Introduction to Statistics and Data Analysis I: Lab 2 Summary Statistics

Dr. Niccole M. Pamphilis

Loading Data

Recall last week we covered a few different ways to load data in to R. First we could load a file that R is familiar with, or we could load a data file from another stats program. I will be loading an R data file with the command below. You may follow this or you can open the "haven" package and open the .dta file. Please note that both data files are available on the Course Canvas page in the section for today's lab.

```
data2 <-readRDS(file = "Scotland_data.rds")

#To use the command above you will need to make sure your working directory is set to
#where you have stored this data file. Otherwise R will not be able to locate it.
```

A dataset called data2 should now have appeared in your environment (top-right pane in RStudio).

Information on the number of observations and variables should also be provided there. You can also get the number of observations, variables and variable names using functions in R. To see the variable names use the names() function. R will only provide you variable names and the description of variables will be provided separately in a codebook.

To get the number of observations and variables use the dim() (dimensions) function. This function gives you two numbers, the first is the number of observations (this is also the number of rows in your dataset) and the second is the number of variables (the number of columns) in your dataset. In this dataset we have 100 observations (rows) and 12 variables (columns).

```
#which variables are in the dataset

names(data2)
```

```
##  [1] "ID"                     "age"                    "region_Scot"
##  [4] "gross_personal_income"  "education_level_simple" "leftright"
##  [7] "partyid"                "polattention"           "july19_glasgow_23"
## [10] "july19_glasgow_24"      "july19_glasgow_25"      "july19_glasgow_26"
```

```
#how many observations and variables are there in your dataset, dimension 1 (100, in this example) is t
#of observations and dimension 2 the number of variables (12)

dim(data2)
```

```
## [1] 100  12
```

Whenever R provides more than one piece of information, we can choose to access or display these one at a time by using square brackets []. For example, if we wanted to see only the number of observations we can write dim(d)[1], while dim(d)[2] will give us the number of variables. The principle here works the same in other functions as well. For example, using names(d)[3] will give the name of the third variable in the dataset.

```r
dim(data2)[1]
```

```
## [1] 100
```

```r
dim(data2)[2]
```

```
## [1] 12
```

```r
names(data2)[3]
```

```
## [1] "region_Scot"
```

There are many other functions that help you get a glimpse of your dataset. For example you can use the function str() (for structure) to get a quick look at the dataset. This will provide you with the number of observations, variables, variable names and also information on the variable type and values for first few observations. The variable type is listed num for numbers, int for integers (numbers with no decimals) and Factor for factors (categorical variables).

```r
str(data2)
```

```
## 'data.frame':    100 obs. of  12 variables:
##  $ ID                   : num  1 2 3 4 5 6 7 8 9 10 ...
##  $ age                  : num  47 43 65 42 70 74 66 49 74 46 ...
##  $ region_Scot          : num  NA 4 8 7 3 7 3 2 3 4 ...
##  $ gross_personal_income : num  7 6 16 16 9 9 8 4 3 15 ...
##  $ education_level_simple: num  3 2 3 3 2 2 2 2 3 2 ...
##  $ leftright            : num  1 12 6 4 7 5 7 6 8 12 ...
##  $ partyid              : num  10 2 2 10 1 3 1 10 1 99 ...
##  $ polattention         : num  8 3 11 8 8 8 8 7 6 6 ...
##  $ july19_glasgow_23    : num  2 4 5 6 6 3 4 2 6 4 ...
##  $ july19_glasgow_24    : num  1 4 5 7 5 4 3 2 6 4 ...
##  $ july19_glasgow_25    : num  5 4 7 2 1 3 5 6 2 4 ...
##  $ july19_glasgow_26    : num  2 2 3 3 2 2 2 1 3 5 ...
##  - attr(*, "datalabel")= chr ""
##  - attr(*, "time.stamp")= chr " 3 Aug 2020 14:30"
##  - attr(*, "formats")= chr [1:12] "%12.0g" "%12.0g" "%12.0g" "%12.0g" ...
##  - attr(*, "types")= int [1:12] 255 255 255 255 255 255 255 255 255 255 ...
##  - attr(*, "val.labels")= chr [1:12] "" "age" "region" "gross_personal_income" ...
##  - attr(*, "var.labels")= chr [1:12] "" "Age" "Scottish Region" "Income - gross personal" ...
##  - attr(*, "expansion.fields")=List of 10
##   ..$ : chr [1:3] "leftright" "note1" "In political matters people talk of ‘the left’ and ‘the right
##   ..$ : chr [1:3] "leftright" "note0" "1"
##   ..$ : chr [1:3] "partyid" "note1" "Generally speaking, do you think of yourself as Labour, Conserva
##   ..$ : chr [1:3] "partyid" "note0" "1"
##   ..$ : chr [1:3] "july19_glasgow_23" "note1" "How much do you agree or disagree with the following s
##   ..$ : chr [1:3] "july19_glasgow_23" "note0" "1"
##   ..$ : chr [1:3] "july19_glasgow_24" "note1" "Do you agree or disagree that there is enough voter fi
##   ..$ : chr [1:3] "july19_glasgow_24" "note0" "1"
##   ..$ : chr [1:3] "july19_glasgow_25" "note1" "Do you agree or disagree that there are sufficient sa
##   ..$ : chr [1:3] "july19_glasgow_25" "note0" "1"
##  - attr(*, "version")= int 12
```

