Chapter 5: Interactive Notebook for Students

Ram Gopal, Dan Philps, and Tillman Weyde

Summer 2022

Contents

Introduction **1**

Random Variables **1**

Sample Size **2**

Empirical Distribution Functions **4**

Mean and Variance of a Distribution **9**

Working With Distributions in R **9**

Introduction

The concepts of random variables, probability, and and distributions are central to statistics. These concepts are introduced in this chapter.

Random Variables

Consider the simplest example of tossing a coin. A priori (i.e. before you actually toss the coin) you do not know what the outcome of the coin toss is going to be. It is a **random variable** as the outcome is uncertain. Probability is the chance that an outcome of interest will occur for a random variable. Probability is always between 0 and 1. When it is 0 the event cannot occur and if it is 1 then the event is certain to occur.

Let us represent heads as 1 and tails as 0. There are only two possibilities. Therefore, the sample space, which is all the outcomes that can occur, is (0,1). This is also called **support** or **domain** of a distribution. Assuming it is a fair coin, the probability of each is 0.5. This can be represented as follows:

Outcome	Probability
1(heads)	.5
0(tails)	.5

This is called a **distribution**. A distribution identifies all possible outcomes and an associated probability for each outcome. The probabilities have to add up to 1, as one of the outcomes must occur. In other words, the outcomes are mutually exclusive and collectively exhaustive.

Let us consider another example of rolling a die.

How do we describe this distribution?

Outcome	Probability
1	1/6
2	1/6
3	1/6
4	1/6
5	1/6
6	1/6

In statistics we often describe a population in terms of a distribution. In that sense, the population is an abstract concept that is mathematically described. We just saw two simple examples.

A sample simply draws from the distribution. Let us simulate tossing a coin 10 times. We will first define the domain and the probabilities.

```r
coindomain = c(0,1)
coinprob = c(.5,.5)

sample(x = coindomain, size = 10, replace = T,prob = coinprob)
```

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

You will notice that the outcomes are different each time you run the code.

Now, let us simulate rolling a die 15 times.

```r
diedomain = seq(1,6)
dieprob = rep(1/6,6)

sample(x = diedomain, size = 15, replace = T, prob = dieprob)
```

```
##  [1] 5 1 6 1 4 2 2 3 1 5 5 5 4 2 6
```

Sample Size

Let us study the effect of sample size. Let us toss the coin ten times and from the sample we observe, let us compute the probability of heads and tails.

```r
res1 = sample(x = coindomain, size = 5, replace = T,prob = coinprob)
prop.table(table(res1))
```

```
## res1
##   0   1
## 0.4 0.6
```

Now we will toss it 1000 times and compute the probability from the sample.

```
res1 = sample(x = coindomain, size = 1000, replace = T,prob = coinprob)
prop.table(table(res1))
```

```
## res1
##     0     1
## 0.494 0.506
```

Now, let us do the same 100000 times.

```
res1 = sample(x = coindomain, size = 100000, replace = T,prob = coinprob)
prop.table(table(res1))
```

```
## res1
##      0      1
## 0.4995 0.5005
```

The **law of large numbers** as that as the sample size gets larger, the probabilities computed from the sample begin to converge to the true probability of the distribution. T

The observed difference in the probabilities between the sample and true distribution is called the **sampling error**. Let us check this out with the die example, using sizes of 5, 1000, and 100000.

```
res2 = sample(x = diedomain, size = 5, replace = T, prob = dieprob)
prop.table(table(res2))
```

```
## res2
##   1   4   5
## 0.4 0.4 0.2
```

```
res2 = sample(x = diedomain, size = 1000, replace = T, prob = dieprob)
prop.table(table(res2))
```

```
## res2
##     1     2     3     4     5     6
## 0.155 0.150 0.175 0.153 0.182 0.185
```

```
res2 = sample(x = diedomain, size = 100000, replace = T, prob = dieprob)
prop.table(table(res2))
```

```
## res2
##      1      2      3      4      5      6
## 0.1686 0.1656 0.1666 0.1678 0.1668 0.1645
```

Empirical Distribution Functions

The strategy we will follow to get the empirical distribution is the following:
1. Write R code to simulate one outcome.
2. Put this in a function.
3. Replicate running the function a large number of times to get the empirical distribution.

