Exercise #3: Other probability distributions and random sampling

Patrick Applegate, patrick.applegate@psu.edu

26 October 2015

Learning Goals

After completing this exercise, you should be able to

- generate random numbers from various probability distributions in R
- construct histograms and empirical cumulative distribution functions from sets of random numbers
- describe how changing the parameters of various probability distributions affects the shape of the distributions

Introduction

Risk analysis often involves simulating processes that have stochastic (that is, random) components. Many of the random elements within computer models can be represented by normal distributions; however, as we saw in Part 2 of the last exercise, not all data sets are normally distributed. We need a set of methods for simulating random effects within numerical models, while accounting for the possibility that these random effects may not obey normal distributions.

But, what is a probability distribution? Loosely speaking, a probability distribution function (pdf) is a mathematical relationship that ties values of a quantity x to the chance of observing particular values of that quantity. These chances (probabilities) range from 0, indicating that the associated value of x is impossible, to 1, indicating that the associated value of x must occur on any given trial. Assuming that *something* occurs on any given trial, integrating these probabilities over all possible values of x must yield 1. Probability distributions have parameters that change the distribution's shape, sometimes drastically.

In this exercise, you'll perform random sampling from several probability distributions and examine two ways of plotting the results from this random sampling. You'll also see how changing the parameters of statistical distributions change the distributions' shape.

Tutorial

If you haven't already done so, download the .zip file containing the scripts associated with this book from www.scrimhub.org/raes. Put the file `lab3_sample.R` in an empty directory. In RStudio, open the file `lab3_sample.R` and change R's working directory to the location of that file on your hard drive. Next, execute the script (look back at Exercise 1 if you need help with changing R's working directory or with executing the script). A directory called `figures` should appear that contains a file called `lab3_sample_plot1.pdf`. Open this file and look at the contents. This file should contain a pair of plots that resembles Figure 1.

Histograms and empirical cumulative distribution functions

In Figure 1, the top panel shows a histogram, and the lower panel shows an empirical cumulative distribution function (ecdf). These plots both display the same set of randomly-generated synthetic data, in this case $n = 10^4$ random draws from a normal distribution with mean μ of 5 and a standard deviation σ of 3.

