DapR1: Notes on the Live R Session, Week 8

This week, we will be picking up where we left off the last time. We'll continue working with probability using operators. We'll also talk about writing a clean report using Markdown, including learning about in-line R coding and notation.

First, let's load the tidyverse and create our sample space. We'll also recreate our sample data from last week. Because we used the *set.seed()* function last week, we can use the same seed value to make R generate the same data we used last week.

```
library(tidyverse)
skittles <- c('red', 'green', 'yellow', 'purple', 'orange')

set.seed(2210)
s5 <- tibble(Obs = sample(skittles, 5, replace = T))
s100 <- tibble(Obs = sample(skittles, 100, replace = T))
```

Although we're working with character data in this example, you can also create sample data with numeric values. Consider an example where you want to calculate the probability of rolling an even number on a die. You could use *seq()* function, which allows you to generate sequences of numbers, to create your sample space and your events of interest:

```
dSp <- seq(1, 6)
dEv <- seq(2, 6, by = 2)
```

Joint Probability

With joint probability, we're interested in looking at the probability of multiple outcomes. For example, if we were to select a skittle from the bag, what's the likelihood we would select a yellow **OR** a purple?

Because these events are mutually exclusive, we would expect their joint probability to be the sum of their individual probabilities.

$p(yellow \cup purple) = .20 + .20 = .40$

Here, we'll review two ways you can specify multiple events of interest and use these methods to compute joint probability in your sample.

The AND and OR operators

Two operators that may be used for specifying multiple conditions are **&** and **|**. As you might expect, **&** refers to *'and'*. It allows you to specify values that meet *all* given conditions. The **|** operator stands for *'or'* and allows you to specify values that meet *any* of the given conditions.

```
successes <- s100 == 'yellow'|s100 == 'purple'
failures <- s100 != 'yellow' & s100 != 'purple'

sum(successes)/(sum(successes) + sum(failures))
```

```
## [1] 0.4
```

%in% or is.element()

You can get the same results using the **%in%** operator. This operator can be used to determine whether a given element (or values from a vector of elements) is found in a dataset or vector. Also, note the use of another operator. The **!** operator stands for *'not'* and allows you to select values that do not meet a specified condition.

```
events <- c('purple', 'yellow')
successes <- s100$Obs %in% events
failures <- !s100$Obs %in% events

sum(successes)/(sum(successes) + sum(failures))
```

```
## [1] 0.4
```

You can also use the **is.element()** function in the same way. *is.element(x, y)* is identical to *x %in% y*

```
is.element(s100$Obs, events)
```

```
##   [1] FALSE  TRUE  TRUE FALSE FALSE  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE
##  [13] FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE  TRUE  TRUE FALSE  TRUE FALSE
##  [25] FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE
##  [37]  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE
##  [49] FALSE FALSE FALSE  TRUE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE
##  [61] FALSE  TRUE FALSE FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE
##  [73]  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
##  [85] FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE
##  [97] FALSE  TRUE FALSE FALSE
```

```
successes <- is.element(s100$Obs, events)
failures <- !is.element(s100$Obs, events)
sum(successes)/(sum(successes) + sum(failures))
```

```
## [1] 0.4
```

Now, let's compute the probability of selecting a yellow **AND** a purple skittle in two selections.

To do this, we would use we use the following formula:

$p(yellow \cap purple) = .20 * .20 = .04$

Let's check to see if our data follow the expected result. To do this, we'll need to add a new column that represents the second skittle selection:

```
s100$Obs2 <- sample(skittles, 100, replace = T)
head(s100)
```

```
## # A tibble: 6 x 2
##   Obs    Obs2
##   <chr>  <chr>
## 1 green  orange
## 2 yellow orange
## 3 yellow orange
## 4 orange purple
## 5 red    purple
## 6 yellow purple
```

```r
successes <- (s100$Obs=='purple'&s100$Obs2=='yellow')|(s100$Obs=='yellow'&s100$Obs2=='purple')
failures <- !successes

sum(successes)/(sum(successes) + sum(failures))
```

```
## [1] 0.05
```

