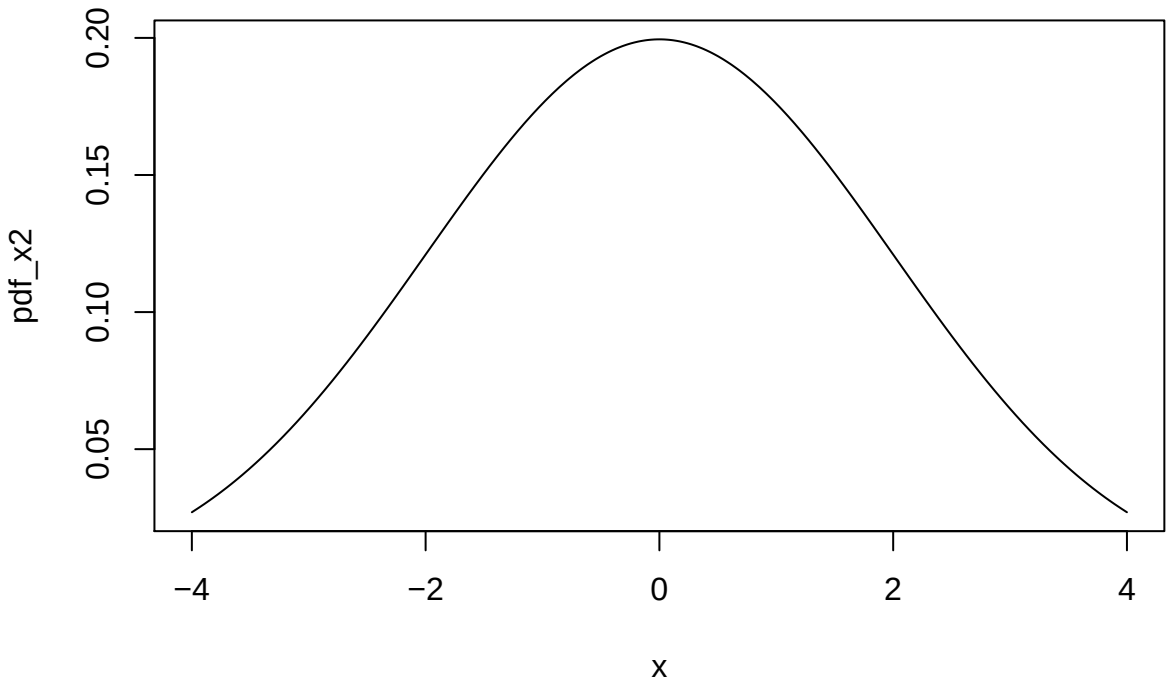


`dnorm` by default assumes a standard deviation of 1, but we can tell it that our theoretical distribution has a different standard deviation:

```
pdf_x2 = dnorm(x, sd = 2)
```

Notice that the theoretical distribution over the range -4 to 4 is now much wider:

```
plot(
  x = x,
  y = pdf_x2,
  type = "l"
)
```

