MAGIST

Geog 413

Module 1 Exercise 3: Simple Descriptive Statistics

Simple Statistical Functions

R is handy when it comes to generating the most commonly used descriptive statistics. Generate the following vector x

```
> x <- c(0,1,1,2,3,5,8,13,21,34)
```

Find the mean, median, variance and standard deviation of the variable x:

```
> mean(x)
```

[1] 8.8

```
> median(x)
```

[1] 4

```
> var(x)
```

[1] 121.7333

```
> sd(x)
```

[1] 11.03328

Now reinforce your understanding of these measures by calculating them on your own. From your lecture materials, you know that the mean is the sum of the values of all the observations in a dataset divided by the number of observations, so

```
> mymean <- sum(x)/length(x)
```

```
> mymean
```

[1] 8.8

The variance of x is equal to the average sum of the squares of each observation about its mean, so

```
> myvar <-sum((x-mean(x))^2)/length(x)
```

```
> myvar
```

[1] 109.56

Wait a minute, this value is not the same as that given above by the function var(x). Any ideas? Perhaps R calculates the variance and standard deviation with (n-1) in the denominator. Let's check

```
> myvar <-sum((x-mean(x))^2)/(length(x)-1)
```

```
> myvar
```

[1] 121.7333

Yes, that was the issue. Be careful then with your use of built in functions, make sure you know exactly what they are calculating. It should be easy then to generate the standard deviation

```
> mysd <-sqrt(sum((x-mean(x))^2)/(length(x)-1))
```

```
> mysd
```

[1] 11.03328

When you start working with "real" datasets, you might often find that you have some missing data. Look at the following example where NA means the value is not available

```
> x2 <- c(0,1,1,2,3,NA)
```

```
> mean(x2)
```

[1] NA

```
> sd(x2)
```

[1] NA

Don't forget that the mean and standard deviation, as well as other descriptive statistics are calculated using all values of a variable of interest. R is being very cautious by not returning the mean and standard deviation in this example. It is really asking, what do I do with the value NA? If you have some NA observations in a dataset, and you don't think that they represent anything important, you can tell R to ignore them in most functions using the option na.rm=TRUE

```
> mean(x2, na.rm=TRUE)
```

[1] 1.4

```
> sd(x2, na.rm=TRUE)
```

[1] 1.140175

The Apply Function

R is also pretty handy because many of its functions can be applied to matrices, arrays and data frames. Let's look at a simple example, using different code to generate a matrix of 12 values separated into 3 rows

```
> x3 <- matrix(runif(12), nrow=3)
```

```
> x3
```

```
          [,1]        [,2]      [,3]      [,4]
[1,] 0.9697116 0.001156777 0.1134691 0.9827973
[2,] 0.6399909 0.132097497 0.9312456 0.9922162
[3,] 0.9441731 0.446453034 0.2634679 0.6853969
```

The runif function generates random observations from a uniform probability distribution. Without specifying further options, the random observations are generated from the range of values between 0 and

1. Don't worry about the arguments here, we will think about them in more detail in Exercise 4. For now, just imagine that you have a 3 row and 4 column matrix stored in the object x3.

For x3, we could generate the overall mean value

```
> mean(x3)
```

[1] 0.591848

What if we wanted the means for each row or each column of the matrix (or data frame)? You could use the apply function

```
> apply(x, margin, function)
```

Where x identifies the object, margin identifies the dimension of the object you are interested in (for a matrix 1=rows, 2 = columns), and function specifies what type of value to generate). Let's try this to generate row means

```
> apply(x3, 1, mean)
```

[1] 0.5167837 0.6738876 0.5848728

And for column standard deviations

```
> apply(x3, 2, sd)
```

[1] 0.1834370 0.2288572 0.4353514 0.1744868

Now let's read into R the file "children.csv" used in the lecture on frequency distributions. A copy of this file is in the class data folder. You can download the file onto your own computer and then read it into R as follows. (Remember to included the filepath or set your working directory first.)

```
> children <- read.csv(filepath here, sep=",")
```

Preview the observations

```
> children$x
```

[1] 0 0 1 1 1 1 2 2 2 2 2 2 2 2 2 3 3 3 4 4

You can ask for an overall summary of the number of children within the 20 families of this dataset

```
> summary(children$x)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00    1.00    2.00    1.95    2.25    4.00
```

Frequency Tables and Histograms

Just for fun, how about a quick way of building the frequency table and a histogram of the children data in R.

For ease, let's create a new object

```
> x <-children$x
```

Find the number of observations for the variable x

```
> length(x)
[1] 20
```

A quick way of finding the absolute frequency of a variable is to use the table function

```
> af <-table(x)
> af
x
0 1 2 3 4
2 4 9 3 2
```

Define the relative frequency

```
> rf <-af/length(x)
> rf
x
  0   1   2   3   4
0.10 0.20 0.45 0.15 0.10
```

Define the cumulative relative frequency

```
> crf <-cumsum(rf)
> crf
  0   1   2   3   4
0.10 0.30 0.75 0.90 1.00
```

Convert the 3 variables into a data frame (albeit a bit messy)

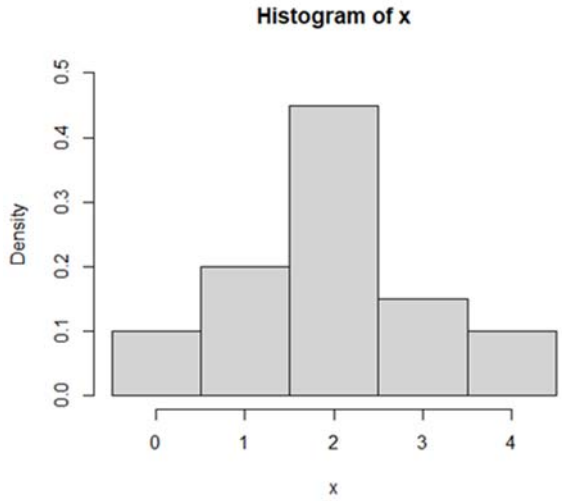
```
> freqtab <- data.frame(af, rf, crf)
> freqtab
  x Freq x.1 Freq.1  crf
0 0    2   0   0.10 0.10
1 1    4   1   0.20 0.30
2 2    9   2   0.45 0.75
3 3    3   3   0.15 0.90
4 4    2   4   0.10 1.00
```

We could use cbind for a tidier object

```
> freqtab <- cbind(af, rf, crf)
> freqtab
  af   rf  crf
0  2 0.10 0.10
1  4 0.20 0.30
2  9 0.45 0.75
3  3 0.15 0.90
4  2 0.10 1.00
```

Generate a histogram (You have to specify the range of the x-axis and identify breaks for the histogram bars, check ?hist)

```
> hist(x, xlim=c(-0.5,4.5), breaks=c(-0.5,0.5,1.5,2.5,3.5,4.5), ylim=c(0,0.5), prob=T)
```



How to generate a cumulative frequency histogram?

```
> x <- c(0,0,1,1,1,1,2,2,2,2,2,2,2,2,3,3,3,4,4)

> x

 [1] 0 0 1 1 1 1 2 2 2 2 2 2 2 2 3 3 3 4 4

> d <- hist(x, xlim=c(-0.5,4.5), breaks=c(-0.5,0.5,1.5,2.5,3.5,4.5), prob=T)

> d$counts <- cumsum(d$counts)

> plot(d)
```