Data Import into R

Susan Holmes (c)

Example 1: From a txt file

```
setwd("/Users/susan/RWork/")
```

In the session 6 on matrices and vectors we created a matrix and saved it to a text file (`matHap.txt`).

We are now going to see various ways of importing that data. A `.txt` file is an ascii file and we can explore it using:

```
file.show("matHap.txt")
```

It is important to notice that at the top of the file we have a different pattern than later, here are the first lines:

```
"DYS19" "DXYS156Y" "DYS389m" "DYS389n" "DYS389p" "DYS389q" "DYS390m" "DYS390n" "DYS390p" "DYS390q" "DYS392"
"H1" 14 12 4 12 3 10 8 10 1 4 15 13 0 1 1
"H3" 15 13 4 13 3 9 8 10 1 4 13 12 0 1 1
"H4" 15 11 5 11 3 10 8 10 1 4 11 14 0 1 1
```

- If we count the elements in these rows we see that the first row has 15 character strings.
- The second row (and following rows) have one character string followed by 15 numbers.
  So the rows are not all the same length.

Some formats are not humanly readable in this way (in particular excel `.xls` and `.xlsx` as well as `.RData`)

We are starting with the `.txt` file because it is simpler to follow exactly what is going on. As you see, the file contains a mixture of characters and numbers.

We can use the function `read.table`

```
haptable=read.table("matHap.txt")
haptable
```

```
##    DYS19 DXYS156Y DYS389m DYS389n DYS389p DYS389q DYS390m DYS390n DYS390p
## H1    14       12       4      12       3      10       8      10       1
## H3    15       13       4      13       3       9       8      10       1
## H4    15       11       5      11       3      10       8      10       1
## H5    17       13       4      11       3      10       7      10       1
## H7    13       12       5      12       3      11       8      11       1
## H8    16       11       5      12       3      10       8      10       1
## H9    16        11        5       11        3       10       8       10       1
##    DYS390q DYS392 DYS393 YAPbcbc SRY1532bb X92R7bb
## H1       4     15     13       0         1       1
## H3       4     13     12       0         1       1
## H4       4     11     14       0         1       1
## H5       4     14     12       0         1       1
## H7       4     14     14       0         1       1
## H8       4     11     15       0         1       1
## H9       4     11     14       0         1       1
```

```
class(haptable)
```

```
## [1] "data.frame"
```

The function `read.table` has automatically created a `data.frame` with columns named as given in the first line (the header) of the data file. It has also used the first column to name the observations.

Reading in a file from excel in csv format

Many difficulties can arise when importing files from other software. We will start with a file that was created in excel by choosing the export as `csv` (comma separated variables). A `csv` file is also a text file, with commas separating all the values. Using such a file is the best strategy because you can carefully monitor the characters in the file.

Importing includes several difficulties can occur because - You have to know where the file is, if you download it from the web it may not be in the folder you are working from.
- The variables may have spaces or strange characters in them. - Somethimes the header has a # in front which makes R think that it has to ignore that row. - The file might have unlabeled missing values, causing the array not to seem rectangular.
- The row.names and col.names identifiers should not have duplicates, this troubles many functions.

```
file.show("Haplotype.csv")
```

There are many special read functions, try typing `?read` then the `tab` key to see a few of them.

We are going to read this file with `read.csv` which supposes that the separator is the comma `,` character.

You need to download the file `Haplotype.csv` to your computer. Here I have used my own path to this file as an illustration; your file might not necessarily be in your working directory. You should edit the command below to show the path where you downloaded the file to.

```
Hap=read.csv("/Users/susan/Dropbox/stat32/Slides_ABCofR/Haplotype.csv")
head(Hap)
```

```
##   Individual DYS19 DXYS156Y DYS389m DYS389n DYS389p DYS389q DYS390m
## 1         H1    14       12       4      12       3      10       8
## 2         H3    15       13       4      13       3       9       8
## 3         H4    15       11       5      11       3      10       8
## 4         H5    17       13       4      11       3      10       7
## 5         H7    13       12       5      12       3      11       8
## 6         H8    16       11       5      12       3      10       8
##   DYS390n DYS390p DYS390q DYS392 DYS393 YAPbcbc SRY1532bb X92R7bb
## 1      10       1       4     15     13       0         1       1
## 2      10       1       4     13     12       0         1       1
## 3      10       1       4     11     14       0         1       1
## 4      10       1       4     14     12       0         1       1
## 5      11        1        4      14      14        0          1        1
## 6      10        1        4      11      15        0          1        1
```

The function `read.csv` looks for a header and if it sees a vector of characters as long as the number of columns, it supposes these are all variable names.

