Data Frames and Clean Data Files

Levente Orban

2018-07-26

Data Frames

Until now, we loaded data into vectors using the function. However, this is quite limited. Most of our data will have several variables or factors that we need to load into a single data set for statistical analysis. This is where data frames enter.

A data frame is used for storing tables of values. For example, we created a bunch of vectors yesterday, which we can combine now into a single data frame:

```
employmentStatus = c(TRUE,FALSE,FALSE,TRUE,TRUE,TRUE,FALSE)
siblings = c(1,3,3,1,2,2,1)
mydata = data.frame(employment = employmentStatus, sibs = siblings)
mydata
```

```
##   employment sibs
## 1       TRUE    1
## 2      FALSE    3
## 3      FALSE    3
## 4       TRUE    1
## 5       TRUE    2
## 6       TRUE    2
## 7      FALSE    1
```

Importing Data

However, this data was hand-coded into R, which is a very time consuming process, especially if you have hundreds, thousands, or worse, hundreds of thousands of records. R can handle importing data from a variety of formats including the friendly csv (**c**omma **s**eparated **v**alues) format, or less ideal Excel or SPSS formats.

Data can come from a variety of sources. One increasingly more common source is the Internet. To import data directly from the web:

```
myData = read.csv(url("http://llorban.com/psyc2300/1100_exam_grades.csv"))
```

If we wanted to take a look at this data, we could simply type the name of the variable and hit enter:

```
myData
```

```
##    student_id exam_2 exam_3
## 1           1  49.62  43.13
## 2           2   0.00   0.00
## 3           3  84.62  82.50
## 4           4  64.23  68.13
## 5           5  66.15   0.00
## 6           6  49.81  36.13
## 7           7  44.42  52.50
## 8           8  45.96   0.00
## 9           9  54.62  26.25
## 10         10  86.92  73.75
## 11          11  40.77  50.00
## 12          12  50.58  60.00
## 13          13  75.58  79.38
## 14          14   0.00   0.00
## 15          15  42.12   0.00
## 16          16  43.65   0.00
## 17          17  72.31  71.88
## 18          18   0.00   0.00
## 19          19  80.77  74.38
## 20          20  63.01  42.57
## 21          21  62.66  54.05
## 22          22  52.32  39.20
## 23          23  42.94  35.80
## 24          24  84.94  91.22
## 25          25  59.45  65.67
## 26          26  61.94  58.11
## 27          27  41.18  34.46
## 28          28  58.02  65.55
## 29          29  67.83  49.32
## 30          30  51.39  52.70
## 31          31  51.50  43.24
## 32          32  88.59  91.90
## 33          33  48.73  34.46
## 34          34  38.24  49.32
## 35          35  81.82  87.16
## 36          36  60.25  53.39
## 37          37  52.50  43.24
## 38          38  68.18  68.24
## 39          39  64.71  52.03
## 40          40  58.81  47.96
## 41          41  52.14  47.95
## 42          42   0.00   0.00
## 43          43  70.32  50.66
## 44          44  64.44  52.04
## 45          45  70.14  66.24
## 46          46  77.90  77.04
## 47          47  60.96  60.82
## 48          48  67.38  50.68
## 49          49  60.52  54.74
## 50          50  56.77  45.26
```

However, this is produces a long fairly useless output. If we are just interested in a quick overview with the variables and types of values they take, we can run the *head()* function:

```r
head(myData)
```

```
##   student_id exam_2 exam_3
## 1          1  49.62  43.13
## 2          2   0.00   0.00
## 3          3  84.62  82.50
## 4          4  64.23  68.13
## 5          5  66.15   0.00
## 6          6  49.81  36.13
```

Much better. We can tell that three variables, one of which takes on an integer, and the other two taking on float values.

In some cases, the data will reside on your computer. In this case, we need to set the path to the file on the computer. This will get tricky now, because each of our computers could have different paths. In the case of my computer, I can load a data set I just downloaded like so:

```r
dataFromFile = read.csv("~/Downloads/levente_weight.csv")
head(dataFromFile)
```

```
##                  Date Weight Fat.mass Bone.mass Muscle.mass Hydration
## 1 2017-12-29 10:00:18  169.2     38.8        NA          NA        NA
## 2 2017-12-28 11:06:28  170.8     38.1        NA          NA        NA
## 3 2017-12-24 10:28:29  168.8     35.4        NA          NA        NA
## 4 2017-12-21 09:08:12  168.9     36.6        NA          NA        NA
## 5 2017-12-20 09:14:08  168.1     36.3        NA          NA        NA
## 6 2017-12-19 10:45:56  168.6     38.6        NA          NA        NA
##   Comments
## 1       NA
## 2       NA
## 3       NA
## 4       NA
## 5       NA
## 6       NA
```

If you have a directory that you would like to use, and don't want to type in the whole path each time, R can help. getwd() will tell you where R will look by default.

```r
getwd()
```

```
## [1] "/Users/llorban/Documents/Service to the Profession/2018 R Workshop/git_repo/rworkshop/day2"
```

So if I were to type in just the file name, *r read.csv("levente_weight.csv")* without the path, R would assume *read.csv("/Users/llorban/levente_weight.csv")* . In my particular case, because the file is in the Downloads folder, typing in just the file name would result in an error. Go ahead, try it.

It is possible to change the directory R searches by default with the *r setwd()* command.

```r
setwd("/Users/llorban/Downloads")
```

Once this command is executed, I can simply use the *read.csv()* with just the file name.

Importing Excel or SPSS files is also possible with addons.

{r} # library(xlsx) # myData = read.xlsx("Grossman and Kross 2014 Study 2.xlsx") # head(myData)

{r} # library(Hmisc)