

Computing With R – Handout 2

The purpose of this handout is to provide you with R commands that will allow you to plot some of the more common probability mass functions for discrete random variables and some of the more common density functions for continuous random variables, as well as distribution functions for these cases.

Plotting Probability Mass Functions

Step 1. Creating a Vector of Possible Values.

For plotting probability mass functions we first need a vector of possible values. In many cases, such as for the Poisson distributions illustrated below, the possible values extend to infinity. In such situations it is helpful to know the location of the distribution, as this is where the majority of the probability mass will occur, but we have not covered this yet in class. So, for our purposes here we will simply guess as to an appropriate range of values. That is, for any probability mass function that extends to infinity, the probabilities of possible values will tend to (but never reach) zero. Why this is the case we will discuss soon. As an introduction, suppose the set of possible values is the set of non-negative integers, and we believe most of the probability will occur around a value of about 5. We might then create a vector of possible values as,

```
> ys<-0:10
```

Step 2. Evaluate the Probability Mass Function at Each Possible Value

There are two major ways to accomplish this step. The easiest, when it is possible, is to use built-in functions in R to evaluate the mass function at each value created in Step 1. For example, suppose we wish to evaluate a Poisson probability mass function with parameter 5 (more on this soon) at each of the values 0, 1, . . . , 10 as saved in the object called “ys” above. We could accomplish this as,

```
> fs<-dpois(ys,lambda=5)
```

Here, the function “dpois” is a built-in R function that evaluates the Poisson probability mass function at each value of the vector “ys”, using a parameter value of 5 (R calls this parameter “lambda”). The “d” in “dpois” stands for “density”, which R uses for both discrete and continuous distributions. There are other forms of this function that compute other things:

ppois computes cumulative probabilities or distribution function values
rpois generates simulated values from a Poisson distribution
qpois calculates quantiles for given probability values

Since the syntax for other distributions is very similar to these functions, it is good to understand it at this point. The syntax given in the R Help files is

```
dpois(x, lambda, log = FALSE)
ppois(q, lambda, lower.tail = TRUE, log.p = FALSE)
qpois(p, lambda, lower.tail = TRUE, log.p = FALSE)
rpois(n, lambda)
```

x vector of (non-negative integer) quantiles.

q vector of quantiles.

p vector of probabilities.

n number of random values to return.

lambda vector of positive means.

log, log.p logical; if TRUE, probabilities p are given as log(p).

lower.tail logical; if TRUE (default), probabilities are $P[X \leq x]$, otherwise, $P[X > x]$.

The function arguments followed by an = sign are set to their default values as shown unless you specify them. That is, if the argument “log” is not given in a call to the function dpois it automatically defaults to the value “FALSE”.

