Case Study in Probability

Lt Col Kriss Pruitt Professor Bradley Warner

18 May, 2020

Objectives

1) Use R to simulate a probabilistic model.

2) Use basic counting methods.

Introduction to Block 2 - Probability Models

In this second block of material we will focus on probability models. We will take two approaches, one is mathematical and the other is computational. The mathematical model allow us insight and ability to understand the process. Simulation has a much greater ability to generalize but can be time intensive to run and often requires the writing of custom functions.

Probability models

Probability models are an important tool for data analysts. They are used to explain variation in outcomes that cannot be explained by other variables. We will use these ideas in block 3 to help us make decisions about our statistical models.

Often probability models are used to answer a question of the form "What is the chance that .....?"

For this block we will focus just on probability models. To apply a probability model we will need to

1. Select the outcome of interest and its possible values
2. Have probability values for the outcomes which may include **parameters** that determine the probabilities
3. Understand the assumptions behind the model

Case study

There is a famouse example of a probability question that we will attack in this case study. The question we want to answer is "In a room of n people what is the chance that at least two people have the same birthday?"

> **Exercise**:
> The typical classroom at USAFA has 18 students in it. What do you think the chance that at least two students have the same birthday?[1]

[1]The answer is around 34.7%

Break down the question

The first action we should take is to understand what is being asked.

1. What does it mean to have the same birthday?
2. What about leap years?
3. What about the frequency of births? Are some days less likely than others?

  Exercise:
  Discuss these questions and others that you think are relevant.[2]

The best first step is to make a simple model, often these are the only ones that will have a mathematical solution. For our problem this means we answer the above questions

1. We don't care about the year, only the day and month. Thus two people born on May 16th are a match.
2. We will ignore leap years.
3. We will assume that a person has equal probability of being born on any of the 365 days of the year.
4. At least two means we could have multiple matches on the same day or several different days where multiple people have matching birthdays.

Simulate

Now that have an idea about the structure of the problem, we next need to think about how we would simulate a single classroom. We have 18 students in the classroom and they all could have any of the 365 days of the year as a birthday. What we need to do is sample birthdays for each of the 18 students. But how do we code the days of the year?

An easy solution is to just label the days from 1 to 365. The function `seq()` does this for us.

```
days <- seq(1,365)
```

Next we need to pick one of the of the days using the sample function. Note that we set the seed to get repeatable results, this is not required.

```
set.seed(2022)
sample(days,1)
```

```
## [1] 228
```

The first person was born on the 228th day of the year.

Since `R` is works on vectors, we don't have to write a loop to select 18 days, we just have `sample()` do it for us.

```
class <- sample(days,size=18,replace = TRUE)
class
```

```
##  [1] 206 311 331 196 262 191 206 123 233 270 248   7 349 112   1 307 288 354
```

Notice we have at least one match, although it is difficult to look at this list and see the match. Let's sort them to make it easier for us to see.

[2] Another question may be What does it mean at least two people have matching birthdays?

```r
sort(class)
```

```
##  [1]   1   7 112 123 191 196 206 206 233 248 262 270 288 307 311 331 349 354
```

The next step is to find a way in `R` for the code to detect that there is a match.

> **Excercise**:
> What idea(s) can we use to determine if a match exists?

We could sort the data and look at differences in sequential values and then check if the set of differences contains a zero. This seems to be computationally expensive. Instead we will use the function `unique()` which gives a vector of unique values in an object. The function `length()` gives the number of elements in the vector.

```r
length(unique(class))
```

```
## [1] 17
```

Since we only have 17 unique values in a vector of size 18, we have a match. Now let's put this all together to generate another classroom of size 18.

```r
length(unique(sample(days,size=18,replace = TRUE)))
```

```
## [1] 16
```

The next problem that needs to solved is how to repeat the classrooms and keep track of those that have a match. There are several functions we could use to include `replicate()` but we will use `do()` from the `mosaic` package because it returns a data frame so we can use `tidyverse` verbs to wrangle the data.

The `do()` function allows us to repeat an operation many times. The following template

```
do(n) * {stuff to do}              # pseudo-code
```

where {stuff to do} is typically a single `R` command, but may be something more complicated.

Load the libraries.

```r
library(mosaic)
library(tidyverse)
```

```r
do(5)*length(unique(sample(days,size=18,replace = TRUE)))
```

```
##   length
## 1     18
## 2     17
## 3     17
## 4     17
## 5     18
```

Let's repeat for a larger number of simulated classroom, remember you should be asking yourself:

What do I want `R` to do?
What does `R` need to do this?

```
(do(1000)*length(unique(sample(days,size=18,replace = TRUE)))) %>%
  mutate(match=if_else(length==18,0,1)) %>%
  summarise(prob=mean(match))
```

```
##   prob
## 1 0.36
```

How many classrooms to we need to simulate to get an accurate estimate of the probability of a match? That is a statistical modeling question and it depends on how much variability we can accept. We will discuss these ideas later in the semester. For now, you can run the code multiple times and see how the estimate varies. If computational power is cheap you can increase the number of simulations.

```
(do(10000)*length(unique(sample(days,size=18,replace = TRUE)))) %>%
  mutate(match=if_else(length==18,0,1)) %>%
  summarise(prob=mean(match))
```

```
##     prob
## 1 0.3442
```

Mathematical solution

```
Births %>%
group_by(day_of_year) %>%
summarise(n=sum(births))
```

```
## # A tibble: 366 x 2
##    day_of_year      n
##          <int>  <int>
##  1           1 160369
##  2           2 169896
##  3           3 180036
##  4           4 182854
##  5           5 184145
##  6           6 186726
##  7           7 188277
##  8           8 185186
##  9           9 181511
## 10          10 183668
## # ... with 356 more rows
```

```
Births %>%
group_by(year) %>%
summarise(n=n())
```

```
## # A tibble: 20 x 2
##     year     n
##    <int> <int>
##  1  1969   365
##  2  1970   365
## 3  1971   365
## 4  1972   366
## 5  1973   365
## 6  1974   365
## 7  1975   365
## 8  1976   366
## 9  1977   365
## 10 1978   365
## 11 1979   365
## 12 1980   366
## 13 1981   365
## 14 1982   365
## 15 1983   365
## 16 1984   366
## 17 1985   365
## 18 1986   365
## 19 1987   365
## 20 1988   366
```

File creation information

- File creation date: 2020-05-18
- Windows version: Windows 10 x64 (build 17763)
- R version 3.6.3 (2020-02-29)
- `mosaic` package version: 1.6.0
- `tidyverse` package version: 1.3.0