Mathematics 375 – Probability Theory
Computer Lab Day 2 – Probability Distributions in R
October 19, 2011

Goals

The goals of today's lab are:

- to gain some more practice with basic features of the R statistics package, and
- to reinforce intuition about cumulative distribution functions (CDFs), and probability density functions (PDFs) for continuous random variables.

Basic R Mechanics

Refer to the assignment sheet for the first lab day if you need to refresh your memory about running R in Haberlin 136, about how to save and print your results, etc. The Lab 1 assignment is posted on the course homepage, so you can open it and look at the instructions there as needed.

Background on Probability Distributions in R

The R package contains built-in functions for computing probabilities and cumulative density functions for many of the discrete random variables we have discussed, plus functions for the probability densities and cumulative distributions of many of the standard types of continuous random variables that we will study this semester and next. Here's how it works. (Also look at table on 332 of Dalgaard *Introductory Statistics with R.*) Each type of random variable is covered by a *family* of 4 functions distinguished by a *prefix letter* `d, p, q`, or `r`:

- `d` – the probability function in the discrete case, or the density function continuous case,
- `p` – the cumulative distribution
- `q` – the quantile function (essentially the *inverse function* of the cumulative distribution, but of course the cumulative distribution can fail to be injective, so this needs to be taken with a grain of salt)
- `r` – random number generator

Following the prefix letter comes the rest of the function name and the inputs needed to compute the corresponding values. For instance the family of functions for Poisson random variables (discrete) looks like this:

- `dpois(y,lambda)` – the probability function $P(Y = y)$ for $Y \sim Poisson(\lambda)$; the first input is the y (which must be an integer), the second is the parameter λ.
- `ppois(y,lambda)` – the cumulative distribution $P(Y \leq y)$

- `qpois(q,lambda)` – computes the y such that $P(Y \leq y) = q$
- `rpois(N,lambda)` – generates a list of N random numbers following a $Poisson(\lambda)$ distribution

The distributions that we will use today have these names (after the prefix letter explained above):

- `binom` – binomial (need to supply a y, q, or N as above, and the n, p values)
- `norm` – normal (need to supply a y, q, or N as above, and values for `mean` and `sd` parameters)
- `exp` – exponential (need to supply a y, q, or N as above, and a value for the `rate` parameter)

Today's Lab

Visualizing Probability (or Density) and Distribution Functions

Work through the following examples before attacking the question A. below. You do not need to hand in any work for this preliminary discussion, just for the question.

1. The *probability histogram* for a discrete random variable that we saw in Chapter 3 can be viewed as a sort of graph of the probability function. This can be can be drawn in `R` as follows. Suppose for instance that $Y \sim Binomial(50, .3)$. The y for which $P(Y = y) \neq 0$ are $0 \leq y \leq 50$, so we begin by creating that sequence of y-values, then plot the `dbinom` function like this:

```
y <- seq(0,50,1)
plot(y,dbinom(y,50,.3),type="h")
```

   This should all be relatively self-explanatory after you execute the commands and see the graph. (If you don't understand what the y is here, just enter the command `y` to print out what is in that variable.) You want to think of the vertical "pins" as representing the boxes in the probability histogram. The `type = "h"` indicates this style of plotting. What happens if you change that to `type = "l"` (that's an ell, not the digit 1), or leave out the type option all together?

2. Now, to see the cumulative distribution function, enter

```
plot(y,pbinom(y,50,.3),type="h")
```

   Note how this relates to the general properties we discussed last time. The same types of commands will work for any of the functions described above. For instance, these commands

```
y <- seq(-4,4,0.1)
plot(y,dnorm(y,0,1),type="l")
```

will plot the graph of the probability density of a random variable with a normal distribution with mean 0 and sd (standard deviation) 1. Of course, we have not discussed these in class yet, but for now you can just take this graph as a sort of *definition* for what that type of continuous random variable looks like.

3. In the normal family of random variables, the mean can be any real number and the sd can be any positive number. You may want to experiment with changing those and the `y` in the commands immediately above so that you get a good understanding of how the density function changes.

4. As you do this, you should notice that it is necessary to change the set of values supplied for plotting to get reasonable graphs. If you want to have `R` choose the values automatically, you can use another command called `curve`. For instance

```
curve(dnorm(x,0,1))
```

will do essentially the same thing as the sequence of two commands for the `dnorm(y,0,1)` function above. *Note:* In the `curve` command the "variable" in the `d`-function (or `p`-function) always has to be called `x`, though. Another nice feature of the `curve` command is that you can add an option `add = T` inside the parentheses if you want to overlay the curve on the active plot. See p. 42-43 in the R book for a detailed example.

Question A.

The probability histogram for $Y \sim Binomial(50, .3)$ is actually very close to the graph of the density function for a normal random variable with some mean and sd. Thinking about what we know about binomial random variables, determine the appropriate mean and sd values, and generate a composite plot showing the probability histogram overlaid with the graph of the normal density. In the output you submit, include the commands that generated the output, and an explanation of how you determined which mean and sd values to use.

Sampling from Given Distributions

The `r-` function in the family for each type of distribution as described above generates lists of random numbers following the corresponding distribution. This means, for instance, that for the continuous types, the probability that each number comes out in a given interval is determined by the density function:

$$P(a \leq Y \leq b) = \int_a^b f(y)\ dy.$$

In this problem, we will generate two collections of 1000 random numbers, one from a normal distribution with mean 1 and sd 1, and the other from an exponential distribution with rate 1. We

have not discussed these yet formally either, but the exact formula for the exponential density is one we used in an example in class on Friday:

$$f(y) = \begin{cases} e^{-y} & y > 0 \\ 0 & y \leq 0. \end{cases}$$

Question B.

1. You can get an idea of what this all means by plotting the probability densities as in question A. Generate these plots and write a short description of the differences between these two distributions.

2. Enter the commands

```
nr <- rnorm(1000,1,1)
er <- rexp(1000,1)
```

   These generate two collections of 1000 random numbers, one from an normal distribution with mean 1 and sd 1, and the other from an exponential distribution with rate 1. Suppose you did not know which list had been produced which way. Thinking about what we know about normal distributions from the "empirical rule," determine a way to test which is which by considering the intervals $\overline{y} \pm ks$ for $k = 1, 2, 3, ...$ Carry out your calculations for both lists `er` and `nr` and discuss your results. *Note:* If you reexecute the `rnorm` and `rexp` commands, you will be generating different lists of random numbers each time. So the values of the (sample) mean and (sample) sd will be changing each time. However, they should be quite similar and consistent enough that your conclusion does not change.

Assignment

Lab reports containing input, output, and answers to the questions posed above are due in class no later than Monday, October 24. If you do not finish during the hour today, you can return to HA 136 any time it is not in use by another class.