Let us write the code.

```r
coindomain = c(0,1)
coinprob = c(.5,.5)

f1 = function(){
  y = sample(x = coindomain, size = 50, replace = T,prob = coinprob)
sum(y)
}
```

Let us try the function.

```r
f1()
```

```
## [1] 22
```

Now, we need to run the function, say, 10000 times, to get a large sample of outcomes. Let us try the `rep()` function as we did before.

```r
rep(f1(),10)
```

```
##  [1] 28 28 28 28 28 28 28 28 28 28
```

You will notice it does not work. The problem with this is it runs the function only once and repeats the outcome 10 times. What we want is to run the function 10 times, and not run the function once and repeat the outcome many times. For this, we use a function called `replicate()`.

```r
replicate(n = 10,f1())
```

```
##  [1] 27 22 28 31 31 25 24 23 29 24
```

Now we are ready to create the empirical distribution function.

```r
s1 = replicate(n = 10000, f1())
edf1 = prop.table(table(s1))
```

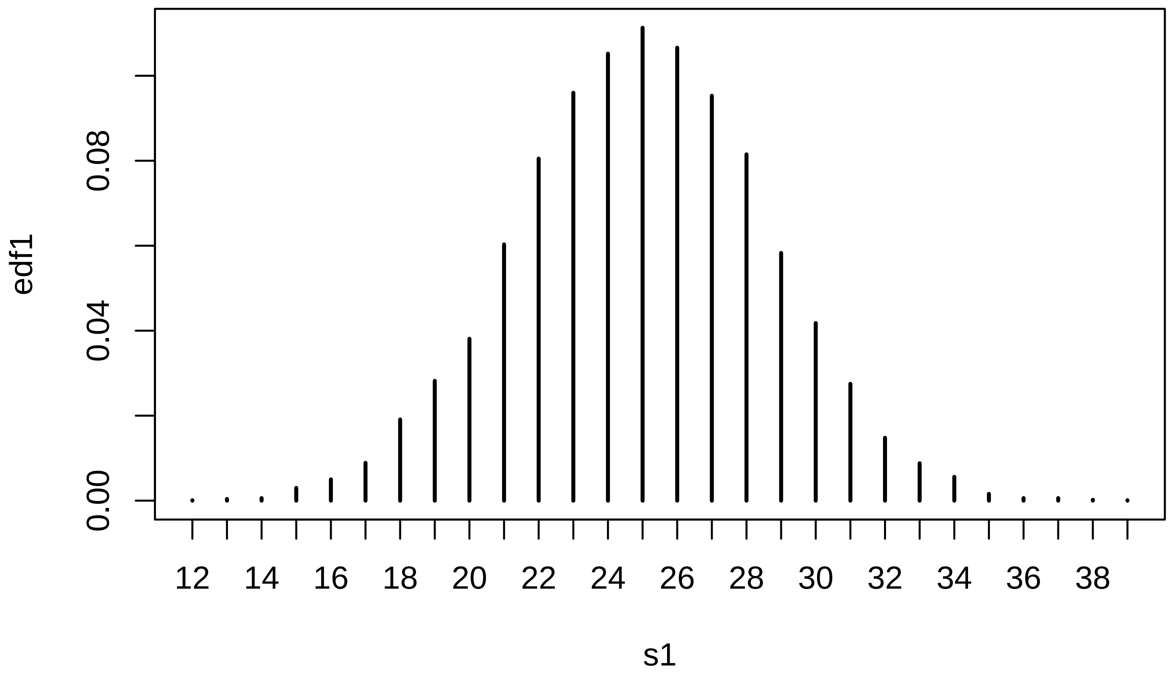
Now you can see the distribution function.

```r
edf1
```

```
## s1
##     12     13     14     15     16     17     18     19     20     21     22 
## 0.0001 0.0004 0.0006 0.0030 0.0050 0.0089 0.0191 0.0282 0.0381 0.0603 0.0805 
##     23     24     25     26     27     28     29     30     31     32     33 
## 0.0960 0.1052 0.1113 0.1066 0.0953 0.0815 0.0583 0.0418 0.0275 0.0148 0.0088 
##     34     35     36     37     38     39 
## 0.0056 0.0016 0.0006 0.0006 0.0002 0.0001
```