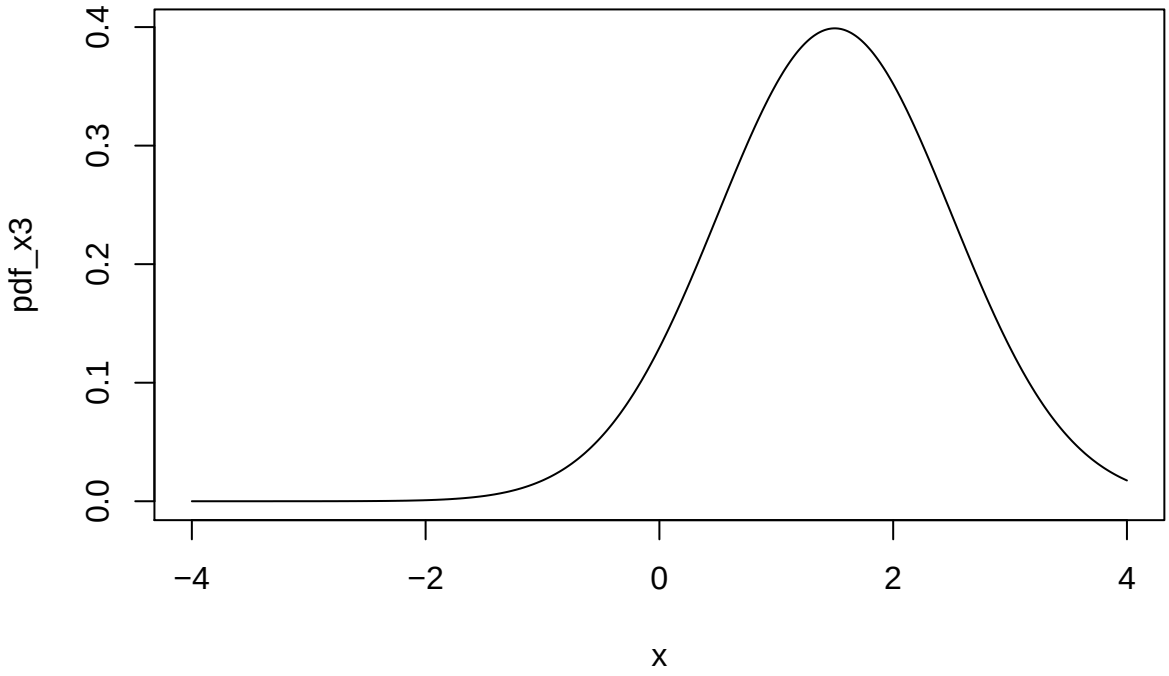


We can also specify a different mean:

```
pdf_x3 = dnorm(x, mean = 1.5)
```

Notice how our bell curve has shifted to the right:

```
plot(x, pdf_x3, type = "l")
```

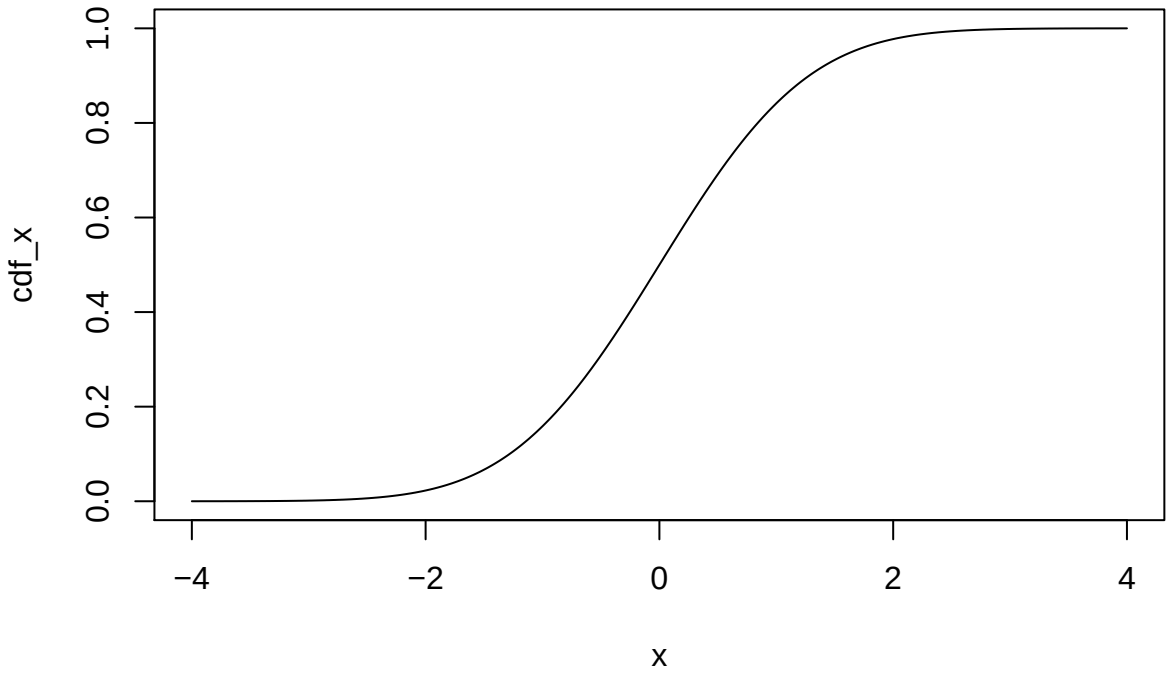


We can also plot the theoretical cumulative distribution for a standard normal variable. We do this with `pnorm`:

```
cdf_x = pnorm(x)
```

