stcp-karadimitriou-InputR

Inputting Data and Opening Datasets in R

Inputting data in R

The marks for a group of students before (pre) and after (post) a teaching intervention are recorded below:

Student	Pre Marks	Post Marks	Difference
1	18	22	4
2	21	25	4
3	16	17	1
4	22	24	2
5	19	16	-3
6	24	29	5
7	17	20	3
8	21	23	2
9	23	19	-4
10	18	20	2
11	14	15	1
12	16	15	-1
13	16	18	2
14	19	26	7
15	18	18	0
16	20	24	4
17	12	18	6
18	22	25	3
19	15	19	4
20	17	16	-1
MEAN (SD)	18.40 (3.15)	20.45 (4.05)	20.5 (2.83)

Data should be entered with one row per subject and one column per variable. Input the data using the `matrix()` command. Remember that all the elements are inputted by column by default. If generally you would like to input the entries by row, put the attribute `byrow=TRUE` into the matrix command.

```
marks<-
matrix(c(1:20,18,21,16,22,19,24,17,21,23,18,14,16,16,19,18,20,12,22,15,
17,22,25,17,24,16,29,20,23,19,20,15,15,18,26,18,24,18,25,19,16),nrow=20,n
col=3)
```

where `c()` introduces the vector of all the values separated by commas starting with the student column where 1:20 stands for allocating the values 1 through 20 for the number of each student (ID). Next the data for pre marks, followed by the data for post marks and the `nrow=` and `ncol=` attributes specify the number of rows and columns that are desired for the matrix respectively. The difference column will be calculated later.
In order to add variable names to the columns, use the `colnames()` command and in order to give a name for each row, use the `rownames()` command.

```
colnames(marks)<-c('Student','Pre_Marks','Post_Marks')
```

It is always need to be specified that the data that are inputted are datasets. This has to do with specific running commands, for linear regression for instance, where a dataset or else a `data.frame` object needs to be inputted.

```
marks<-data.frame(marks)
```

Variables are referred to by datasetname$variablename e.g. `marks$Pre_Marks` but attaching the data you are working on enables variables to be referred to by their name only e.g. `Pre_Marks`.

```
attach(marks)
```

To view the dataset, type its name.

```
marks
```

Creating new variables in the dataset

You can calculate new variables and put them into the dataset with the $ command. In order to calculate the difference in marks for instance, use the following commands

```
marks$difference<-Post_Marks-Pre_Marks
```

then attach the dataset again

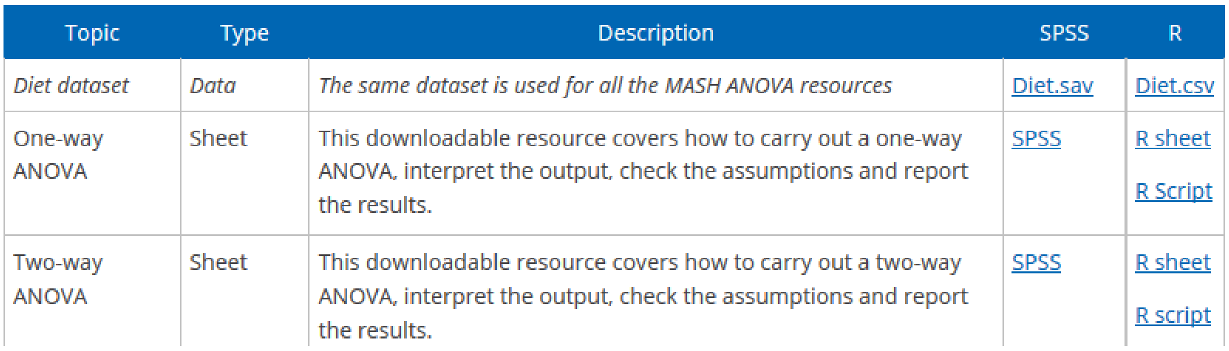
```
attach(dietR)
```

Check the command has worked by looking at the dataset.

```
marks
```

Opening data sets in R

The datasets and script files for all the R resources are downloadable from the website. For instance, if someone is interested on downloading and implementing a One-way ANOVA in R, the Diet.csv dataset and ANOVA script files are needed. This is an example of opening a csv (Comma delimited) file in R.