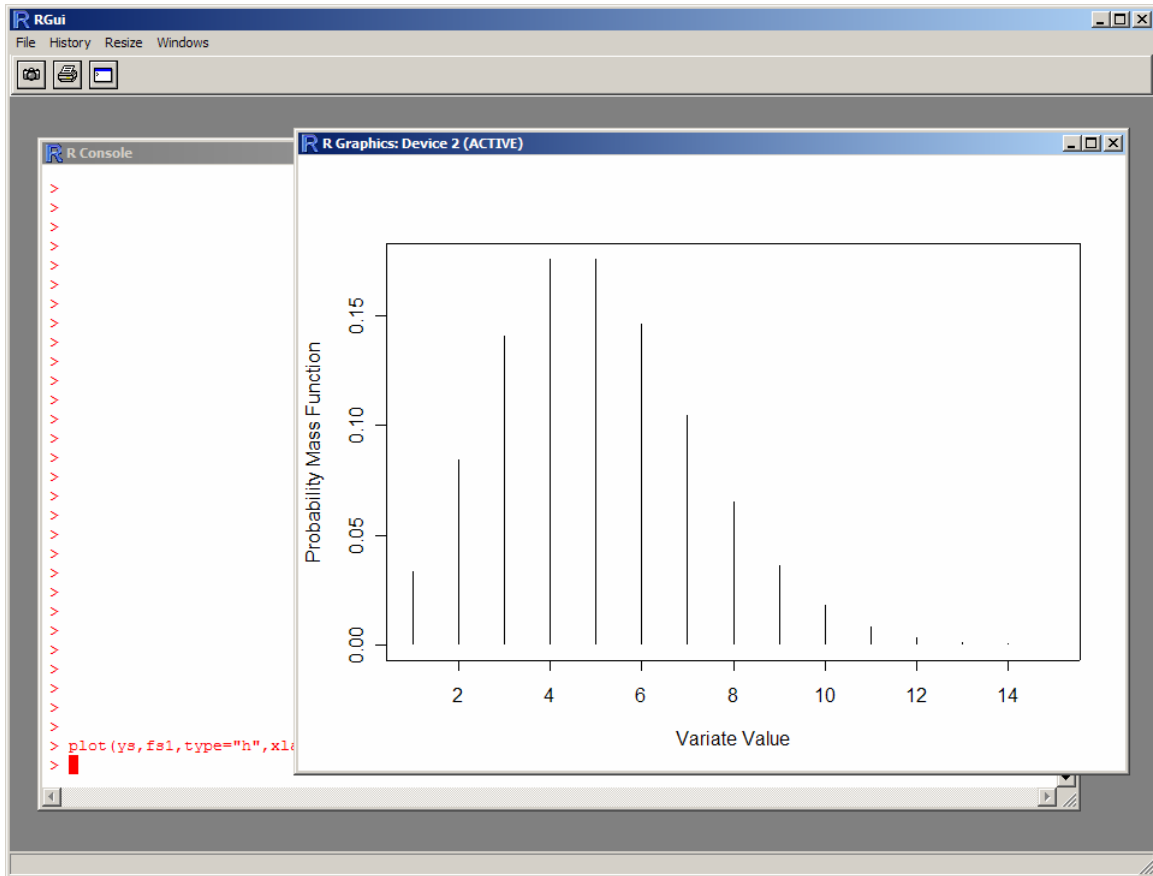
The second way to evaluate probability mass functions is to compute them from the formula for a probability mass function (or density for continuous distributions). While more involved than using built-in functions, it is useful to know how to do this because not all distributions you might ever want to know about are contained in the built-in functions provided. The probability mass function for a Poisson random variable with parameter lambda may be written as

$$f(y|\lambda) = (1/y!) \lambda^y \exp(-\lambda)$$

To compute this for our vector of values “ys” from Step 1 with lambda=5, we could compute values as,

```
> fs<- (1/c(1,cumprod(ys[-1])))*(5^ys)*(exp(-5))
```

The most complicated portion of this is the first part, which is one way to get a vector of factorial values by using a cumulative product function (“cumprod”). The major complication is that the first element of the vector “ys” is 0 and so if it is left in then “cumprod” simply produces a vector of zeros. So, the “ys[-1]” is used as the argument to “cumprod”, which says use the vector “ys” except for the first element. But then we need to stick a value of 1 back on to the vector for 0! which is accomplished by the



Notice that this distribution is located roughly around a value of 5, the value of the parameter lambda in the probability mass function.

To create this plot we have used the optional argument `type="h"` in the plot command, which tells the command to produce a line graph. If we just left this argument off, we would get points at the mass function values (try it to see).