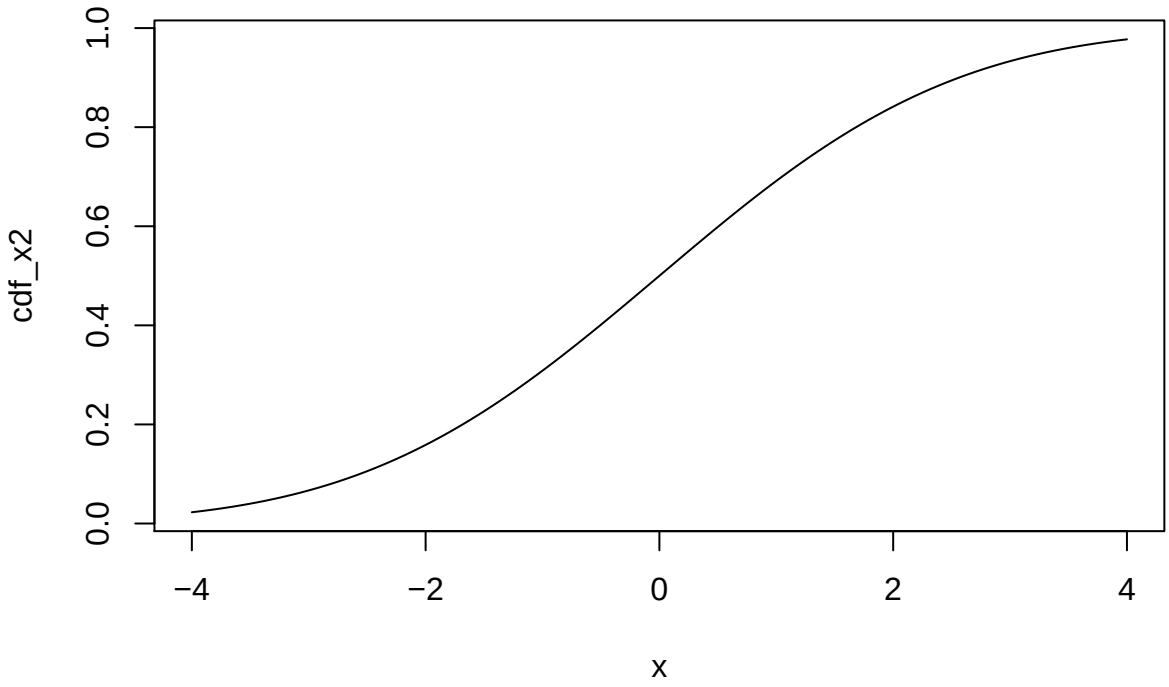
We can plot this just as we plotted the output from `dnorm`.

```
plot(
  x = x,
  y = cdf_x,
  type = "l"
)
```



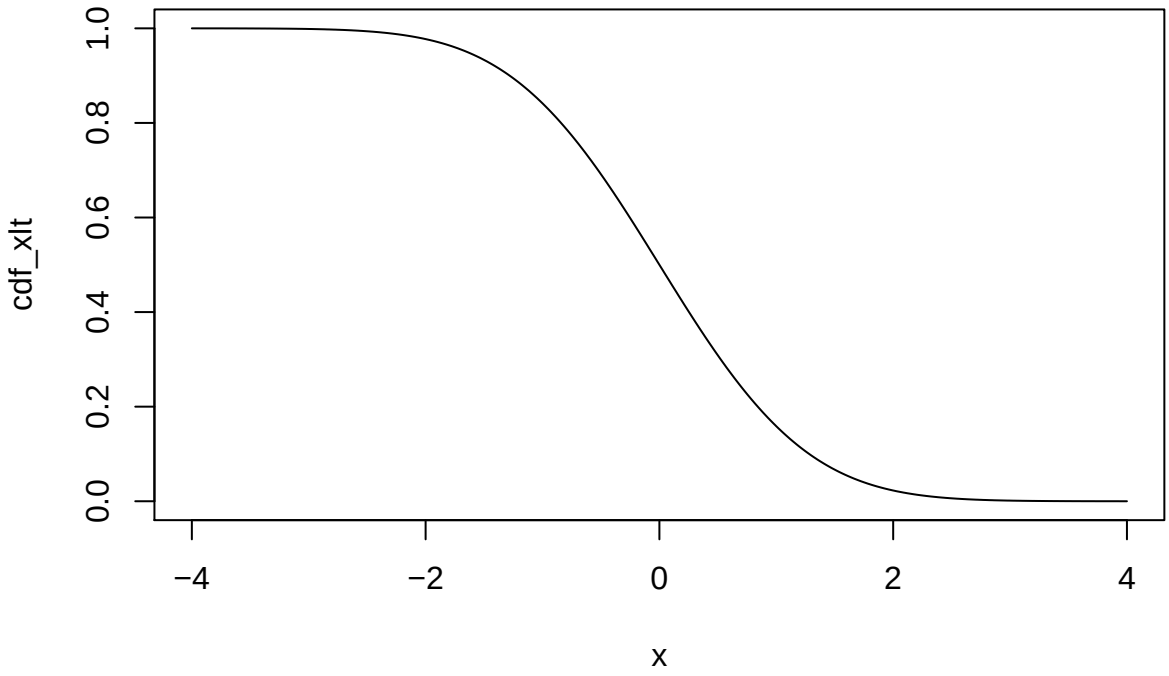
Also just like `dnorm`, we can adjust the default settings of `pnorm`. For instance, we can change the standard deviation:

```
cdf_x2 = pnorm(x, sd = 2)
plot(x, cdf_x2, type = "l")
```



