R Demonstration – Probability Distributions

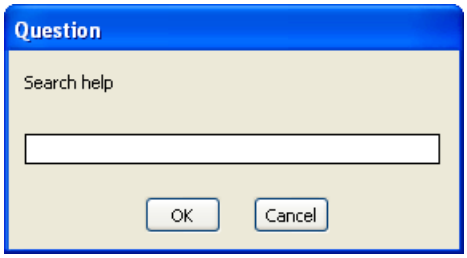
Objective: The purpose of this week's session is to demonstrate some of the important R functions related to random variables and probability distributions. In addition, basic R functionality related to getting help and saving/restoring workspaces will be introduced.

Part I. Getting Help in R

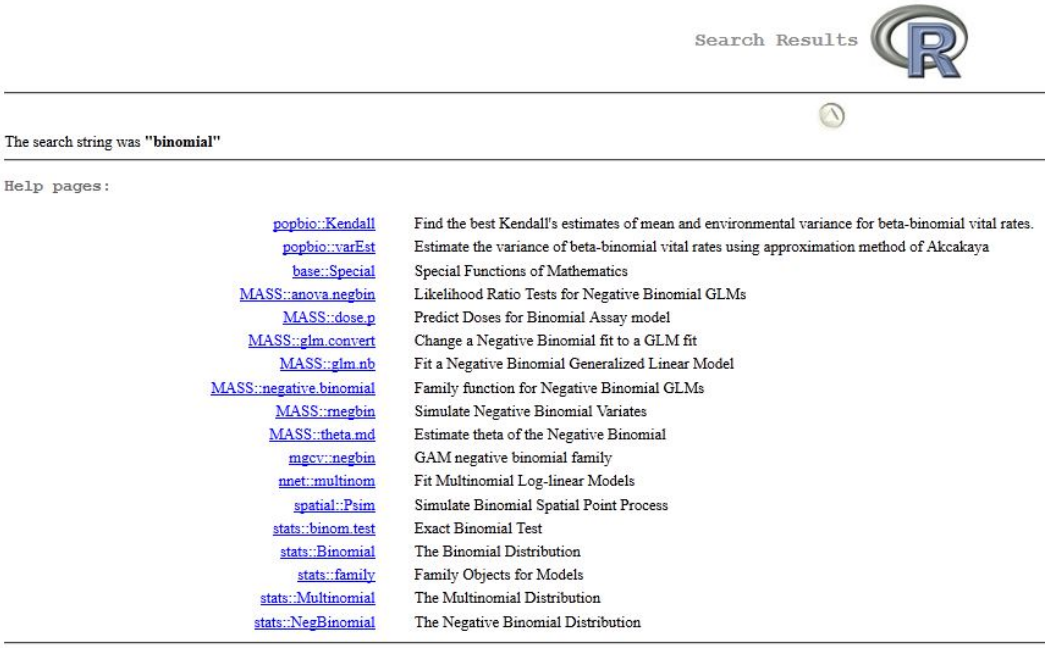
When you are learning R, one of the most important things you can do is to familiarize yourself with the built-in help system. While not quite as user-friendly as some commercial packages, R does allow you several ways to search for help.

Searching for Help by Keyword:

From the R menu bar, choose *Help→Search help*....This will bring up a dialog box that looks like the following:



Simply type in your keyword (or keywords) and click the OK button. For example, you might enter "binomial" to search for help on the binomial distribution. This would result in something similar to the following window being displayed in your default browser:



Note: on previous versions of R, this view will display in a window within the program and not in your browser.

Searching for Help on a Specific Function:
You can also search for help by the name of a specific R function. For example, in our previous keyword search, one of the results listed was "`Binomial(stats)`". This means that there is an R function named `Binomial` in the `stats` package (NOTE: we will cover R packages in a later session). To search for help on this function, choose *Help→R functions(text)...* from the menu bar and enter "Binomial" in the search field. Function-based searches are case-sensitive, so make sure you type the capital "B" at the beginning of the search string. R will then display the following help window in your default browser:

Binomial {stats}

The Binomial Distribution

Description

Density, distribution function, quantile function and random generation for the binomial distribution with parameters `size` and `prob`.

Usage

```
dbinom(x, size, prob, log = FALSE)
pbinom(q, size, prob, lower.tail = TRUE, log.p = FALSE)
qbinom(p, size, prob, lower.tail = TRUE, log.p = FALSE)
rbinom(n, size, prob)
```

Arguments

`x, q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of observations. If `length(n) > 1`, the length is taken to be the number required.
`size`	number of trials (zero or more).
`prob`	probability of success on each trial.
`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, $P[X > x]$.

Note: on previous versions of R, this view will display in a window within the program and not in your browser.