Now, let us plot this.

```r
plot(edf1,type = "h")
```



We can now compute any probability we wish from this empirical distribution. Let us compute the probability that the number of heads will be less than or equal to 20. Notice that our vector `s1` contains the outcomes from 10000 samples. We simply need to count how many of them satisfy the condition to get our probability.

```r
length(s1[s1<=20]) / (length(s1))
```

```
## [1] 0.1034
```

Let us compute the probability of getting 15 or more heads.

```r
length(s1[s1>=15]) / (length(s1))
```

```
## [1] 0.9989
```

Now let us try something a bit more complicated. We want to repeatedly keep adding the numbers we get from each roll. The outcome of interest is the number of times you have to roll the die until you get a sum of 35. The domain here is between 6 and 35, because the minimum number of rolls needed to obtain a sum of 35 is 6, and the maximum is 35.

```r
diedomain = seq(1,6)
dieprob = rep(1/6,6)

v1 = sample(x = diedomain, size = 35, replace = T, prob = dieprob)
c1 = cumsum(v1)
which(c1 > 35)[1]
```

```
## [1] 10
```

There are two functions we used here. The first is `cumsum()`, which creates a vector whose ith element is the sum from x[1] to x[i]. The second is `which(x>c)`, which gives you the index of all the elements of vector x which are greater than the number c. We want the first time we see a number greater than 35. Therefore, we used `which(c1 > 35)[1]`. Now, let us put this into a function.

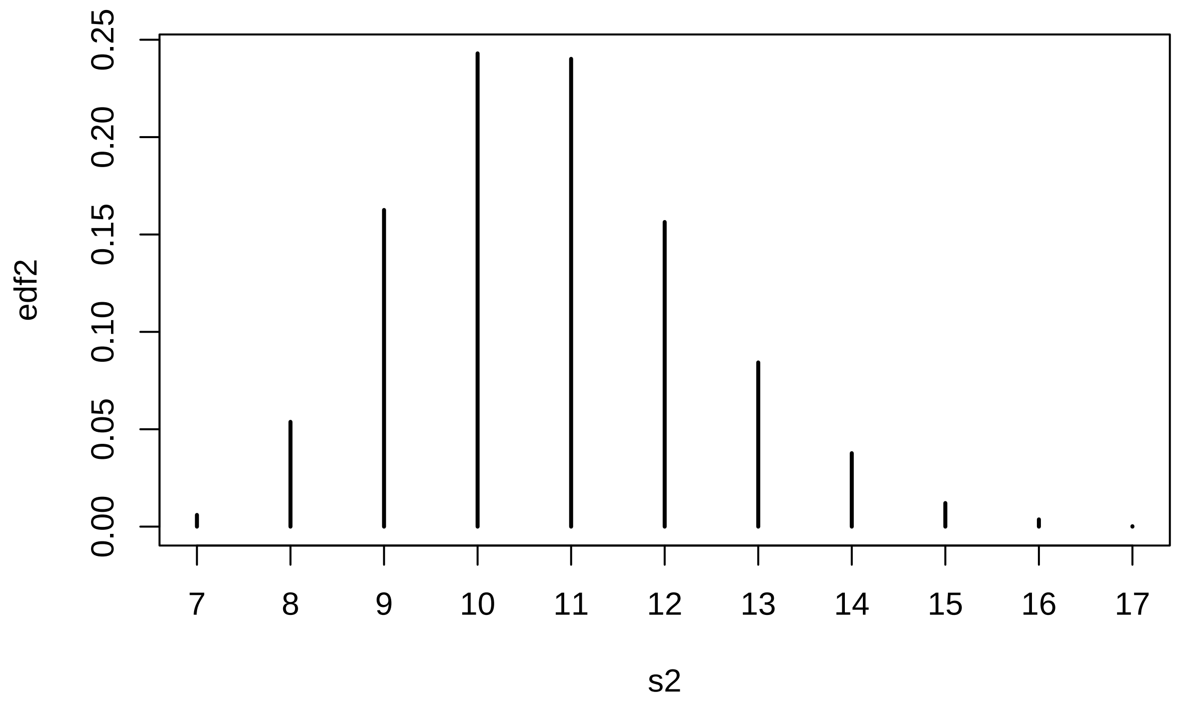
```r
f2 = function(){
  v1 = sample(x = diedomain, size = 35, replace = T, prob = dieprob)
c1 = cumsum(v1)
which(c1 > 35)[1]
}
```