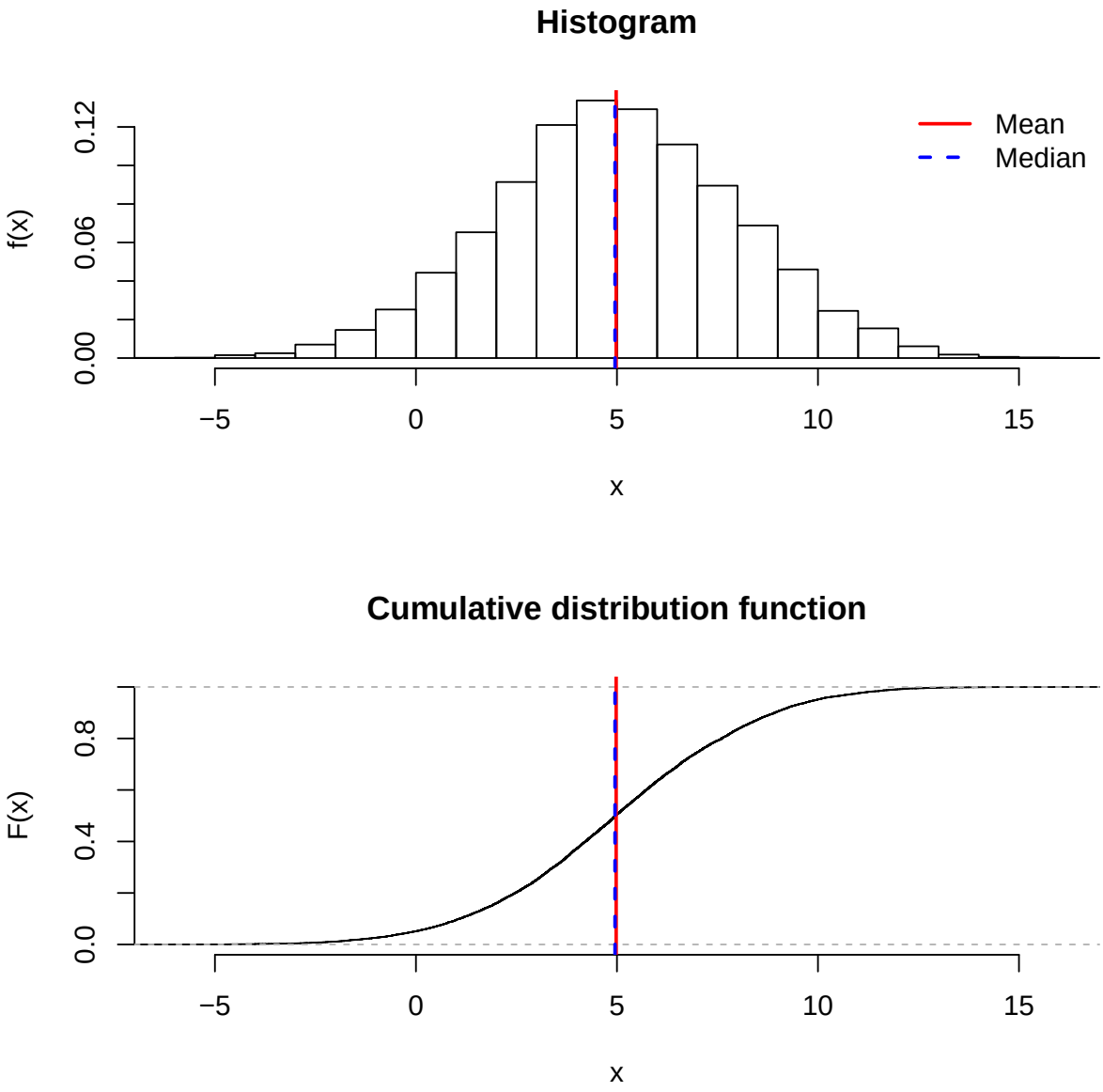

Figure 1: Contents of `figures/lab3_sample_plot1.pdf` after running `lab3_sample.R`. See text for discussion.

In a histogram, a group of observations is sorted into bins that usually have evenly-spaced edges, and the plot is then drawn as a series of rectangles with heights that are proportional to the number of observations in each bin. In general, then, bins with taller rectangles contain more observations than those with shorter rectangles.

To construct an empirical cumulative distribution function, the data are first sorted so that the data are in order from the smallest value to the largest. The plot is drawn as a series of "stairsteps" that connect $y = 0$ on the side of the plot corresponding to small values of x, to $y = 1$ on the right-hand side of the plot. In between, the value of the quantity plotted on the y-axis increases by $1/n$ at each x-value corresponding to one of the observations. On Figure 1, these "stairsteps" are too small to see individually, because each one corresponds to a change in y of $\Delta y = 1/n = 10^{-4}$. As a consequence of how they are constructed, empirical cumulative distribution functions always increase from small values of x to larger ones; that is, they are monotonically increasing.

The quantity plotted on the y-axis differs between histograms and empirical cumulative distribution functions. On a histogram, the y-axis represents the number of observations in each bin or (as in Figure 1) this number divided by the total number of observations. On an empirical cumulative distribution function, the y-axis represents the running total of the observations, going from small values of x to large ones, again divided by the total number of observations.

Measures of central tendency: the mean vs. the median

On Figure 1, the colored, vertical lines indicate the mean (specifically, the arithmetic mean) and median of the data. These quantities are both measures of the central tendency of a data set. The arithmetic mean is just the average of a group of numbers; the median represents the value that separates a sorted data set into two groups that contain the same number of observations.

In Figure 1, the lines representing the mean and the median lie on top of one another, indicating that these metrics are very close to each other. This result is expected for normally-distributed data sets that contain many observations, like the one shown in Figure 1. However, this result is often *not* true for data sets containing only a few values, or for those that are drawn from distributions that are not symmetrical.

Picking apart `lab3_sample.R`

Let's examine the R code in `lab3_sample.R`. The top few lines should look familiar; they tell us what the code does and delete any existing variables and figures from the workspace.

```
# lab3_sample.R
# Patrick Applegate, patrick.applegate@psu.edu
#
# Produces (pseudo-)random numbers from a normal distribution and plots a
# histogram and empirical cumulative density function using these numbers.

# Clear away any existing variables or figures.
rm(list = ls())
graphics.off()
```

The next block of code sets some values that we will use further down in the script. We set these values here so that they will be easy to find and change in the future. Also, it's good programming practice to avoid *magic numbers* (in this case, unnamed numerical constants). Note the comments that explain what each value represents.

```
# Set some values.
mu <- 5          # mean of normal distribution
sigma <- 3       # standard deviation of normal distribution
n.obs <- 10^ 4   # number of random values to generate
n.bins <- 25     # number of bins in the histogram
```