Measures of Central Tendency

The Mean

The basic command to produce the mean in R is "mean". After telling R what to do, you will need to tell R where to find the variable. To do this we start with the name of the dataset (here it is data2) then we tell it which variable in the dataset to use (here age). Notice we connect the variable to the data set using the "$" sign.

```
mean(data2$age)
```

```
## [1] 59.52
```

A lot of times our variables will have missing data. This it typically represented by an NA comment in R. If this is in your variable and you use the command above, R will give you an error because it does not know what you want it to do with an "NA".

When this happens we use an option to tell R how to handle the missing data. The option is "na.rm=T"" (read that as "NA" remove equals true) tells R to ignore the NA comments for this command.

```
mean(data2$leftright, na.rm=T)
```

```
## [1] 8.99
```

If other issues arise it can be helpful to checkout the help file. Try typing help(mean) and the help window on the lower right should open up.

Median

Median. The median is easily calculated using the function median(), and for variables with missing values we can again set na.rm=T.

```
median(data2$age)
```

```
## [1] 62
```

```
quantile(data2$age)
```

```
##    0%   25%   50%   75%  100% 
## 33.00 48.75 62.00 70.00 83.00
```

```
#for variables with missing values

median(data2$leftright, na.rm=T)
```

```
## [1] 6
```

Mode

The mode is the value of a variable that appears most often in your data. It is an appropriate measure of central tendency for categorical variables (e.g. nominal or ordinal variables).

```
table(data2$region_Scot)
```

```
## 
##  1  2  3  4  5  6  7  8 
## 12 14 19 10 11 16  8  9
```

```r
table(data2$partyid)
```

```
##
##  1  2  3  4  7 10 12 99
## 17 15 11 18  4 26  2  7
```

```r
#missing values are not displayed in the table by default. To find out how to see missing
#values use the help (?)

table(data2$region_Scot, useNA="always")
```

```
##
##    1    2    3    4    5    6    7    8 <NA>
##   12   14   19   10   11   16    8    9    1
```

We may wish to get proportions rather than counts in our frequency tables. For this, you can use the prop.table() function. The first of the examples below shows the proportions of respondents by region in Scotland. In the second example we have multiplied the table by a 100 to get percentages. This shows that about 12.12% of respondents were from Region 1 in Scotland. (In two weeks we will explore how to add labels to our data sets so 1 corresponds to the name of region).

```r
#to get proportions you can use the prop.table function
prop.table(table(data2$region_Scot))
```

```
##
##          1          2          3          4          5          6          7
## 0.12121212 0.14141414 0.19191919 0.10101010 0.11111111 0.16161616 0.08080808
##          8
## 0.09090909
```

```r
#to get percentages
prop.table(table(data2$region_Scot))*100
```

```
##
##         1         2         3         4         5         6         7         8
## 12.121212 14.141414 19.191919 10.101010 11.111111 16.161616  8.080808  9.090909
```