Let us replicate to create the empirical distribution.

```r
s2 = replicate(n = 10000, f2())
edf2 = prop.table(table(s2))
edf2
```

```
## s2
##      7      8      9     10     11     12     13     14     15     16     17
## 0.0060 0.0538 0.1626 0.2430 0.2402 0.1564 0.0843 0.0377 0.0121 0.0037 0.0002
```

```r
plot(edf2, type = "h")
```



From the plot we see the high probability events are 10 and 11 rolls of the die.

What is the probability that we will hit 35 in less than or equal to 8 rolls?

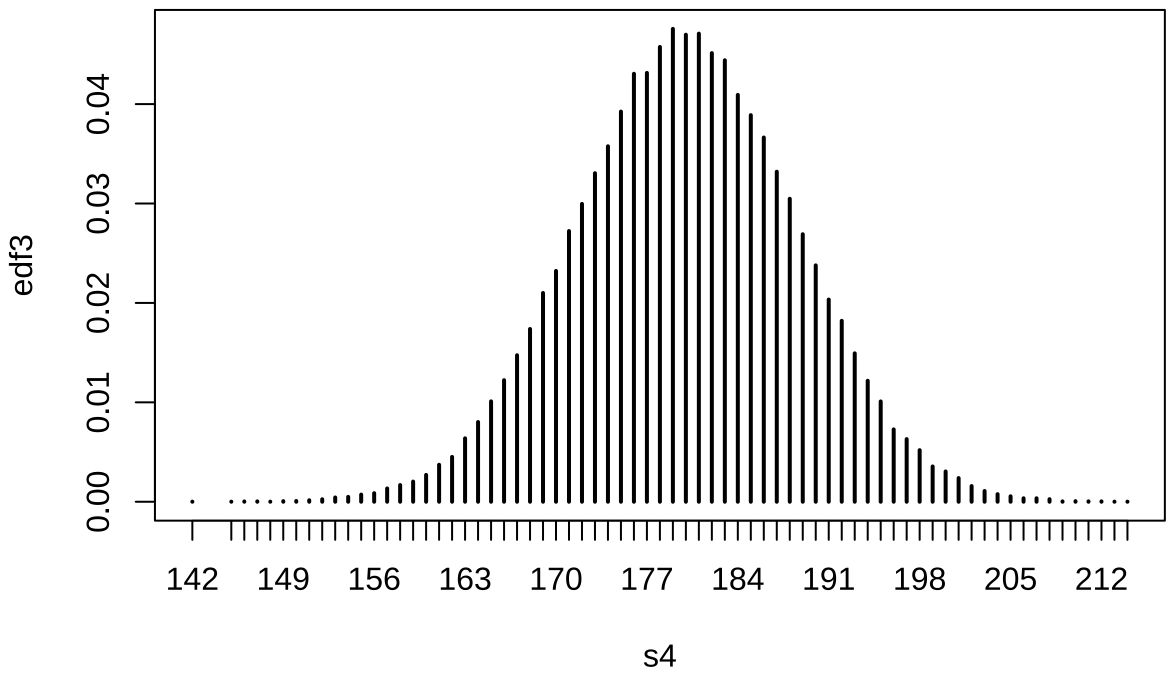
```
length(s2[s2<=8])/(length(s2))
```

```
## [1] 0.0598
```

Suppose a drug manufacturer guarantees that a particular drug has a .6 probability of working. If the drug was administered to 300 patients, we want to compute the probability that no more that 160 patients improved after taking the drug. This example is similar to the previous coin toss example. However, this time, the number of tosses and the success probability are different. We will create a more general function where we can specify the number of trials and the success probability as inputs to the function. These are the **parameters** of the distribution. The following code creates the empirical distribution function.

```
drugdomain = c(0,1)
f3 = function(sz, successprob){
  drugprob = c(1-successprob, successprob)
  v2 = sample(x = drugdomain, size = sz, replace = T, prob = drugprob)
  s3 = sum(v2)
  return(s3)
}

s4 = replicate(n = 100000, f3(300,.6))
edf3 = prop.table(table(s4))
plot(edf3, type = "h")
```



Now we can compute the probability that no more than 160 patients are cured.

```
length(s4[s4<=160])/length(s4)
```

```
## [1] 0.01089
```

The probability is very small.