Next, we set the *seed* for random number generation. This command ensures that the code will give the same answers each time it is run, which is helpful for debugging. In general, any piece of code that depends on random number generation should include this command (though the value in parentheses can be changed).

```
# Set the seed for random sampling.
set.seed(1)
```

The following code block generates the random values. Look carefully at `help(rnorm)` to see what the arguments of the `rnorm()` command do. How do these arguments relate to the code block that starts, "`# Set some values`," above?

```
# Generate the random values.
data <- rnorm(n = n.obs, mean = mu, sd = sigma)
```

Next, we calculate the mean and the median of the values in `data`.

```
# Extract some relevant quantities from the random values.
mean.data <- mean(data)
median.data <- median(data)
```

The last, somewhat long, code block creates a `.pdf` file in a new directory called `figures` and plots the values in `data` in this file. The new commands in this code block include

- `hist` (plots histograms)
- `plot.ecdf` (plots empirical cumulative distribution functions)
- `abline` (overlays straight lines on an already-existing plot)
- `legend` (puts a legend on an already-existing plot)

```
# Make a plot.
dir.create('figures')
pdf('figures/lab3_sample_plot1.pdf')
par(mfrow = c(2, 1))

# Plot the histogram.
hist(data, breaks = n.bins, freq = FALSE, xaxs = 'i', main = 'Histogram',
     xlab = 'x', ylab = 'f(x)')

# Show the mean and the median on the histogram.
abline(v = mean.data, lty = 1, lwd = 2, col = 'red')
abline(v = median.data, lty = 2, lwd = 2, col = 'blue')

# Put a legend on the histogram.
legend('topright', legend = c('mean', 'median', '0.025 and 0.975'),
       lty = c(1, 2), lwd = 2, bty = 'n', col = c('red', 'blue'))

# Find the extents of the histogram's axes.
```

```
axis.lims <- par('usr')

# Now, plot the empirical cumulative distribution function.
plot.ecdf(data, xlim = axis.lims[1: 2], bty = 'n', xaxs = 'i',
          main = 'Cumulative distribution function', xlab = 'x', ylab = 'F(x)')

# Plot the mean and the median on the ecdf.
abline(v = mean.data, lty = 1, lwd = 2, col = 'red')
abline(v = median.data, lty = 2, lwd = 2, col = 'blue')
dev.off()
```

To see what the arguments of the `hist()` and `plot.ecdf()` commands mean, look at `help(hist)`, `help(plot.ecdf)`, and `help(par)`. Note that the `help(par)` page in particular is very long; it can be helpful to search within the help page for the names of particular arguments. To search within a help page, click on the Help window in RStudio, and (on a Mac) press and hold Command and then press F.

Exercise

Part 1. Adjust one of the variables near the top of `lab3_sample.R` (immediately after "`# Set some values`") and rerun the code. Repeat this process for each of these variables. How do each of these variables affect the figure that appears in `figures/lab3_sample_plot1.pdf`? You might want to change the file name in the `pdf()` command so that the new figures don't overwrite the old ones.

Part 2. Next, we'll experiment with other probability distributions. Change `n.obs` and `n.bins` back to their original values (see above). Create a copy of `lab3_sample.R` by saving the file under a different name (click on `File > Save As...` in RStudio).

Change your new script so that it generates random draws from each the following probability distributions. You'll need to make a new copy of the script each time you move to a new distribution.

- (continuous) uniform distribution
- lognormal distribution
- triangular distribution (you'll need the `triangle` package to work with this distribution; check Lesson 2 for a reminder about how to install and load packages)
- exponential distribution

To carry out this task for one of the distributions above, you'll need to

1. find out what the parameters of the new distribution are by reading the appropriate Wikipedia article
2. look at the help page associated with the new distribution in R using, for example, `help.search('uniform distribution')` – in particular, note which command on that help page generates random numbers, and what the arguments of that command are. How do these arguments match up with the parameters of the distribution from the Wikipedia article?
3. change the R script so that it sets the parameters of the new distribution under "`# Set some values`," and populates the vector `data` using the random number generation command for the new distribution.

Questions

1. How does changing the values of each of the variables near the top of `lab3_sample.R` adjust the appearance of the plots? Write a one-sentence description of the effects of each variable on how the plots look.

2. Make a table with the name of each distribution, the parameters of the distributions, and a short description of the plots associated with each histogram.
3. Which of the distributions place the mean and the median close to one another, and which have them far apart? Be sure to use a large value of `n.obs`, say `10^4`, for all the distributions when answering this question.