Math 365 Introduction to R and Stochastic Processes

The objective of these exercises is to familiarize you with R and RStudio by exploring a few examples of stochastic processes. If possible, you should install RStudio on your own computer (see the links on the course webpage). If you have a laptop, please bring it regularly to class; if you don't, please come see me and we'll try to borrow one for you. Running the R simulations yourself is more fun and will improve your coding skills.

In the Console window of RStudio (lower left corner), try entering a few basic commands like `1+1`, `2^4`, `cos(pi)`, `exp(1)`, `log(exp(3))`, `factorial(4)`, and `sqrt(16)`. Always feel free to ask me if you have trouble with anything, whether syntax errors or coding issues or other problems. You can also look up commands in the help window in the lower right corner of RStudio, which provides information and examples illustrating proper syntax.

1 Vectors in R

We frequently work with lists of numbers, so we need a few basic commands for generating vectors or sequences. Successive numbers can be generated via `a:b`. Try a few examples like `1:10`, `-5:5`, and `2.5:7.5`.

If you want a pattern that skips by a number other than 1, use `seq`. See what `seq(1,9)`, `seq(1,9,2)`, `seq(9,0,-3)`, and `seq(1,3,0.2)` do. To assign the sequence of numbers to a variable name, use a left-pointing arrow: `x <- seq(1,10)`. To access a component of the vector, give the index in square brackets: `x[4]` or `x[3:6]`.

To combine (concatenate) different numbers or vectors together, use `c`: `x <- c(1,1,2,3,5,8)` and `x <- c(x,13,21)`. To see what `x` currently equals, look at the Environment tab in the upper right corner of RStudio, which also indicates the size of the vector and type of entries. The History tab next to it shows recent commands and allows you to re-enter them in the Console by double-clicking on them.

You can do basic operations on vectors like `sum(x)`, `mean(x)`, `sd(x)`, and `length(x)`. To practice writing a script, open up a new R script by clicking the button in the upper left corner of the RStudio window. Type in commands that define a vector `x` and then find its mean, sum, and length. To evaluate the commands, either put the cursor on the first line of the script and hit Run for each line, or highlight all of the commands you want to run and then hit Run.

Arithmetic on vectors is done component-wise. For example, `seq(10,1,-1)/(1:10)` creates a new vector whose first component is 10/1, second component is 9/2, then 8/3, and so on.

2 Coin flips script

In a new R script, type and then run the following code:

```
# Coin flips
n <- 1000 # Number of coin flips
coinflips <- sample(0:1,n,replace=TRUE)
coinflips
heads <- cumsum(coinflips)
prop <- heads/(1:n)
plot(1:n,prop)
```

These commands generate a random sample of 1000 coins flips (really, 0s and 1s), calculates the cumulative sum to get a running count of how many heads (1s) have been flipped at each point, then calculates the proportion of heads after each flip and plots it. The plot can be improved with some labeling and a line marking the expected proportion of heads over time:

```
plot(1:n,prop,type="l",xlab="Number of coin flips",
ylab="Running average",main="Proportion heads")
abline(h=0.5)
```

3 Random walk demo

R will allow us to run simulations of the various stochastic processes we will be studying. Here is a first peek at such a simulation, involving a `for` loop, which is a very handy construct.

```
# Bernoulli process random walk (heads move to right, tails move to left)
n <- 100 # Number of steps
coinflips <- sample(c(-1,1),n,replace=TRUE)
walk <- cumsum(coinflips) # position after each flip
for (k in 1:n) {
plot(1:k, walk[1:k],xlim=c(1,n),
ylim=c(-n/5,n/5),type="l",xlab="Step",ylab="Position")
Sys.sleep(0.1) # short pause so can watch random walk step by step
}
```

Run a longer simulation to get a feel for long-term behavior of the random walk:

```
n <- 100000
coinflips <- sample(c(-1,1),n,replace=TRUE)
walk <- cumsum(coinflips)
plot(1:n, walk,xlim=c(1,n),type="l",xlab="Step",ylab="Position")
```

To see the distribution of positions during the random walk, look at the histogram:

```
hist(walk, main=paste(n, "step random walk with mean", round(mean(walk),3)))
```

You don't need to submit anything from today's R work. Do let me know if you have any questions about using R, any of the code we did today, or installing R on your own computer. For a more complete introduction to R, see `http://cran.us.r-project.org/doc/manuals/R-intro.pdf`.