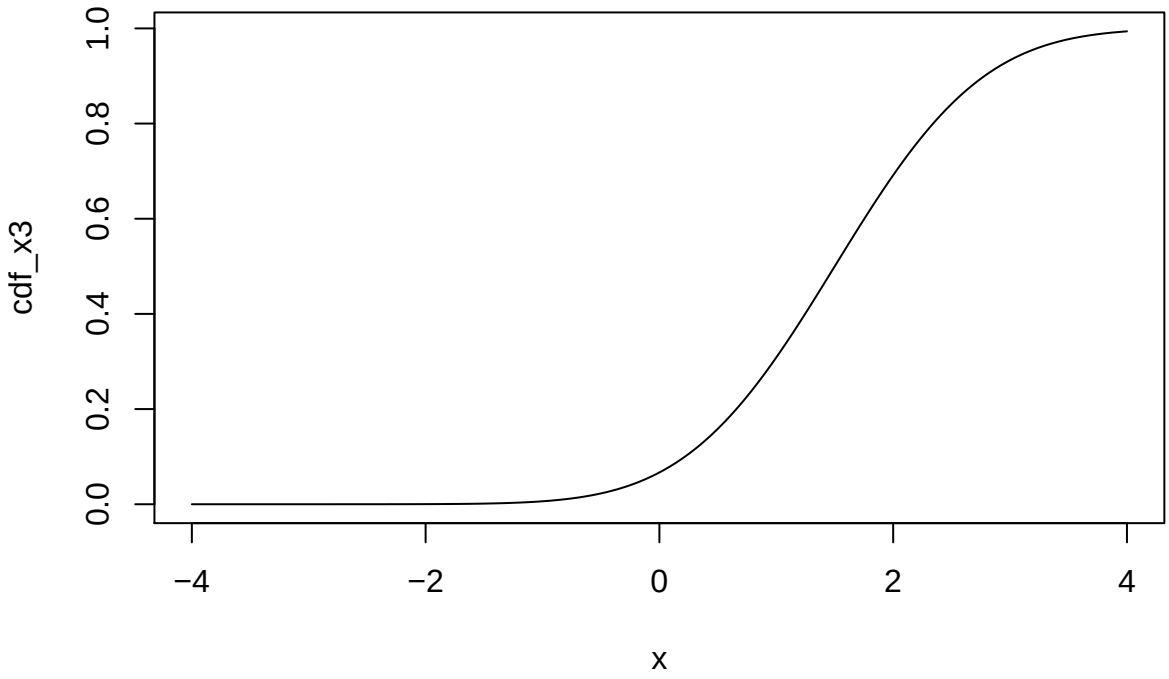
Or we can set `lower.tail = FALSE` so that rather than plot $P(X \leq x)$, we plot $P(X > x)$.

```
cdf_xlt = pnorm(x, lower.tail = F)
plot(x, cdf_xlt, type = "l")
```



Or we can shift the mean:

```
cdf_x3 = pnorm(x, mean = 1.5)
plot(x, cdf_x3, type = "l")
```



Empirical Distributions with `rnorm` and `rbinom`

Though it's useful to know how to plot theoretical distributions, in most cases you'll probably be more interested in plotting the empirical distribution of one or more variables. To show you how to do this we introduce two other useful functions, `rnorm` and `rbinom`. The first randomly generates values from a standard normal distribution. By default it draws from a distribution with mean of 0 and standard deviation of 1. The second randomly pulls 0 and 1 values from a binomial distribution (think coin flipping with 0 = tails and 1 = heads).