Note that the search returned a list of functions related to the binomial distribution: *`dbinom`*, *`pbinom`*, *`qbinom`*, and *`rbinom`*. In brief, *`dbinom`* gives the probability density function (i.e., `P(x)`) for the binomial distribution with `size` trials and `prob` probability of success. *`pbinom`* gives the cumulative probability distribution function (i.e., `P(X<=x)`) and *`qbinom`* gives the quantile function. Finally, the *`rbinom`* function returns a vector of `n` random values drawn from a binomial distribution with `size` trials and `prob` probability of success.

R provides a similar function of families for the other probability distributions we will explore in this lesson: the Poisson, uniform, normal, and lognormal distributions. For example, R has the *`dpois`*, *`ppois`*, *`qpois`*, and *`rpois`* for the Poisson distribution. The function that begins with *`d-`* is always the PDF, the one that begins with *`p-`* is the CDF, the one that begins with *`q-`* is the quantile function, and the one that begins with

r- is a function to draw random samples from the distribution. We will mostly use the density and random sampling functions (e.g., *dbinom* and *rbinom*) in today's lesson, but we will make use of the other functions in future exercises.

Searching for Help from the Command Line:
You can also perform a keyword search from the R command line by typing

```
> help.search(<search string>)
```

at the command prompt. Note that `<search string>` must be enclosed in quotation marks (for example, "binomial distribution"). Finally, you can search for a specific function from the command line by typing

```
> help(<function>)
```

at the command prompt, where `<function>` is the function name. For example, to get detailed help on the `rbinom` function you would type the following:

```
> help(rbinom)
```

We will use the `rbinom` function in the following section.

Note: depending on which version of R you are using, you may need to enclose the function name in quotation marks.

Part II. Exploring Probability Distributions

A Note on Functions in R:
In the most general sense, a function in R is a utility that performs some operation on the data that you provide as arguments to the function. For example, consider the *factorial* function, which has the format `factorial(x)`. This function has a single argument, `x`, which is the data for which you want to compute the factorial. To compute the factorial of 4, you would type the following at the command prompt:

```
> factorial(4)
```

R will respond by outputting the following line of text:

```
[1] 24
```

The value 24 is the factorial of 4, since 4*3*2*1 = 24 (NOTE: the [1] indicates that this is the first value returned by the function. Some functions will return more than a single value). What if you wanted to store this factorial in a variable so that you could use it in later computations? You could use the R operator `<-` to do this, as shown below:

```
> factorial_of_four <- factorial(4)
```

When you type this command, R won't respond with any output but it will store the value computed in the function into the new variable called `factorial_of_four`. To see this value, you can type the name of the new variable at the command prompt:

```
> factorial_of_four
[1] 24
```

Many functions in R take multiple arguments as input data. For example, the R function *`choose`* is used to calculate the number of combinations of a binomial random variable. It has the syntax `choose(n, k)`, where `n` is the number of trials and `k` is the number of successes. To calculate the number of combinations in which you can choose 3 successful outcomes from 6 trials, you would type the following at the command prompt:

```
> choose(6, 3)
[1] 20
```

Binomial Distributions:
A binomial random variable is defined by the relationship $X \sim Bin(n, p)$, where p is the probability of a successful outcome, n is the number of independent Bernoulli trials, and X is the number of successful outcomes (Gotelli & Ellison 2004, pp. 28-29).

Given this definition, the probability of obtaining X successes for such a binomial random variable is calculated with the following formula:

$$P(X) = \frac{n!}{X!(n-X)!} \cdot p^X (1-p)^{n-X}$$

In R, we can calculate this probability directly. For example, if we wanted to know the probability of obtaining 3 successes from 6 Bernoulli trials, each with a probability of 28/52, we could type:

```
> p <- 28/52
> choose(6,3)*(p^3)*((1-p)^(6-3))
[1] 0.3069854
```

Alternatively, we could use the *`dbinom`* function as follows:

```
> dbinom(3, 6, 28/52)
[1] 0.3069854
```

The first argument to the *`dbinom`* function is the number of successes (X = 3), the second argument is the number of trials (n = 6), and the third argument is the probability of success (p = 28/52).