Let us think about the following: suppose you know that a medical facility administered the drug to 300 patients and no more than 160 patients actually improved after taking the drug. What can we say about this situation?

1. The probability of no more than 160 patients getting better is quite small. Even though this is unlikely, it is not impossible.This could just be bad luck. For example, you could toss a fair coin ten times and get no heads. The probability is very small but it's not zero. If I am another medical facility, how would I process this information? To rule out the possibility that it was just bad luck, perhaps I would want to see more evidence. This can come from procuring a larger sample size.
2. If you believe the sample size is big enough as it is, then you would not attribute the outcome to just bad luck. You will begin to question the claims made by the drug manufacturer. Perhaps their claim that the drug is effective 60% of the time is not true. This is the core idea behind **hypothesis testing**. In this case, the drug manufacturer's claim of 60% success constitutes the null hypothesis. The sample data allows use to either reject the null hypothesis or not reject the null hypothesis. Clearly, in this case we would reject the null hypothesis as the probability we computed was very small. We will expand on this concept later.

For now, just note that the null hypothesis describes the population, and the population is described by a distribution. More on this later.

Mean and Variance of a Distribution

As we described before, a discrete distribution is defined by a support, which is the set of all possible outcomes, and by the corresponding probability of each outcome. Let `x_1,...x_N` denote the possible outcomes. Let `P(x_i)` denote the probability of outcome `x_i`. The mean and variance are defined as:

$$\mu = \sum_{i=1}^{N} x_i P(x_i)$$

$$\sigma^2 = \sum_{i=1}^{N} (x_i - \mu)^2 P(x_i)$$

Let us compute the mean and variance of a distribution.

```
ddomain = seq(1,4)
dprob = c(.7,.1,.1,.1)
meandistribution = sum(ddomain * dprob)
meandistribution
```

```
## [1] 1.6
```

```
vardistribution = sum((dprob)*(ddomain-meandistribution)^2)
vardistribution
```

```
## [1] 1.04
```

Let us summarize what we know about distributions.

1. The distribution describes all possible outcomes and the corresponding probability for each outcome. Each distribution has its support, which describes the possible outcome values. For tossing a coin, it is 0 and 1. For rolling a die, it is 1 to 6.

2. A distribution is used to describe a population of interest. In this sense, it is somewhat of an abstract concept.

Working With Distributions in R

R provides easy ways to work with various distributions (see the short reference card). For a given distribution (**dist**), you can get a variety of information in R. We precede the distribution name with the letters **r** (for random values from the distribution), **d** (probability or density value), **p** (cumulative probability), or **q** (value of quantile). Check the R reference card for a list of common distribution functions.

Recall the drug example with 300 trials and a success rate of .6. The probability of having no more than 160 cured patients was 0.0116. Let us do this using the above.

```
pbinom(160,300,.6)
```

```
## [1] 0.01118
```

As you can see, the answers are very close.