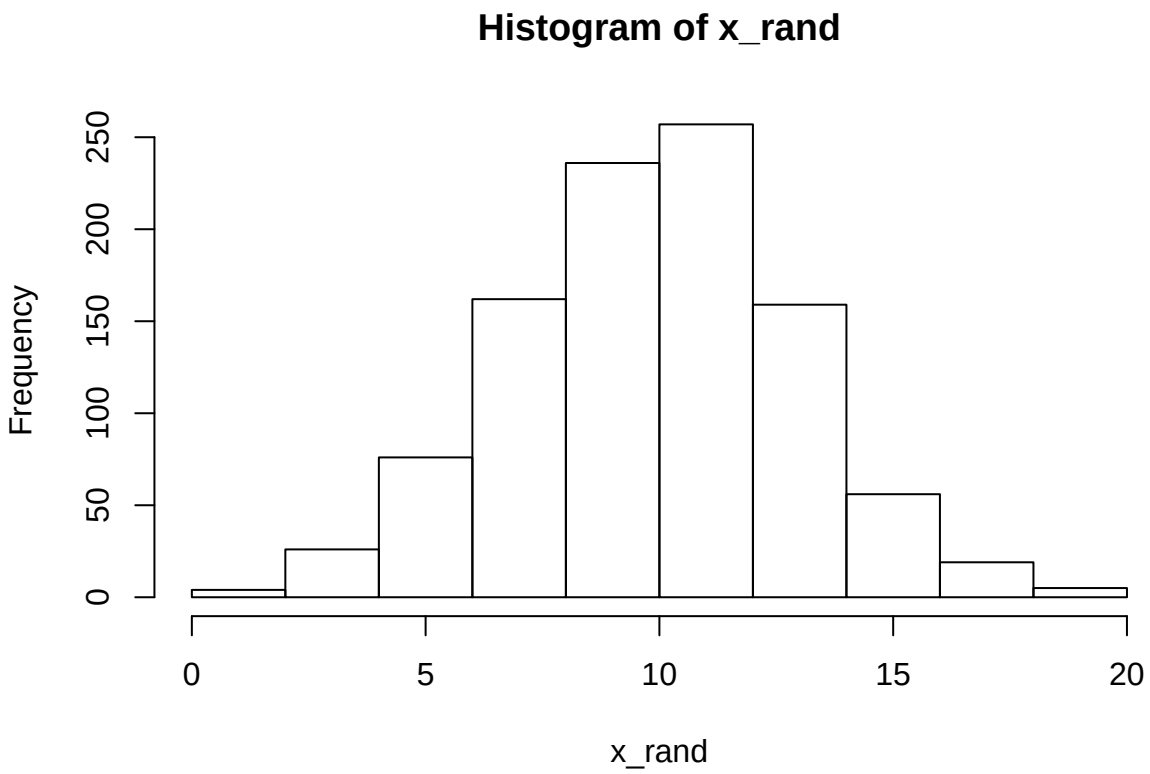
Let's first generate a random variable with `rnorm`, which we'll call `x_rand`. In `rnorm` we'll specify the number of draws (1,000) and the mean (10) and standard deviation (3) of the distribution from which to draw values:

```
x_rand = rnorm(
  n = 1000,  # number of draws
  mean = 10, # mean of the theoretical distribution
  sd = 3     # standard deviation of the theoretical distribution
)
```

To plot the empirical distribution of `x_rand` we can either conveniently use `hist` to plot a histogram, or we can use `density` to plot the probability density of `x_rand`.