Measures of Dispersion

Standard Deviation

Standard deviation measures dispersion around the mean. It can be used only for numeric, interval/ratio type data. In R the standard deviation can be calculated using the the function sd(). Again, if you have missing values in the data, include na.rm=T in the code (as shown with the mean earlier).

```r
sd(data2$age)
```

```
## [1] 13.10446
```

```r
sd(data2$leftright, na.rm=T)
```

```
## [1] 13.17596
```

Percentiles and interquartile range.

Percentiles and interquartile range can be used to measure dispersion for both interval/ratio and ordinal variables. Percentiles are calculated using the quantile() function. By default this gives you the smallest value, the 1st quarter, the 2nd quarter (median), the 3rd quarter and the maximum value of your variable (respectively 0%, 25%, 50%, 75% and 100% in the output).

You can also specify exactly which percentiles you wish R to calculate, for example, using 0.5 will give you the median, and using c(0.25, 0.75) and (data2$age, c(0.25, 0.75)) will give you the 1st and 3rd quarters.

```
#quartiles-data split into 4 groups
quantile(data2$age)
```

```
##    0%   25%   50%   75%  100%
## 33.00 48.75 62.00 70.00 83.00
```

```
#median
quantile(data2$age, 0.5)
```

```
## 50%
##  62
```

```
#1st and 3rd quartile
quantile(data2$age, c(0.25, 0.75))
```

```
##   25%   75%
## 48.75 70.00
```

```
#Calculating the IQR by hand
quantile(data2$age, 0.75) - quantile(data2$age, 0.25)
```

```
##   75%
## 21.25
```

```
#interquartile range
IQR(data2$age)
```

```
## [1] 21.25
```

Minimum, maximum and range.

The minimum and maximum values are the lowest and highest values of your variables and the difference between these will be the range. These values are easily calculated using functions min(), max() and range() for both minimum and maximum values. If you have missing values in the data, use na.rm=T.

```
#To get the minimum value for a variable
min(data2$age)
```

```
## [1] 33
```

```
#To get the maximum value for a variable
max(data2$age)
```

```
## [1] 83
```

```
#Gives us both maximum and mimimum values
range(data2$age)
```

```
## [1] 33 83
```

```r
#Calculate the range
max(data2$age) - min(data2$age)
```

```
## [1] 50
```

```r
#Another way to calculate the range
#R takes the maximum value, which is the second piece of data produce by the range command
#and subtracts the minimum, which is the first piece of data produced by the range command

range(data2$age)[2] - range(data2$age)[1]
```

```
## [1] 50
```

Nominal Level Data

There is no good measure of dispersion for nominal variables, but we can describe variation in a nominal variable by looking at the frequency table. Let's take a look at the frequency distribution for confidence in integrity of UK elections. The codebook for the data tells us that there were five possible answer choices: very confident (1), fairly confident (2), not very confident (3), not at all confident (4), and Do Not Know (5). The table below shows that 100 respondents answered the question.

```r
table(data2$july19_glasgow_26)
```

```
##
##  1  2  3  4  5
##  5 64 18  6  7
```

The table shows us most respondents were fairly confident in the integrity of the elections. Most respondents were clustered between fairly with a few being not very confident, before decreasing at the extremes of very confident and not confident at all.

Summary Measures

Finally, to get multiple descriptive statistics all at once you can use the summary() function. It will also give you the number of missing values. Note however that for very large numbers this function provides the first three digits and rounds everything after this. Thus the individual functions such as mean() and sd() will be more accurate. For nominal and ordinal variables that are coded in words in your dataset, the summary function will give you a frequency table. Nominal and ordinal variables that are coded in numbers are treated as numeric interval/ratio data by summary() and thus the results will not be appropriate.

```r
summary(data2$age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   33.00   48.75   62.00   59.52   70.00   83.00
```