LAB01 – WEEK 2

DATA 101 – MAKING PREDICTIONS USING DATA

CONTACT INFO

Victor Frunza

UTA Studying Computer Science & Data Science

vfrunza@student.ubc.ca

CONVERTING SUMS TO R

The symbol for sigma, $\sum$, means to “sum”.

Sum of the first 4 integers

$$\sum_{j=1}^{4} j = 1 + 2 + 3 + 4$$

CONVERTING SUMS TO R

The symbol for sigma, $\sum$, means to "sum".

Sum of the first 4 integers

$$\sum_{j=1}^{4} j = 1 + 2 + 3 + 4$$

Sum of the first 4 squares

$$\sum_{j=1}^{4} j^2 = 1^2 + 2^2 + 3^2 + 4^2$$

CONVERTING SUMS TO R

The symbol for sigma, $\sum$, means to “sum”.

Sum of the first 4 integers

$$\sum_{j=1}^{4} j = 1 + 2 + 3 + 4$$

Sum of the first 4 squares

$$\sum_{j=1}^{4} j^2 = 1^2 + 2^2 + 3^2 + 4^2$$

In R code:

```
sum(1:4)
## [1] 10
```

CONVERTING SUMS TO R

The symbol for sigma, $\sum$, means to “sum”.

Sum of the first 4 integers

$$\sum_{j=1}^{4} j = 1 + 2 + 3 + 4$$

In R code:

```
sum(1:4)
## [1] 10
```

Sum of the first 4 squares

$$\sum_{j=1}^{4} j^2 = 1^2 + 2^2 + 3^2 + 4^2$$

In R code:

```
sum((1:4)^2)
## [1] 30
```

CONVERTING SUMS TO R

Sums can be converted into equations rather than being a series that must be calculated.

$$\sum_{j=1}^{n} j = \frac{n(n+1)}{2}$$

CONVERTING SUMS TO R

Sums can be converted into equations rather than being a series that must be calculated.

$$\sum_{j=1}^{n} j = \frac{n(n+1)}{2}$$

In R we could write the following to compare the two methods:

```
n<- c(100,200,400,800)
n*(n+1)/2
```

```
## [1] 5050 20100 80200 320400
```

CONVERTING SUMS TO R

Sums can be converted into equations rather than being a series that must be calculated.

$$\sum_{j=1}^{n} j = \frac{n(n+1)}{2}$$

In R we could write the following to compare the two methods:

```
n<- c(100,200,400,800)
n*(n+1)/2

## [1] 5050 20100 80200 320400
```

```
c(sum(1:100),sum(1:200),sum(1:400),sum(1:800))

## [1] 5050 20100 80200 320400
```

CONVERTING SUMS TO R

Once a sum has been computed, we can store the result into a vector for later processing.

```r
n <- 1:100 # this stores all the possible n values
sumsUp2n <- n*(n+1)/2 # this stores each of the sums
```

CONVERTING SUMS TO R

Once a sum has been computed, we can store the result into a vector for later processing.

```
n <- 1:100 # this stores all the possible n values
sumsUp2n <- n*(n+1)/2 # this stores each of the sums
```

We can check the first 5 elements of sumsUp2n to see that the calculation was done correctly:

```
sumsUp2n[1:5]
## [1] 1 3 6 10 15
```

CONVERTING SUMS TO R

We can calculate the sum $\sum_{j=1}^{n} j^2$ and compare it with $\frac{n(n+1)(2n+1)}{6}$ for $n = 10, 20, 40, 80$.

```
n<- c(10,20,40,80)
n*(n+1)*(2*n+1)/6
```

```
## [1] 385 2870 22140 173880
```

CONVERTING SUMS TO R

We can calculate the sum $\sum_{j=1}^{n} j^2$ and compare it with $\frac{n(n+1)(2n+1)}{6}$ for $n = 10, 20, 40, 80$.

```
n<- c(10,20,40,80)
n*(n+1)*(2*n+1)/6
```

```
## [1] 385 2870 22140 173880
```

```
c(sum((1:10)^2),sum((1:20)^2),sum((1:40)^2),sum((1:80)^2))
```

```
## [1] 385 2870 22140 173880
```

CONVERTING SUMS TO R

Just like before, we can save the results of our squared sum into a vector.

```r
n <- 1:100 # this stores all the possible n values
sumSquaresUp2n <- n*(n+1)*(2*n+1)/6 # this stores each of the sums

sumSquaresUp2n[1:5]
```

```
## [1] 1 5 14 30 55
```

CONVERTING SUMS TO R

Sometimes you may want to sum non-regular series. You can do that by creating a vector and using the sum function. For example, if your series was 15, 28, -42:

$$\sum x_j = x_1 + x_2 + x_3 = 15 + 28 - 42$$

In R, we would write:

```
x <- c(15, 28, -42)
sum(x)

## [1] 1
```

CONVERTING SUMS TO R

To square the same series in R:

```
x <- c(15, 28, -42)
sum(x^2)

## [1] 2773
```