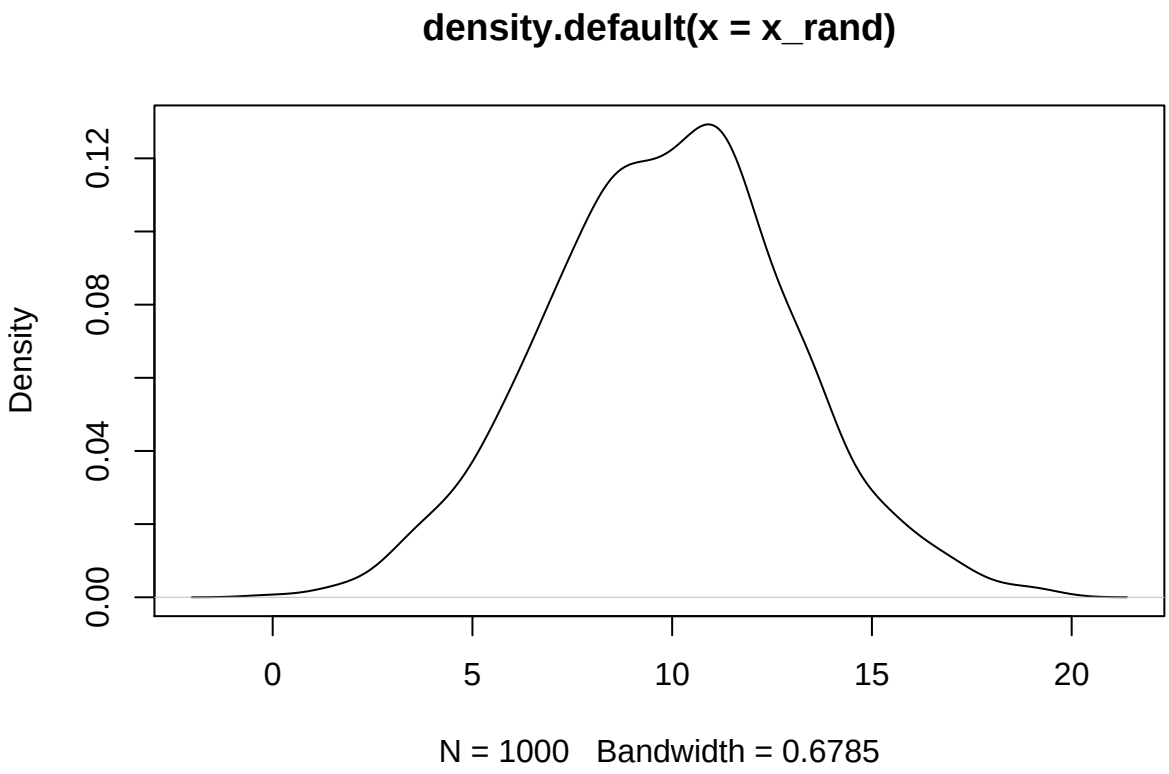
A histogram:

```
hist(x_rand)
```



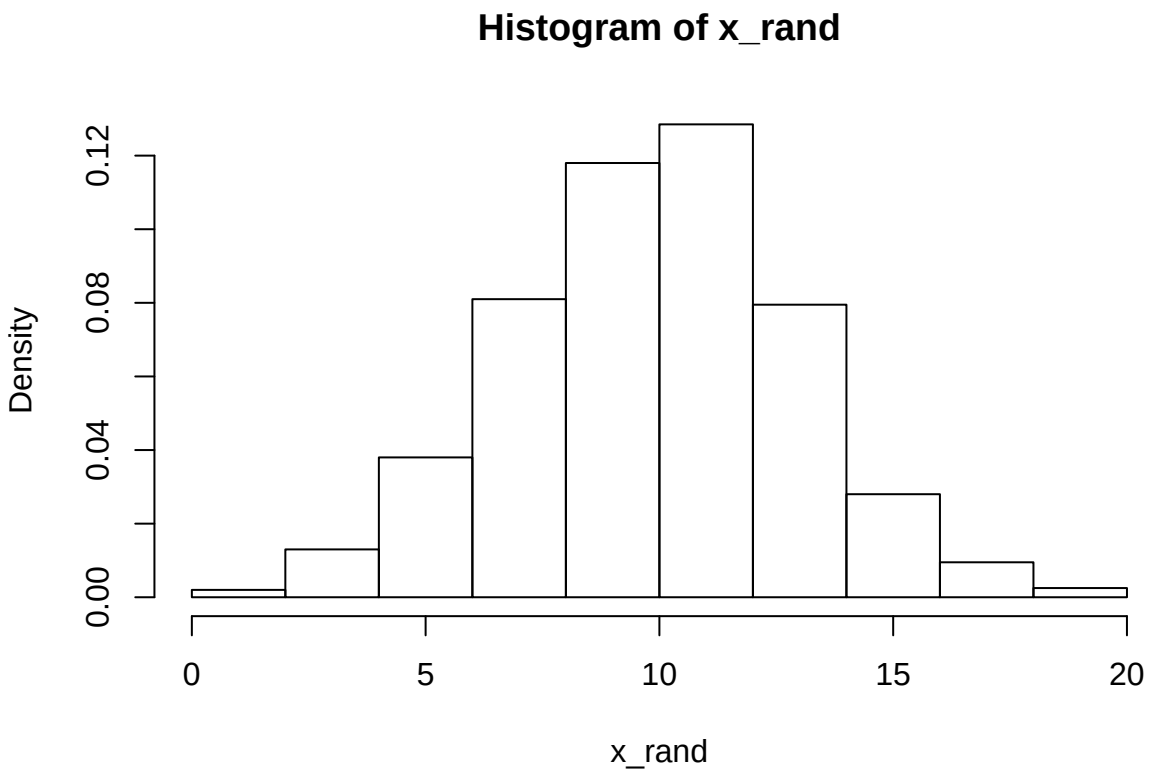
A probability density plot:

```
plot(density(x_rand))
```



Alternatively, we can use the command `probability = TRUE` to plot the probability density, rather than frequency, of values.

```
hist(x_rand, probability = T)
```