A distribution can also be described by its **cumulative distribution function F(X),** which is the probability that the random variable *X* is less than or equal to a particular value *x*. We can use the function ***pbinom*** to obtain the probability of obtaining 3 successes from 6 Bernoulli trials as follows:

```
> limits <- c(0,1,2,3,4,5,6)
> results <-pbinom(limits,6,28/52)
> results[4]-results[3]
[1] 0.3069854
```

We can also plot the histogram of the cumulative distribution for this binomial trial:

```
> barplot(results, names.arg=limits,xlab="number of
successes", ylab="Cumulative probability")
```

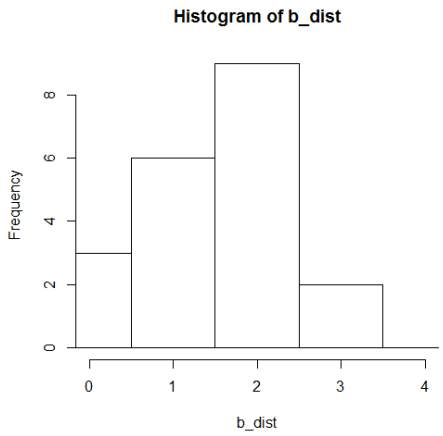
We can use R to emulate the process of drawing random samples from a binomial distribution. For example, in an exercise you were instructed to perform 20 trials in which you drew 4 cards from a randomly shuffled deck and counted cards with a value of A, J, Q and K as a successful capture. To simulate this in R, you would use the ***rbinom*** function as follows (NOTE: your results will be different since this generates a random sample):

```
> binom_distrib <- rbinom(20, 4, 16/52)
> binom_distrib
 [1] 2 1 3 0 2 1 2 0 0 2 2 1 2 2 1 2 1 1 2 3
```

Finally, since we have stored the results of the ***rbinom*** function in the variable `binom_distrib`, we can then use the ***hist*** function to plot a histogram of the distribution:

```
> hist_breaks<-c(-0.5,0.5,1.5,2.5,3.5,4.5)
> hist(binom_distrib, breaks=hist_breaks, xlim=c(0,4))
```

This will produce a plot similar to the following (again, your results will be slightly different because of random sampling):



The *hist* function has many options that allow you to control how the histogram is plotted. You should refer to the R help system (e.g., by typing `help(hist)` at the command prompt) for detailed information on this function.

Poisson Distributions:
A Poisson random variable is defined by the relationship $X \sim Poisson(\lambda)$, where X is the number of occurrences of an event recorded in a sample of fixed area or over a fixed time interval, and λ is the average number of occurrences in each sample (Gotelli & Ellison 2004, pp. 35-36). The probability of any observation x from a Poisson random variable is given by the following formula:

$$P(x) = \frac{\lambda^x}{x!} \cdot e^{-\lambda}$$

Again, we can use R to calculate this probability directly. For example, if we wanted to know the probability of finding 0 populations of a rare plant with a Poisson distribution and a λ of 0.03438 populations/sample, we could type

```
> lambda <- 0.03438
> (lambda^0/factorial(0))*exp(-lambda)
[1] 0.9662043
```

Alternatively, we could use the *dpois* function as follows:

```
> dpois(0, lambda)
[1] 0.9662043
```

The first argument to the *dpois* function is the number of successes ($x = 0$) and the second argument is the rate parameter ($\lambda = 0.03438$).

Finally, as we did for a binomial distribution, we can use R to emulate the process of drawing random samples from a Poisson distribution. For example, we saw in the lecture that the emission of alpha particles by polonium fits a Poisson distribution. We can simulate this process with the following R code:

```
> poisson_distrib <- rpois(2608, 10097/2608)
```

where n = 2608 is the number of random variables to be drawn and $\lambda = 10097/2608$ is the Poisson rate parameter. We can again use the R *hist* function to generate a histogram for the sampling distribution stored in the variable `poisson_distrib`:

```
> hist(poisson_distrib)
```

Normal Distributions:
A normal random variable is a continuous random variable defined as $X \sim N(\mu,\sigma)$, where μ is the mean, σ is the standard deviation, and, thus, σ^2 is the variance (Gotelli & Ellison 2004, pp. 47-48). Given μ and σ, the probability density function for the normal distribution is approximated by the formula:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{1}{2}\left(\frac{X-\mu}{\sigma}\right)^2}$$

If we know, for example, that the mean length of a sample of rock cod is 30 in. and the variance is 4 in., we can use the above formula to calculate the approximate probability that a certain rock cod is 31 in. long with the following R commands:

```
> mu <- 30
> variance <- 4
> sigma <- sqrt(variance)
> 1/(sqrt(2*pi)*sigma)*exp(-((31 - mu)^2/(2*sigma^2)))
[1] 0.1760327
```