Plotting Probability Density Functions

Plotting a probability density function for a continuous random variable follows much the same routine as does plotting a probability mass function for a discrete random variable. The major difference comes in construction of the plot. We will illustrate this for a random variable having an exponential distribution. The parameter of an exponential distribution is called "rate" by R, and the location of this distribution turns out to be the reciprocal of this parameter. Let's produce the density function for a distribution with location of 0.5:

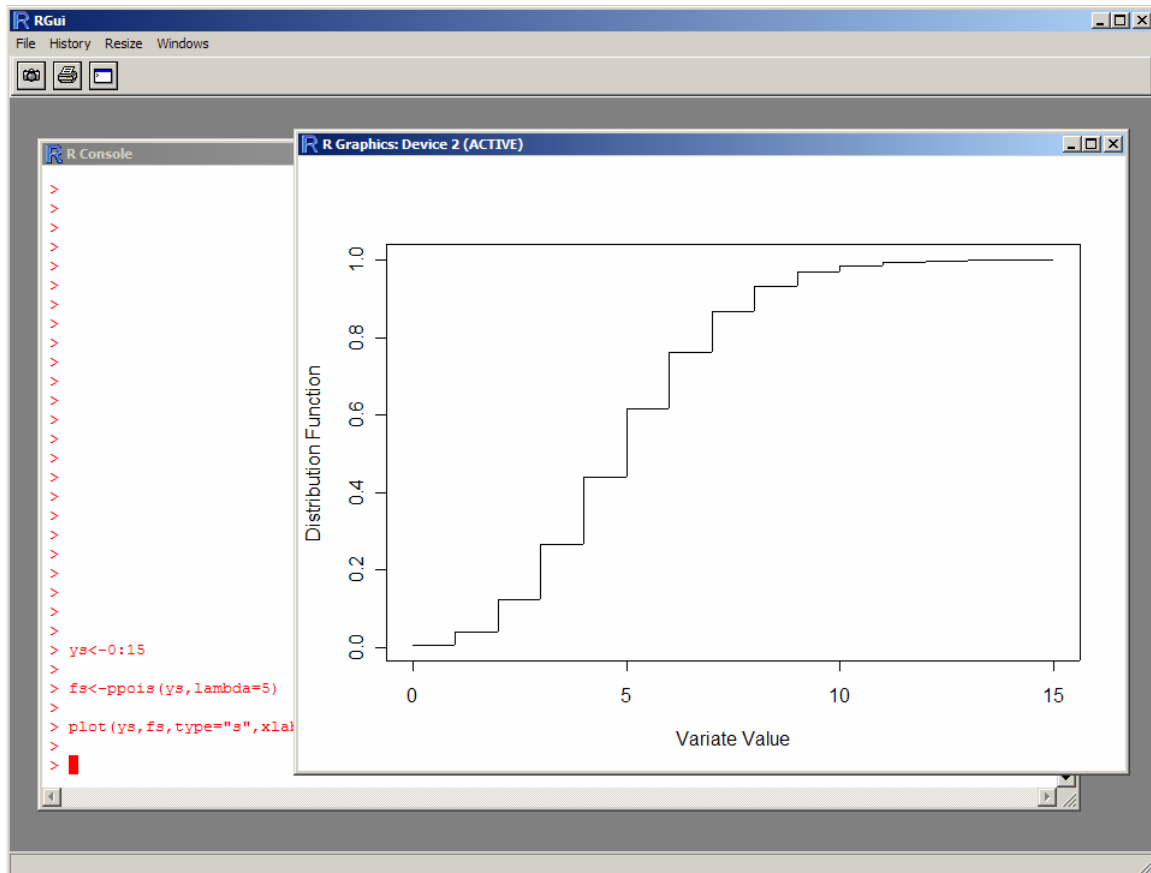
Step 1.

```
> ys<-0.01*(1:500)
```


Plotting Distribution Functions

To obtain graphs of distribution functions we do much the same thing again, changing only a few details. For the Poisson distribution we would use the following commands:

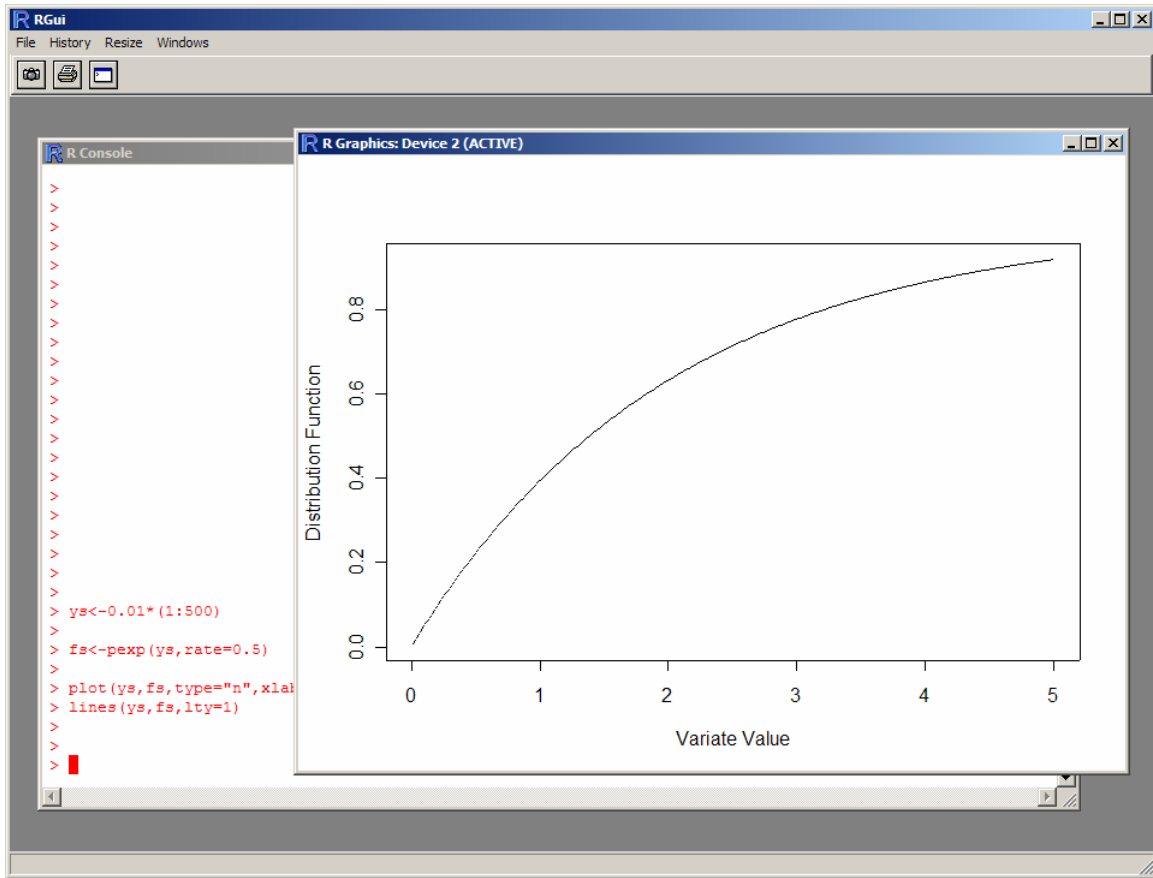
```
> ys<-0:15
> fs<-ppois(ys,lambda=5)
> plot(ys,fs,type="s",xlab="Variate Value",ylab="Distribution Function")
```



Here the command “ppois” computes cumulative probabilities (the distribution function) rather than the mass function values produced by “dpois”. The type=”s” argument in the plot creates a step function, as opposed to the lines produced by the type=”h” argument used previously.

To obtain the distribution function (or cumulative density function) for our exponential example we would enter the commands:

```
> ys<-0.01*(1:500)
> fs<-pexp(ys,rate=0.5)
> plot(ys,fs,type="n",xlab="Variate Value",ylab="Distribution Function")
> lines(ys,fs,lty=1)
```



The only difference between this and what we did to graph the exponential density function was to use the function “pexp” rather than “dexp”.