

AMS 206: Lab 1

Introduction to R

Objectives:

1. To demonstrate basic operations in R.
2. To learn learn relevant commands in R for probability distributions.

Instructions

As you read through, you will see things in typewriter font enclosed in curly braces. You are meant to actually type in those things (but not the braces themselves), and hit enter, so that you are following along and seeing how it all works.

Operating System Choices

Please note that these labs are written assuming you are using R from a unix machine in a CATS lab. As R is open-source, you can get either the source code or pre-compiled binaries for a variety of platforms online (<http://lib.stat.cmu.edu/R/CRAN/>) and you can use them on whatever machine you feel most comfortable with. However, technical support will only be provided for CATS unix versions. The official home page for R is <http://www.r-project.org/>, and you can find additional information there for all versions.

Starting R

If you are on a Mac or a Windows machine, you can usually start R by double clicking on the icon. On a unix machine, first open up a shell terminal window (using an OpenWindows environment, you can right-click on the background, go to the tools menu, and choose shell tool; using CDE, you can right click, go to tools, and choose terminal, or from the file manager, go to the file menu and choose open terminal). In the shell window, enter `{R}`. R will start up and you should get something remotely resembling:

```
R : Copyright 2002, The R Development Core Team
Version 1.6.0 (2002-10-01)
```

```
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
```

```
R is a collaborative project with many contributors.
Type 'contributors()' for more information.
```

```
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for a HTML browser interface to help.
Type 'q()' to quit R.
```

```
>
```

Note that the `>` is the prompt within R. This is where you enter commands. If you ever get the `+` prompt, this means that you didn't finish the command on the previous line, and R wants you to complete it (for example, you may have left off a closing parenthesis).

Quitting R

To quit R, the command is `q()`

Getting Help

R has on-line help available. Depending on your version, you may have a help menu, or you may need to enter `help.start()`. On some versions, you may need to start your web browser before entering that command.

R and S

R was developed as an open-source alternative to the commercial package S-Plus (which was also originally free). As I have used S-Plus at other places, if I forget and refer to "S", you should just assume I mean "R".

Simple Arithmetic

R will do all standard arithmetic operations, and it knows the correct order of operations. For example, enter `{1+2*3}` and you should get the correct answer of 7.

Assigning Objects

You may want to create a variable or assign the result of a function or operation to another object that you can keep around for future use. To do this, you use the assignment operator `<-` (which is a less-than symbol immediately followed by a hyphen). You can also use the notational shortcut of an equals sign (`=`) instead of the "arrow", or on older versions of R, the underscore (`_`).

1. For example, suppose you want `x` to have the value 6. You would enter `{x <- 6}`. It is usually helpful to use descriptive names, such as `{total.revenue <- 18.42}`.
2. To copy an object, we can just assign it to another name: `{expenses <- x}`
3. You can assign the result of an operation to an object:
`{net.profit <- total.revenue - expenses}`
4. To see the result, just enter the name of the object: `{net.profit}`

Note on naming of objects

Because the underscore has had the special meaning of assignment, *do not use an underscore in any object or function names*. You may use a period within a name. Also, the letters `c`, `q`, and `t` are all built-in commands, so R will be unhappy if you try to name an object as one of these letters (but it will let you do it, and then it may cause all sorts of problems later on).

Commands

The structure of functions in R is that you enter the function name followed by parentheses, putting arguments (if any) in the parentheses. If the function doesn't require any arguments (such as `q()` to quit), you still need to include the parentheses so that R knows that it is supposed to run the function. If you type in the name of a function without the parentheses, it will give you the code for the function. Also note that all entries in R are case-sensitive.

Removing Objects

As you create more and more objects, they will start to clutter up your workspace. To see what objects you have, use `ls()`. Objects that you no longer need should be discarded. Objects are removed with the function `rm()`, putting the name of the object in the parentheses, e.g., `{rm(x)}`. You can use `{ls()}` to verify that `x` has been deleted.

Graphics

To make plots, you need a graphical device (window). These days, most versions of R will automatically open one for you when you try to make your first plot. However, if you get an error message (such as “device not active”), then you will need to open a window manually. On most systems, this will be `X11`, although on some it could be `Motif`. On CATS, open the window in R by entering `{x11()}`.

Generating Random Numbers in R

The basis for most random number generation is the uniform distribution. To get a random value in $[0,1]$ with uniform probability over the interval, enter `{runif(1)}`, which will return one value. To get 100 of them, use `runif(100)`. Better yet, we will save a vector of 100 samples and take a look at its properties. Enter `{x <- runif(100)}`. Now look at its mean `{mean(x)}`, variance `{var(x)}` and histogram `{hist(x)}`. Are the mean and variance about where you expect them to be? Does the histogram show a roughly uniform distribution? These are random numbers, so they may not match quite what you expect. As you use a larger sample, they should look more like you expect. Try `{x <- runif(10000)}` and then look at the summary statistics and histogram and see that they should be pretty close to what theory predicts.