You can also look at the proportion table for multiple events:

```r
s100 %>%
  table %>%
  prop.table()
```

```
##         Obs2
## Obs      green orange purple  red yellow
##   green   0.01   0.05   0.01 0.05   0.06
##   orange  0.04   0.05   0.06 0.02   0.03
##   purple  0.03   0.08   0.06 0.02   0.02
##   red     0.06   0.04   0.05 0.04   0.03
##   yellow  0.06   0.06   0.03 0.03   0.01
```

Note that you can sum the appropriate locations in the proportion table to get the corresponding probability.

```r
pTab <- s100 %>%
  table() %>%
  prop.table()

pTab['purple', 'yellow'] + pTab['yellow', 'purple']
```

```
## [1] 0.05
```

You can also compute the proportions for each row or each column using the *margin* argument:

```r
# margin = 1: proportion within each row
pTab <- s100 %>%
  table %>%
  prop.table(margin = 1)

pTab
```

```
##         Obs2
## Obs           green     orange     purple        red     yellow
##   green  0.05555556 0.27777778 0.05555556 0.27777778 0.33333333
##   orange 0.20000000 0.25000000 0.30000000 0.10000000 0.15000000
##   purple 0.14285714 0.38095238 0.28571429 0.09523810 0.09523810
##   red    0.27272727 0.18181818 0.22727273 0.18181818 0.13636364
##   yellow 0.31578947 0.31578947 0.15789474 0.15789474 0.05263158
```

```
sum(pTab[1,])
```

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

```
# margin = 2: proportion within each column
pTab <- s100 %>%
  table %>%
  prop.table(margin = 2)

pTab
```

```
##         Obs2
## Obs           green     orange     purple        red     yellow
##   green  0.05000000 0.17857143 0.04761905 0.31250000 0.40000000
##   orange 0.20000000 0.17857143 0.28571429 0.12500000 0.20000000
##   purple 0.15000000 0.28571429 0.28571429 0.12500000 0.13333333
##   red    0.30000000 0.14285714 0.23809524 0.25000000 0.20000000
##   yellow 0.30000000 0.21428571 0.14285714 0.18750000 0.06666667
```

```
sum(pTab[,1])
```

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

However, the table outputs in R are not appropriate for a formal report. If you want to include a table in your report, you should use the kbl() function to produce a formal table to display your data:

```
library(kableExtra)

pTab %>%
  kbl(booktabs = T, digits = 2, caption = 'Skittles Experiment Data') %>%
  kable_styling(latex_options = c('hold_position', 'striped')) %>%
  column_spec(1, bold = T) %>%
  row_spec(0, bold = T)
```

Table 1: Skittles Experiment Data

	green	**orange**	**purple**	**red**	**yellow**
green	0.05	0.18	0.05	0.31	0.40
orange	0.20	0.18	0.29	0.12	0.20
purple	0.15	0.29	0.29	0.12	0.13
red	0.30	0.14	0.24	0.25	0.20
yellow	0.30	0.21	0.14	0.19	0.07

Conditional Probability Data & Write-Up

Now let's import some data that we can use to demonstrate conditional probability. We'll also write up our results neatly and talk about in-line R coding.