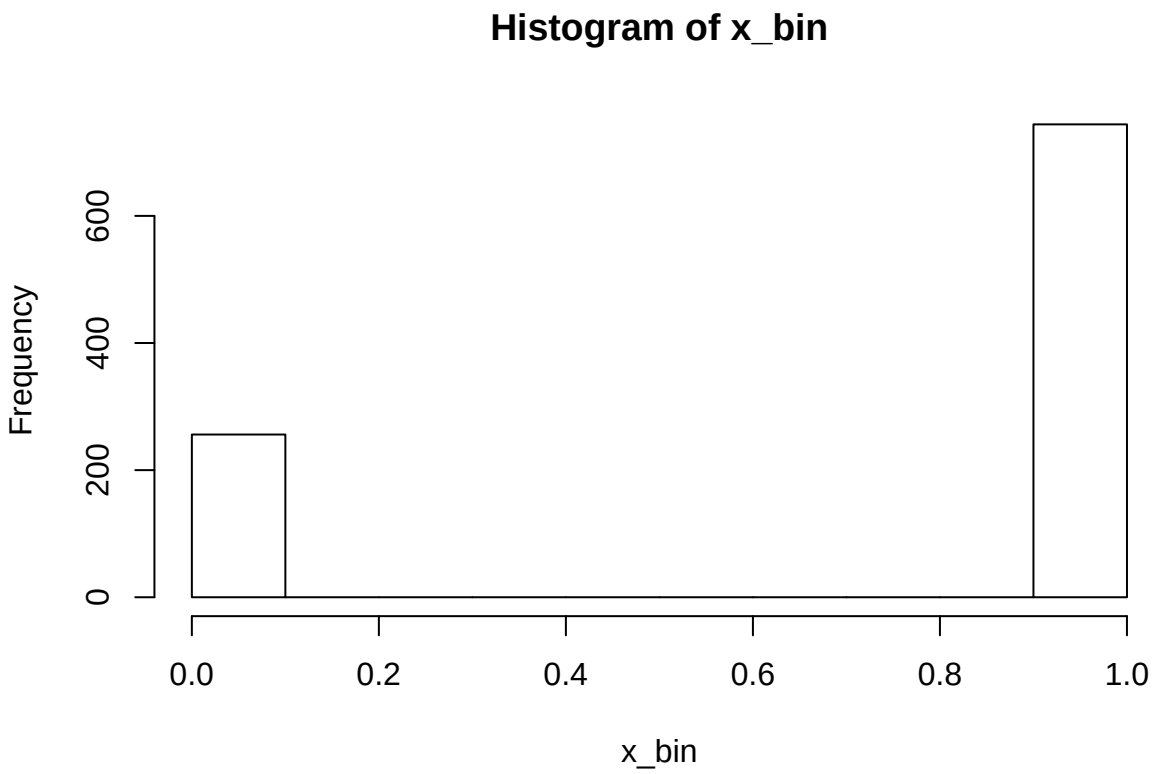


Now, let's turn to `rbinom`. This function allows us to generate values pulled from a binomial distribution. `rbinom` does this by pulling, for example, 0s and 1s from density function $p^x(1-p)^{1-x}$, where $x = 1$ and p is the probability that we choose x. Let's do 1,000 random draws from a binomial distribution, where the probability of getting $x = 1$ is .75.

```
x_bin = rbinom(
  n = 1000,    # 1,000 draws
  prob = 0.75, # probability of 1 = 0.75
  size = 1     # values to randomly draw are 0 and 1.
)
```

