Statistics & Probability
Statistical Programming with R
Class 2 – Discrete Random Variables

- Today we are going to use **R** to simulate the results of a dice-rolling experiment. This is a simple example of working with a "manual" discrete distribution (finite number of outcomes possible); as opposed to one of the "classical" discrete distributions (Poisson, hypergeometric or binomial were the ones we looked at in detail in lectures).

- Later in the class we will look at calculating probabilities from the "classical" distributions using **R**.

- You may wish to look back at the Lab 1 worksheet for pointers if you have any problems with the **R** code.

1 Rolling Dice

- Suppose you roll one red die and one green die and are interested in the sum of the numbers. The result can be anywhere between two and twelve, but all results are not equally likely.

- We want to make a data frame that looks something like this:

red	green	sum
1	1	2
1	2	3
1	3	4
1	4	5
1	5	6
1	6	7
2	1	3
2	2	4
2	3	5
⋮	⋮	⋮

- There are 36 possibilities here and we could type them all out, but that would be very tedious and time-consuming. Instead we will take a shortcut. In **R**, open a new script and run:

```
green <- c(1:6, 1:6, 1:6, 1:6, 1:6, 1:6)
```

Or using a much shorter command:

```
green<-rep(1:6, times=6)
```

(**R** knows that 1:6 means 1, 2, 3, 4, 5, 6.)

```
red <- sort(green, decreasing = FALSE)
```

Can you work out what is going on here? An alternative command is:

```
red <- rep(1:6, times=rep(6,6))
```

(That is, make a column with 6 ones, then 6 twos, and so on)

```
dice <- data.frame(red, green)
dice
```

- On inspection you can see that this matrix contains every possible combination of numbers on the red and green dice. We are interested in the sum of the two numbers

so we create a new column which contains the sum:

```
dice$sum <- dice$red + dice$green
```

Print the new form of the data frame to make sure that worked.

- Now we want to know the probability of each result. Type

```
counts <- table(dice$sum)
counts <- data.frame(counts)
counts
```

- This makes a table showing how many times each number appears in the column `dice$sum`. To get the probability of each result occurring, divide the frequency by 36. Call this column `prob` and print the new counts table for the experiment:

```
counts$prob <- counts$Freq / 36
counts
```

- Of course it is nicer to present the probabilities as fractional values rather than as the decimals allocated by R, but the above code gives the general method for producing the discrete probability distribution under examination.

- Make sure you understand what each line of code does. If you don't, then ask one of the instructors.

Complete part A of your answer sheet

- Now, suppose we were to roll a pair of dice 50 times and write down the sum each time. That would be tedious. But would you agree that picking a number at random from the "sum" column is the same thing as rolling a pair of dice? We can ask R to do this automatically. Type:

```
sample50 <- sample(dice$sum, 50, replace = TRUE)
prob50_table <- table(sample50)
prob50 <- data.frame(prob50_table)
prob50$relfreq <- prob50$Freq/50
prob50
```

- The first line asks for a sample of size 50 from the column `dice$sum`, sampling with replacement where all of the 36 entries are equally likely. The sample is saved as a table called `sample50`.

- The second line calculates how many times each number occurs in `sample50` and saves it as a table called `prob50_table`. The third line turns that table into a dataframe.

- The fourth line adds a new column called `relfreq`, the relative frequency of each result. The last line prints the table.

  Complete part B of your answer sheet

- Now take a sample of size 500 from the column `dice$sum` by typing:

  ```
  sample500 <- sample(dice$sum, 500, replace = TRUE)
  ```

- Find the relative frequency of each number in this sample. (Use similar code as you did in section B – you will have to make changes in some places!)

  Complete Part C of your answer sheet

2 Classical Distributions

- In this section we will revisit the discrete distributions we looked at in lectures using **R**.

2.1 Binomial Distribution

- Recall that the binomial distribution describes situations where we have a number of independent trials, the outcome of each trial is either "success" or "failure" and the probability of success is the same for each trial.

- For example the number of heads observed when a fair coin is flipped 10 times follows a binomial distribution where the number of trials is 10 and the probability of success on each trial is 0.5. A "success" in this case is observing heads.

- Here we will experiment with a hypothetical weighted coin, where the probability of getting heads is slightly greater than the probability of getting tails.

- The function `rbinom` simulates values from a binomial distribution. Run the following code:

  ```
  bin_samp <- rbinom(n=1, size=100, prob=0.55)
  ```

- This code flips the hypothetical coin 100 times, records a 1 if we get heads and 0 if tails is observed, it then returns how many times heads was observed. Is the result close to what you expected?