We can tell it that the first column actually contains the `row.names` as follows:

```
Hap=read.csv("/path/tofile/Haplotype.csv",row.names=1)
```

Question Check the row.names are correct now.

If you have a large file, you might want to start by using the function `scan()` on a small number of values to see what is there.

```
Hapscan<-scan("/Users/susan/Dropbox/stat32/Slides_ABCofR/Haplotype.csv",nlines=5,what="")
Hapscan
```

```
## [1] "Individual,DYS19,DXYS156Y,DYS389m,DYS389n,DYS389p,DYS389q,DYS390m,DYS390n,DYS390p,DYS390q,DYS392,DYS
## [2] "H1,14,12,4,12,3,10,8,10,1,4,15,13,0,1,1"
## [3] "H3,15,13,4,13,3,9,8,10,1,4,13,12,0,1,1"
## [4] "H4,15,11,5,11,3,10,8,10,1,4,11,14,0,1,1"
## [5] "H5,17,13,4,11,3,10,7,10,1,4,14,12,0,1,1"
```

Special package for reading excel files

Files ending in `.xls` and `.xlsx` created as excel files can be difficult because they can contain multiple spreadsheets.

A recent package developed to help import these into R is called `"readxl"`.

Let's look at the possibilities offered by `read_excel`.

```
install.packages("readxl")
```

```
library(readxl)
?read_excel
Hapall <- read_excel("/Users/susan/RWork/data/Haplotype.xlsx")
```

`readxl` is a recent package and creates a different type of output, similar to a data.frame, called a tibble. We reverse this and use the classical data.frame.

```
Hapall.df <- as.data.frame(Hapall)
Hapall.df[1:3,1:5]
```

```
##   Individual DYS19 DXYS156Y DYS389m DYS389n
## 1         H1    14       12       4      12
## 2         H3    15       13       4      13
## 3         H4    15       11       5      11
```

Problem case study.

```
mice<-read_excel("/Users/susan/RWork/data/TypicalExcelMess.xlsx")
mice.df=as.data.frame(mice)
mice.df[1:4,1:15]
```

Note, that this data is not provided - it serves only as an illustration

```
##   # QIIME v1.2.0-dev OTU table                    .1          .2          .3
## 1                         Date    01_11_07    01_14_07    01_16_07    01_18_07
## 2           Day post-infection           2           5           7           9
## 3                  Sample Name mLS.155.58 mLS.155.59 mLS.155.60 mLS.155.61
## 4     Salmonella level (CFU/g)         640         256        5675         7.5
##              .4          .5          .6          .7          .8          .9
## 1   01_25_07    01_29_07    02_07_07    02_16_07    04_13_07    04_18_07
## 2          16          20          29          38          94          99
## 3 mLS.155.62 mLS.155.63 mLS.155.64 mLS.155.65 mLS.155.66 mLS.155.67
## 4          38         113          94          39          14          14
##             .10        .11        .12        .13
## 1   04_21_07  10_25_06  10_27_06  11_03_06
## 2         102        -14        -12         -5
## 3 mLS.155.68 mLS.73.01 mLS.73.02 mLS.73.03
## 4          14          0          0          0
```

Summary of this Session:

- Careful data preparation is needed before importing your data as column names must not have spaces, nor be too long. A good preliminary step is to create a `csv` file from excel and inspect it.
- The header is the first line of the file that R uses to infer the variable names and number.
- We have learned how to import text files and csv files into data.frame in R.
- Useful functions for importing include `read.csv`, `read.table` and `scan`, they have different functions and arguments, the default for `read.csv` is to have a comma delimiting, other delimiters can be used with `read.delim`.

Question: Look up the complete official site R: R Data Import

If you have used other software and want to know how to import from that particular software, try installing the package `foreign` and seeing what is available.