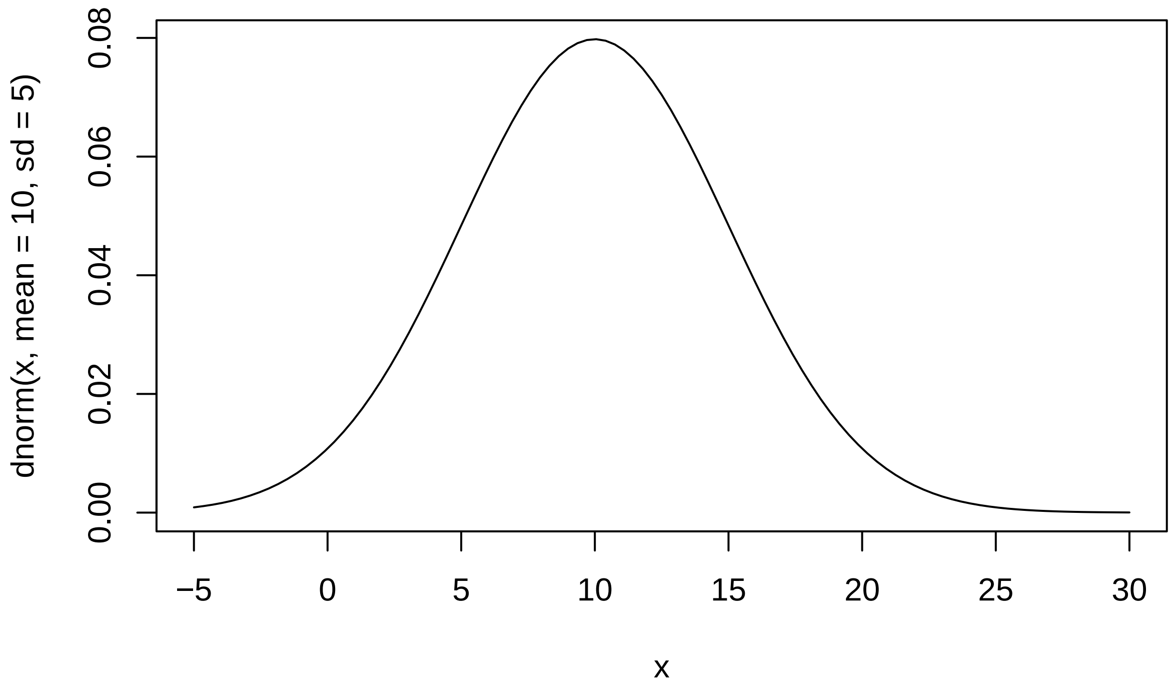
Suppose you want to create 20 random variables from $N(6, 3)$.

```
rnorm(20,mean = 6, sd = 3)
```

```
##  [1]  3.5182  8.6337  5.7652  5.0280  8.5320  6.9490  7.4688 10.0425  9.7360
## [10]  7.4707  7.5684 11.0377  7.0600 -0.6240 -0.5379 11.0063  9.8604  4.8964
## [19]  3.9647  3.9663
```

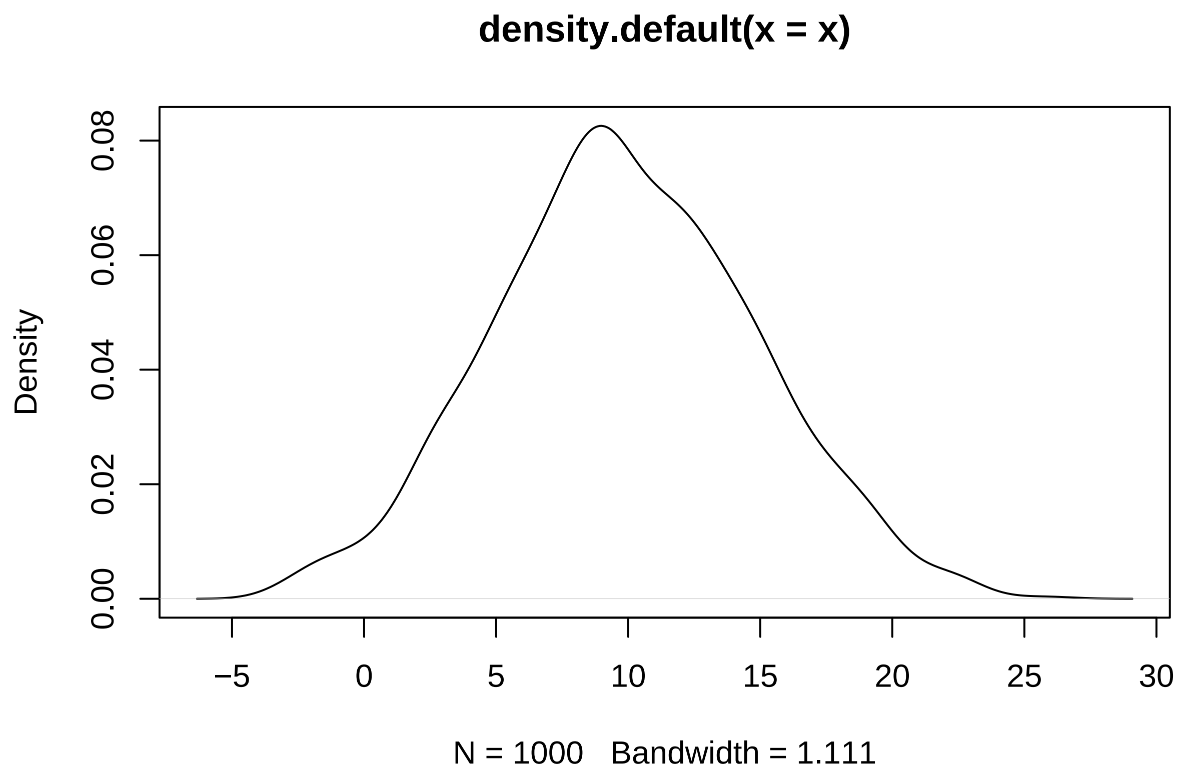
Let us plot a normal distribution.

```
curve(dnorm(x,mean=10, sd = 5),-5, 30)
```