What if you want numbers uniformly distributed over a different interval, such as $[10,20]$? `runif()` can take two more arguments, a minimum and maximum for the interval. Thus `{runif(1,10,20)}` will produce a single random value between 10 and 20. You can try 10,000 of them and see that they are uniformly distributed between 10 and 20. `{hist(runif(10000,10,20))}`

To get random values from other distributions, the syntax is similar. All of the commands start with “r” and then there is some abbreviation for the name of the distribution. Try `{x <- rbinom(1000,100,.5)}` to get 1000 observations of a binomial with $n = 100$ and $p = \frac{1}{2}$, e.g., the result of flipping a fair coin 100 times. Then try `{mean(x)}`, `{var(x)}`, and `{hist(x)}`. Do the results look like you expect? With $p = .5$, the histogram should be symmetric (although it may look a little different depending on the particular bins). Now try a different binomial, such as `{x <- rbinom(1000,100,.8)}`. Check its summary and histogram. Notice how it is skewed, and see that its mean is where you expect it.

Other distributions are similar. For example, to get a single random Poisson with mean 10, you would use `rpois(1,10)`. Note that R parameterizes exponentials in terms of their rate, which

is $1/\text{mean}$. Thus to get a single random exponential with mean 10, you use `rexp(1,1/10)` (i.e., it has rate $1/10$). To convince yourself that this is the right way to get exponentials in R, try `{summary(rexp(1000,10))}` and `{summary(rexp(1000,.1))}` and see what the sample means are.

Note: if you ever forget what the arguments are for a command in R, you can just type the name of the command, and it will usually remind you. Try `{rbinom}`.

Working with the distributions

For a discrete distribution, such as a binomial or a Poisson, the probability density function gives the probability that the random variable will be equal to a particular value. R will give you these numbers with functions starting with “d”. For example, to see the probability that a binomial with $n = 5$ and $p = .25$ is equal to 3, enter `{dbinom(3,5,.25)}`. R will also give you cumulative probabilities, i.e., the probability that a random draw will be less than or equal to a particular value. For example, $P(X \leq 3)$ is `{pbinom(3,5,.25)}`. If you wanted to get the probability that it is larger than some value, say $P(X > 3)$, then you can use the complement rule: $P(X > 3) = 1 - P(X \leq 3)$. Since the binomial is discrete, note that $P(X > 3) = P(X \geq 4)$. The syntax is comparable for the Poisson distribution, `dpois()` and `ppois()`, although instead of giving x , n and p , you only need to give x and λ . For example, if X has a Poisson distribution with mean 10, then to get $P(X \leq 8)$, enter `{ppois(8,10)}`.

For a continuous distribution, such as the exponential or the normal, the probability density function is a function that gives relative likelihoods, and that when integrated over an interval gives the probability that the variable will be in that interval. `dexp()` gives the density function for the exponential distribution. Usually of more interest is the cumulative distribution function, the probability that the random variable is less than or equal to a particular value, i.e., $P(X \leq x)$. Just like with the discrete variables, we can use a “p” prefix in R. So to get the probability that a random exponential with mean 10 (rate $\frac{1}{10}$) is less than 5, enter `{pexp(5,.1)}`. For another example, what is the probability that a random exponential with mean 10 is between 4 and 5? To get the answer, enter `{pexp(5,.1) - pexp(4,.1)}`.

For the normal distribution, the analogous commands are `dnorm()` and `pnorm()`, and note that the normal is parameterized in terms of its mean and standard deviation (not variance). To plot a normal density (for example, one with mean 2 and variance 9 or std. dev. 3), you will need to evaluate it over a sequence, and then plot it against that sequence. First make the sequence with `{x <- seq(from=-7, to=11, by=.1)}`, then plot it with `{plot(x, dnorm(x, mean=2, sd=3))}`. To get a quantile (the inverse-CDF), the commands start with a “q”, i.e., when making a 95% confidence interval using the normal distribution, the cut-offs are at `qnorm(.025)` and `qnorm(.975)` (you don’t need to enter the mean and sd for the standard normal, as they are the default values).

Finishing Up

When you are done, it is good practice to remove any objects you have created that you won’t need again later (use `ls()` to see what you have, and `rm()` to remove anything unnecessary, like `net.profit`). To quit from R, enter `{q()}`. This will also automatically close any graphics windows. When quitting, it will ask you if you want to **save workspace image?** `[y/n/c]`. If you want any of your created objects to be there the next time you run R, you need to enter `y` here, otherwise it won’t save anything you created during this session. Of course, if you don’t want to save what you did, then choosing `n` is an easy way to delete whatever temporary things you made without having to `rm` them individually.