Topic	Type	Description	SPSS	R
Diet dataset	*Data*	*The same dataset is used for all the MASH ANOVA resources*	Diet.sav	Diet.csv
One-way ANOVA	Sheet	This downloadable resource covers how to carry out a one-way ANOVA, interpret the output, check the assumptions and report the results.	SPSS	R sheet R Script
Two-way ANOVA	Sheet	This downloadable resource covers how to carry out a two-way ANOVA, interpret the output, check the assumptions and report the results.	SPSS	R sheet R script

Once you download and save the csv file, you need to know the exact location of the file in order to be able to open it through the R commands. You will need to change the command depending on where you have saved the file. The command that is used in order to open the csv files is `read.csv('location',header=T,sep=',')`

The location indicates the path of the file. For instance if you have the Diet.csv file saved on the Desktop then the path is *C:\Users\studentid\Desktop\Diet.csv*

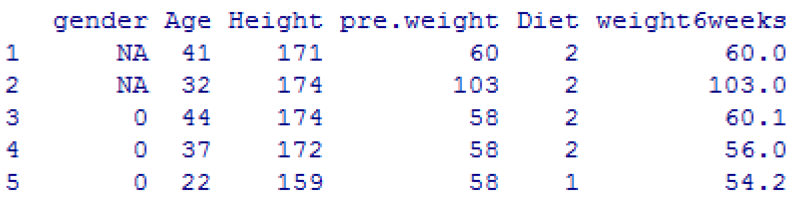
In order for R to understand and open the path we need to use double dash, i.e. write *C:\\Users\\studentid\\Desktop\\Diet.csv* instead.

The header=T attribute specifies that the first row of the dataset has the name for each variable, whilst the `sep=','` indicates that the columns in the csv file are separated with commas. It should be used with care since on each computer this may be different. The most common separation is either a comma or a semicolon. In the case of a semicolon put `sep=';'` instead.

Therefore the appropriate command now becomes

```
dietR<-
read.csv('C:\\Users\\studentid\\Desktop\\Diet.csv',header=T,sep=',')
```

The Diet file that was read in R should look like this



```
  gender Age Height pre.weight Diet weight6weeks
1     NA  41    171         60    2         60.0
2     NA  32    174        103    2        103.0
3      0  44    174         58    2         60.1
4      0  37    172         58    2         56.0
5      0  22    159         58    1         54.2
```

In case that you want to call a variable from the dataset, the '$' can be used. This means that if for instance we need to call the gender variable, `dietR$gender` should be typed.

Alternatively, we can tell R to use the diet dataset until further notice by using *`attach(dataset)`* so that 'gender' can be used instead of `dietR$gender`.

```
attach(dietR)
```

Creating new variables in the dataset

You can calculate new variables and put them into the dataset with the $ command. In order to calculate the weight lost by person for instance we can use the commands

Calculating the (difference in weight before and after the diet) and putting it into the diet dataset

```
dietR$weightlost<-pre.weight-weight6weeks
```

attach again the dataset so that 'weightlost' can be used in further calculations

```
attach(dietR)
```

Specifying factor variables

Variables like gender or Diet where each number indicates a group should be specified as categorical by the used. Tell R that 'Diet' is a factor using `as.factor(variable)`.

```
Diet<-as.factor(Diet)
```

Tell R that 'gender' is a factor and specify that 0 is for women and 1 for men with the labels=c() attribute

```
gender<-factor(gender,labels=c('Woman','Man'))
```

If a categorical variable is ordered, then you should use the command

```
ordered(variable,labels=c('Low','Medium','High'))
```