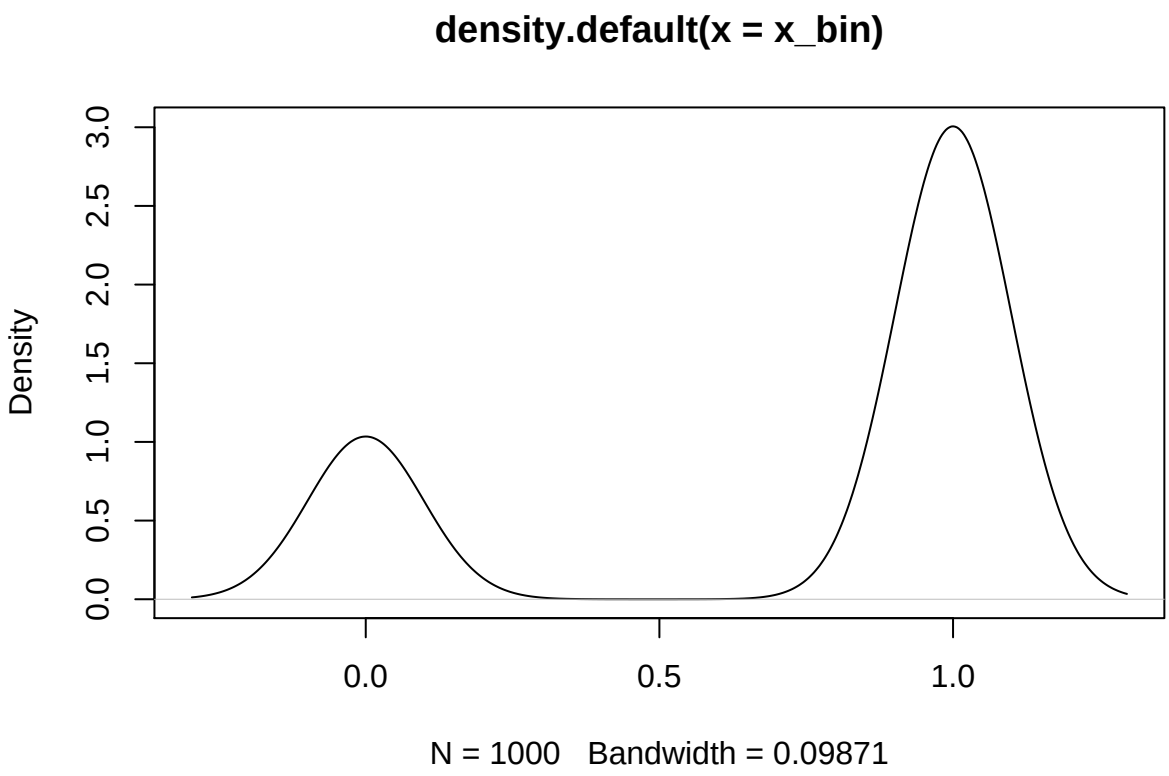
We can create a histogram of random values:

```
hist(x_bin)
```



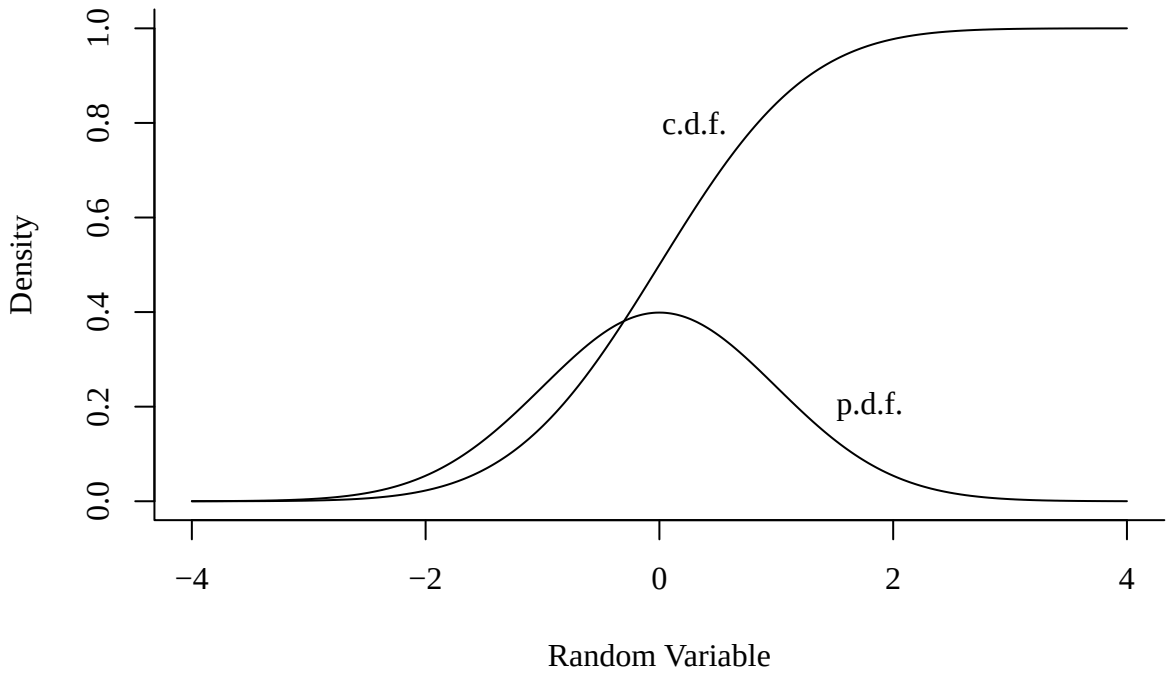
Or we can also create a density plot (though this makes less sense than a histogram since we're dealing with discrete values):

```
plot(density(x_bin))
```



Conclusion

In this document, we've covered some of the basics of working with distributions in R. Of course, we've only scratched the surface, but hopefully this gives you a good starting place going forward.


Distributions Are Fun!
Density
0.0
0.2
0.4
0.6
0.8
1.0
c.d.f.
p.d.f.
−4
−2
0
2
4
Random Variable