Imagine that we want to investigate the relationship between med school acceptance and academic performance. Specifically we will look at the likelihood of acceptance (*yes* or *no*) is related to having higher marks (e.g., above the 75th percentile or greater in our sample; *high* or…less high. We'll just say *low* for simplicity's sake).

```
dat <- read.csv('https://uoepsy.github.io/data/MedGPA.csv')
summary(dat)
```

```
##     Accept            Acceptance          Sex                 BCPM      
##  Length:55          Min.   :0.0000   Length:55          Min.   :2.410  
##  Class :character   1st Qu.:0.0000   Class :character   1st Qu.:3.260  
##  Mode  :character   Median :1.0000   Mode  :character   Median :3.530  
##                     Mean   :0.5455                      Mean   :3.501  
##                     3rd Qu.:1.0000                      3rd Qu.:3.755  
##                     Max.   :1.0000                      Max.   :4.000  
##                                                                         
##       GPA              VR               PS               WS        
##  Min.   :2.720   Min.   : 6.000   Min.   : 5.000   Min.   : 4.000  
##  1st Qu.:3.375   1st Qu.: 8.000   1st Qu.: 9.000   1st Qu.: 6.000  
##  Median :3.580   Median :10.000   Median :10.000   Median : 8.000  
##  Mean   :3.553   Mean   : 9.764   Mean   : 9.709   Mean   : 7.148  
##  3rd Qu.:3.770   3rd Qu.:11.000   3rd Qu.:10.500   3rd Qu.: 8.000  
##  Max.   :3.970   Max.   :13.000   Max.   :14.000   Max.   :10.000  
##                                                    NA's   :1       
##        BS              MCAT            Apps       
##  Min.   : 6.000   Min.   :18.00   Min.   : 1.000  
##  1st Qu.: 9.000   1st Qu.:34.00   1st Qu.: 5.000  
##  Median :10.000   Median :36.00   Median : 7.000  
##  Mean   : 9.782   Mean   :36.27   Mean   : 8.364  
##  3rd Qu.:11.000   3rd Qu.:39.00   3rd Qu.:11.000  
##  Max.   :14.000   Max.   :48.00   Max.   :24.000  
## 
```

```
dat$acceptChar <- ifelse(dat$Acceptance == '0', 'Rejected', 'Accepted')
dat$GPAsplit <- ifelse(dat$GPA >= quantile(dat$GPA)['75%'], 'High', 'Low')

pTab <- table(dat$GPAsplit, dat$acceptChar) %>%
  prop.table()

pTab
```

```
## 
##          Accepted   Rejected
##   High 0.25454545 0.01818182
##   Low  0.29090909 0.43636364
```

Remember from yesterday's lecture that we can calculate conditional probability using the following formula:

$$p(A|B) = \frac{p(A \cap B)}{p(B)}$$

Let's compare the probability of being accepted given that marks are high with the probability given that marks are low. In other words, we are comparing an outcome of event A across different levels of event B.

If these events were unrelated, we would expect the probability of event A to be generally consistent across both levels of B. To do this, we can plug our variables into the formula above:

$p(Accepted|High) = \frac{p(Accepted \cap High)}{p(High)}$

$p(Accepted|Low) = \frac{p(Accepted \cap Low)}{p(Low)}$

Remember, we can pull specific values from the probability table we've created:

```
pTab['High', 'Accepted']/(sum(pTab['High',]))
```

```
## [1] 0.9333333
```

```
pTab['Low', 'Accepted']/(sum(pTab['Low',]))
```

```
## [1] 0.4
```

Here, we see that the values are quite different, which indicates that the probability of acceptance is likely to be related to overall marks (shocking!). Now let's show an example of how to write up these results.

EXAMPLE WRITE-UP

In this experiment, we collected data on medical school acceptance rates and school performance from 55 participants. Specifically, we investigated whether acceptance to medical school was related to overall performance as measured by GPA (High/Low). We calculated the proportion of participants who fell into each category (see Table 2).

Table 2: Medical School Acceptance by GPA

	Accepted	Rejected
High	0.25	0.02
Low	0.29	0.44

We computed the probability of acceptance at both levels of GPA. The probability of acceptance given a high GPA was 0.93. The probability of acceptance given a low GPA was only 0.4. This indicates that the likelihood of being accepted changes at different levels of GPA.

The differences in the probability of acceptance across levels of GPA indicates a possible relationship between school performance and acceptance to medical school.