- We can calculate the theoretical probability of observing this many heads in 100 tosses of the weighted coin using `dbinom`.

  ```
  dbinom(x=number_heads, size=100, prob=0.55 )
  ```

  Enter your observed number of heads where `number_heads` is in the above code.

- Finally, we can calculate the probability of observing less than or equal to this number of heads using `pbinom`.

  ```
  pbinom(q=number_heads, size=100, prob=0.55, lower.tail=TRUE)
  ```

- The probability of observing more than this number of heads can be obtained by replacing `lower.tail=TRUE` with `lower.tail=FALSE` or simply subtracting your original result from 1.

- Look up the help files for these functions to learn more.

  Complete Part D of your answer sheet

2.2 Hypergeometric Distribution

- To win the national lottery jackpot in Ireland you must match the six numbers drawn from a drum containing the numbers 1 to 45.

- The lottery game is the same as taking a sample of 6 balls without replacement from a drum containing 39 red balls and 6 blue balls. To win the jackpot you must find that all the balls in your sample are blue. Thus the number of numbers you match on the lotto follows a hypergeometric distribution.

- To calculate the probability of winning the lotto using **R** we will use the function `dhyper`.

  ```
  dhyper(x=6, m=6, n=39, k=6)
  ```

  Look up the help files to see what `x`, `m`, `n` and `k` are.

- We can also simulate the lotto by using the function `rhyper`. The following code simulates playing the lotto 8,320 times (equivalent to playing twice a week for 80 years!) and reports the number of numbers matched on each occasion. The `table` function then tabulates the results.

  ```
  lotto <- rhyper(nn=8320, m=6, n=39, k=6)
  table(lotto)
  ```

  Did you ever match all 6 numbers? Experiment with the number of times the game is played by changing `nn`. (**Make sure you have everything saved as you may crash the computer by making `nn` too large!**)

- Plot a barchart of your results.

  Complete Part E of your answer sheet

2.3 Poisson Distribution

- Lets suppose that the number of cars passing through the M50 toll follows a Poisson distribution with $\lambda = 90,000$ cars per day (this rate is approximately correct!).

- We can simulate a typical week at the toll using the `rpois` function. The `rpois` function simulates a value from the Poisson distribution with a certain rate parameter. The following code simulates 7 values from a poisson distribution with rate 90,000.

  ```
  rpois(n=7, lambda=90000)
  ```

- The lowest cost of passing through the toll is €2.10. Calculate the lower bound of the amount of money taken in at the toll for the week you have simulated by summing the numbers of cars on each day and multiplying the result by 2.1.

- The probability of observing less than some number of cars per day can be calculated using `ppois`. The `ppois` function returns $\mathbb{P}(X \leq x)$ where $X \sim Poisson(\lambda)$ (It will return $\mathbb{P}(X > x)$ if you add the input `lower.tail=FALSE`. The function then returns the upper tail of the distribution rather than the lower tail.). To calculate the probability that less than or equal to 89,000 cars pass through the toll on a given day, run the following code:

  ```
  ppois(q=89000, lambda=90000)
  ```

  Was this greater or less than you expected? The probability that exactly 89,000 cars pass through the toll on any given day can be calculated using the code:

  ```
  dpois(x=89000, lambda=90000)
  ```

  Complete Part F of your answer sheet

Make sure you save your R script to a USB key or to Google drive before you log off.

Statistics & Probability
2 – Discrete Random Variables

A. What is the probability of getting a total of 8 when you roll two dice?

What is the probability of getting an even number when you roll two dice?

B. In your sample of 50, what proportion of the time did you get a total of 8?

What proportion of the time did you get an even number?

C. In your sample of 500, what proportion of the time did you get a total of 8?

What code did you use to find this?

Compare the proportion of times you got an 8 in each sample with the true probability of getting an 8. What do you notice?

D. How many heads did you observe in tossing the hypothetical coin 100 times? Was this more or less what you expected?

What is the theoretical probability of observing this many heads?

What is the probability of observing more heads than this in 100 tosses of this coin? Give the code you used.

E. What is the probability of winning the lotto jackpot?

Sketch the `barplot` of the results of your simulation.

F. How many cars passed through the M50 toll in your simulation?

What is the lower bound for the amount of revenue taken at the toll that week? Give the code you used to calculate it.