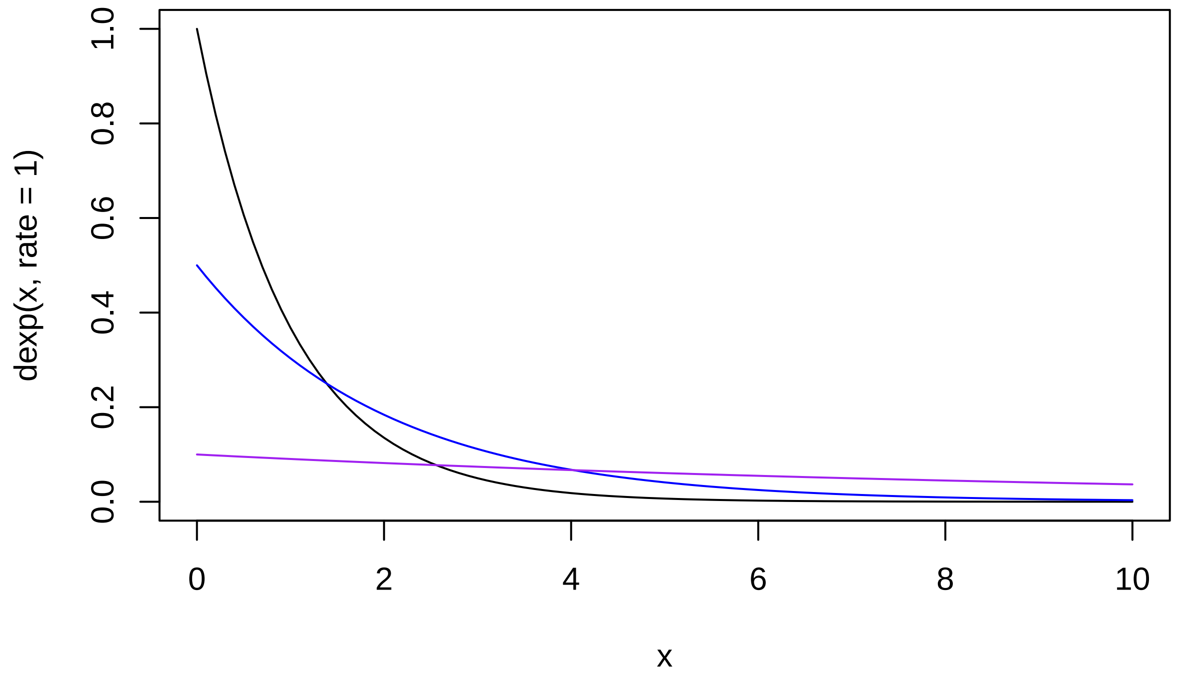
Another way to plot this is to create random numbers from the distribution and plot the density.

```
x = rnorm(1000,mean = 10, sd = 5)
plot(density(x))
```



Let us see what an exponential function looks like.

```
curve(dexp(x,rate = 1), 0, 10)
curve(dexp(x,rate = 1/2), 0, 10,add = T, col = "blue")
curve(dexp(x,rate = 1/10), 0, 10, add = T, col = "purple")
```



Let us say you want to get the 75^{th} percentile value.

```
qexp(.75, rate = 1/2)
```

```
## [1] 2.773
```