THE SEQ() FUNCTION

The seq() function is similar to : but it allows for a larger variety of patterns. If we want a list of the odd numbers from 13 through 26, we can type:

```
seq(13, 26, 2) # seq(start, end, step)
```

```
## [1] 13 15 17 19 21 23 25
```

THE SEQ() FUNCTION

We can also save the results like before. To make a sequence starting at 1, ending at 100, and counting by 7, we can write:

```
count1to100by7 <- seq(1, 100, 7) # seq(start, end, step)
count1to100by7
```

```
## [1] 1 8 15 22 29 36 43 50 57 64 71 78 85 92 99
```

THE LENGTH() FUNCTION

We can use the length() function to count the numbers in this sequence:

```
length(count1to100by7)

## [1] 15
```

THE REP() FUNCTION

The rep() function creates repeated patterns. For example, if you wanted to repeat the number 3, 7 times, you would write:

```
rep(3, 7)
```

```
## [1] 3 3 3 3 3 3 3
```

THE REP() FUNCTION

The rep() function creates repeated patterns. For example, if you wanted to repeat the number 3, 7 times, you would write:

```
rep(3, 7)

## [1] 3 3 3 3 3 3 3
```

If you wanted to repeat the sequence (2, 4, 8), 3 times:

```
rep(c(2, 4, 8), 3)

 ## [1] 2 4 8 2 4 8 2 4 8
```

THE REP() FUNCTION

If you wanted to repeat each element of (2,4,8), 3 times, you can type.

```
rep(c(2, 4, 8), each = 3)

## [1] 2 2 2 4 4 4 8 8 8
```

THE REP() FUNCTION

If you wanted to repeat each element of (2,4,8), 3 times, you can type.

```
rep(c(2, 4, 8), each = 3)

## [1] 2 2 2 4 4 4 8 8 8
```

If I want to repeat each element a different number of times, say 4 2's, 7 4's and 2 3's, you'd write:

```
rep(c(2, 4, 8), c(4, 7, 2))

 ## [1] 2 2 2 2 4 4 4 4 4 4 4 8 8
```

THE REP() & SEQ() COMBOS

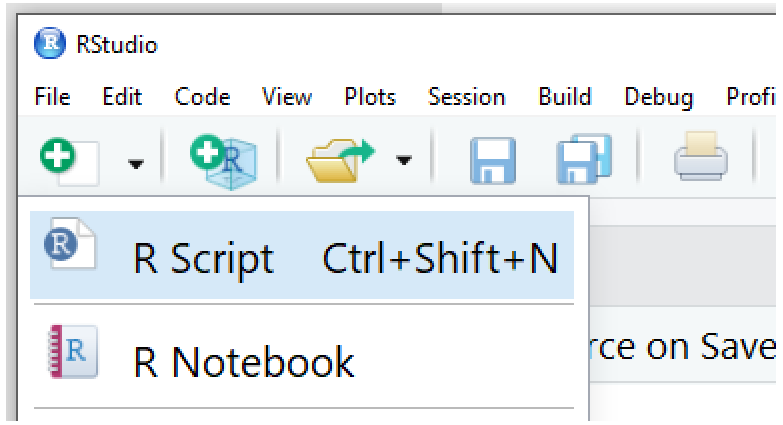
You can combine rep() and seq() to produce all sorts of patterns.

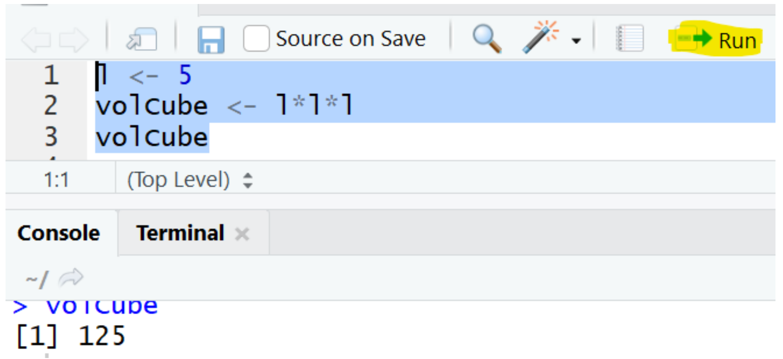
```
rep(seq(2,11,3), seq(7, 1, -2))

# seq(2,11,3) = 2 5 8 11
# seq(7, 1, -2) = 7 5 3 1

## [1] 2 2 2 2 2 2 2 5 5 5 5 5 8 8 8 11
```

R STUDIO

- If you're using R Studio, you can either use the console to write your code or write it in a file you can save. To do this, click on the green plus icon in the top left, and select R Script.

- To run all lines of code you must highlight/select all lines of code. Otherwise the run button will only execute one line.





```
l <- 5
volCube <- l*l*l
volCube
```

```
> volCube
[1] 125
```