R for 2.1-2.3

A.S. Wagaman, CC BY-NC

R for 2.1 - Conditional Probability

The first R code we encounter in Chapter 2 relates to a script to accompany Example 2.4.

SCRIPT - ConditionalDice.R - pg. 37

```
### ConditionalDice.R
###
### Roll 2 dice.
### P(First die 2 | Sum is 7)

n <- 60000
ctr <- 0
simlist <- replicate(n, 0) ## Initialize list with 0's
while (ctr < n)
{ #start of brackets for while loop
    trial <- sample(1:6, 2, replace = TRUE) ## Roll 2 dice
    if (sum(trial) == 7) { #start of brackets for if statement
                           ### Check if sum is 7
                           ### If not, skip through and roll again
                           ### If 7, check if first die is a 2
    success <- if (trial[1] == 2) 1 else 0
    ctr <- ctr + 1
    simlist[ctr] <- success
    ### simlist records successes and failures only for dice rolls that sum to 7
    } #end of brackets for if statement
} #end of brackets for while loop

### Simulated result
mean(simlist)
```

```
## [1] 0.1668167
```

This script introduces a different type of loop - a *while* loop. The idea is that the conditional probability desired is conditional on rolling a sum of 7. Thus, if we set a fixed number of die rolls using a for loop, only a fraction would be useful to us where a 7 sum is rolled. A while loop allows us to accumulate a set of trials were only sums of 7s were rolled. We basically only "count" the trial if it had a sum of 7.

Note that this is accomplished with an if statement inside the while loop. Due to the fact that three lines of code need executed if the sum is 7, these commands are contained within brackets in the while loop. I've added comments so you can see where this happens. Note that the indentation within the code is designed to help you see this as well. Indenting and commenting are conventions that you should work on using in your code. Code that is written without comments can be tricky to re-parse months later if you revisit an old project, for example. Even minor comments about what a loop is for or what the overall function is designed to do can be very useful.

While loops require a bit more savvy coding than for loops, because if you fail to iterate the counter (ctr here) correctly, you can get the computer stuck. Luckily, there are ways to stop the system if you make an error. If this happens to you, please ask for assistance! There are often alternative ways to code that avoid

while loops entirely. For our class, we use a small enough number of repetitions that the code should run fairly quickly. If an individual code chunk is running for more than a minute or two, look for the STOP button in the Console window and press it to stop the code, then ask for assistance.

R for 2.2 - New Information Changes the Sample Space

```
#NA
```

R for 2.3 - Finding P(A and B)

This script accompanies Example 2.9 to verify the theoretical result obtained from using the tree diagram.

SCRIPT - Blackjack.R - pg. 45

```
### Blackjack.R
### What's the probability of Blackjack?

n <- 50000   ### Number of iterations
simlist <- replicate(n, 0) ## Initialize list with 0's

for (i in 1:n){ #start of for loop
    trial <- sample(1:52, 2, replace = FALSE)
    ## Let Ace <--> 1, 2, 3, 4
    ## Let Ten card <--> 37, 38, ... , 51, 52
    success <- if (trial[1] <= 4 && trial[2] >= 37 ||
        trial[1] >= 37 && trial[2] <= 4) 1 else 0
    simlist[i] <- success
} #end of for loop

### Simulated result
mean(simlist)
```

```
## [1] 0.04808
```

The main new concept in this script is that you have to be creative about how the card values are assigned. Based on a 52 card deck, certain values (cards 1-4) are treated as aces, and other values (cards 37-52) are treated as 10s or face cards. The ifelse statement contains an "and" (the double ampersand) to check for blackjack as well as an "or" (the double bars) since we don't care if the ace or ten point card is dealt to us first. Alternative sampling mechanisms using the prob option to set probabilities of drawing an ace versus a 10 point card versus another card are also possible.

If you wanted the result here to be reproducible, you would set a seed before the for loop starts. What would happen if you set a seed inside (at the top of) the for loop?

ANSWER

Supplemental R Commands

In section 2.1, some tables were entered in the text. We can enter tables like this in to R, though you need to pay attention to how you do it. Here, I enter the table from page 34 into R, and follow-up with an additional example. The main commands used are just c(), which concatenates its contents together and *rep*, which

is useful for repeating values and characters. *rep* has many options which you can explore using the help function.

```
#Bear/bee table in R - pg. 34
Sex<-c(rep("F", 36), rep("M", 24)) #enter the marginal counts of one variable
#this creates a vector of 36 Fs followed by 24 Ms
BearBee<-c(rep("Bear", 27), rep("Bee", 9), rep("Bear", 10), rep("Bee", 14)) #enter the conditional brea
tally(~ Sex + BearBee, margins = TRUE)
```

```
##        BearBee
## Sex     Bear Bee Total
##   F       27   9    36
##   M       10  14    24
##   Total   37  23    60
```

Here is another example using survey data about death penalty and marijuana legalization beliefs. Note that although both these examples are 2x2, you can enter larger tables following the same logic.

```
# Death Penalty vs. Marijuana Belief Survey Data
DP<-c(rep("Favor", 532), rep("Oppose", 275)) #enter the marginal of one variable
MarBelief<-c(rep("Legal", 191), rep("NotLegal", 341), rep("Legal", 104), rep("NotLegal", 171)) #enter t
tally(~ DP + MarBelief, margins = TRUE)
```

```
##         MarBelief
## DP       Legal NotLegal Total
##   Favor    191      341   532
##   Oppose   104      171   275
##   Total    295      512   807
```

If you turn off margins = TRUE, you just won't have the total row/column displayed.

Later, we will see some joint probability distributions in table form. Those can also be entered into a table, and you can change the option to display proportions instead of counts to make them valid probability distributions. You just have to work out what the corresponding "counts" would be out of some hypothetical number (1000, 10000, etc. to get whole numbers for the counts) in order to enter values into R using this method.

```
tally(~ DP + MarBelief, margins = TRUE, format = "proportion")
```

```
##         MarBelief
## DP           Legal  NotLegal     Total
##   Favor  0.2366791 0.4225527 0.6592317
##   Oppose 0.1288724 0.2118959 0.3407683
##   Total  0.3655514 0.6344486 1.0000000
```

Textbook Citation

Dobrow, R. P. (2013). Probability: with applications and R. John Wiley & Sons.