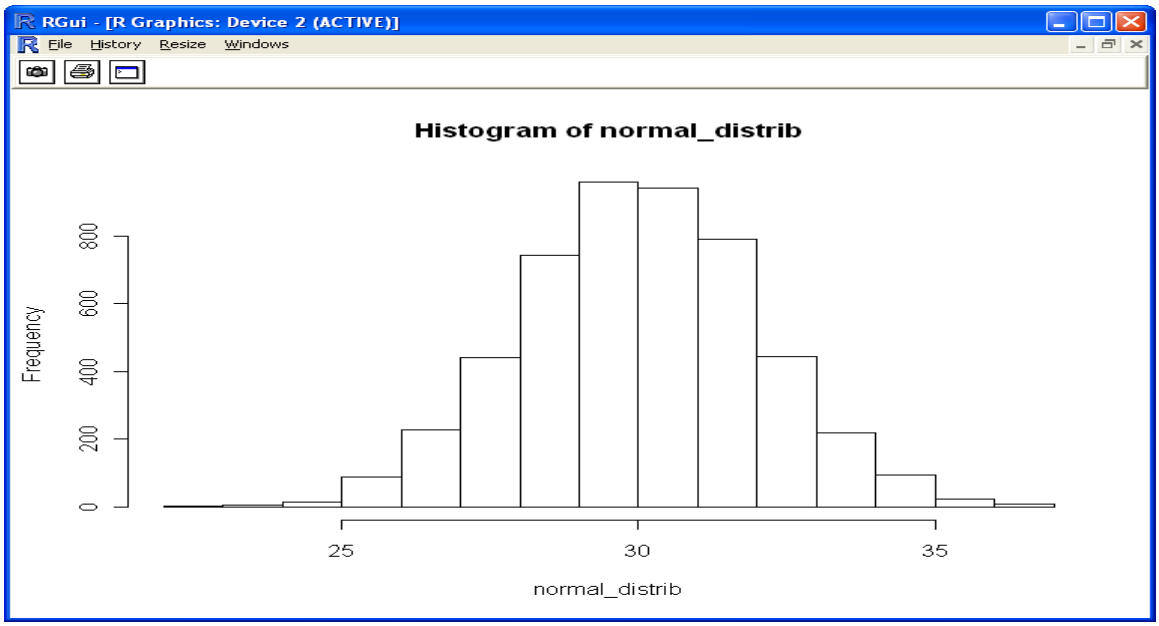
We can also calculate this probability using the *dnorm* function, as follows:

```
> dnorm(31, mean = mu, sd = sigma)
[1] 0.1760327
```

Finally, we can use the *rnorm* function to simulate drawing a number (say n = 5000) of random variables from a normal distribution with mean μ and standard deviation σ, and then use the *hist* function to plot a histogram of our distribution:

```
> normal_distrib <- rnorm(5000, mean=mu, sd=sigma)
> hist(normal_distrib)
```

Note the “bell curve” shape of the histogram, which is symmetric around the mean of 30:



Uniform Distributions:
A uniform distribution is used to describe equiprobable outcomes over a closed interval [*a*,*b*]. Because each outcome has the same probability, the R function ***runif*** is quite useful for creating a sample of random variables between some minimum and maximum value. For example, we could use the following R code to create a sample of 20 random variables in the interval between 1 and 10:

```
> runif(20,1,10)
 [1] 6.734666 9.334942 4.718385 9.982222 1.912516 1.865814
2.001767 8.011947 1.643035 8.711432
[11] 3.584781 7.679167 3.254184 1.273075 9.030344 6.860417
6.125817 4.510118 3.724532 7.548615
```

R also provides probability density, cumulative probability, and quantile functions for the uniform probability distribution (***dunif***, ***punif***, and ***qunif*** respectively).

Part III. Saving and Restoring R Workspaces

You can save all of the work you have done in a particular R session to a workspace file using the ***save.image*** function. For example, to save your current session to a workspace file named `My_Prob_Distrib.RData`, you would type the following:

```
> save.image(file="My_Prob_Distrib.RData")
```

NOTE: this command will only save the data objects you created in your R session, and not any of the output generated during your session. To save the output of the R console, you can use the *File→Save to file…* option from the menu bar.

The workspace file you saved with the `save.image` function will be automatically loaded when you start a new R session. Alternatively, you can use the `load` function.

Part IV. Exercise 1B

Please perform the following exercises to practice the R functions related to probability distributions you learned in this lesson. **Submit a single Word document with your answers and the R code you used to obtain your answers together with the material for exercise 1A. DUE: 09/03/2013.**

1. Generate and plot ***a variety*** of binomial distributions, varying first the probability of success and then the number of trials. Briefly explain what happens when you change each of these parameters.
2. Generate ***a series*** of Poisson distributions by increasing λ, the rate parameter. Briefly explain what happens when you change this parameter.
3. Observe and briefly explain what happens to the normal distribution when you change the mean, the variance, or both simultaneously.
4. Try to generate a lognormal distribution from any of the normal distributions you created in this exercise (NOTE: this is not as simple as it sounds). Briefly explain how you generated this distribution.