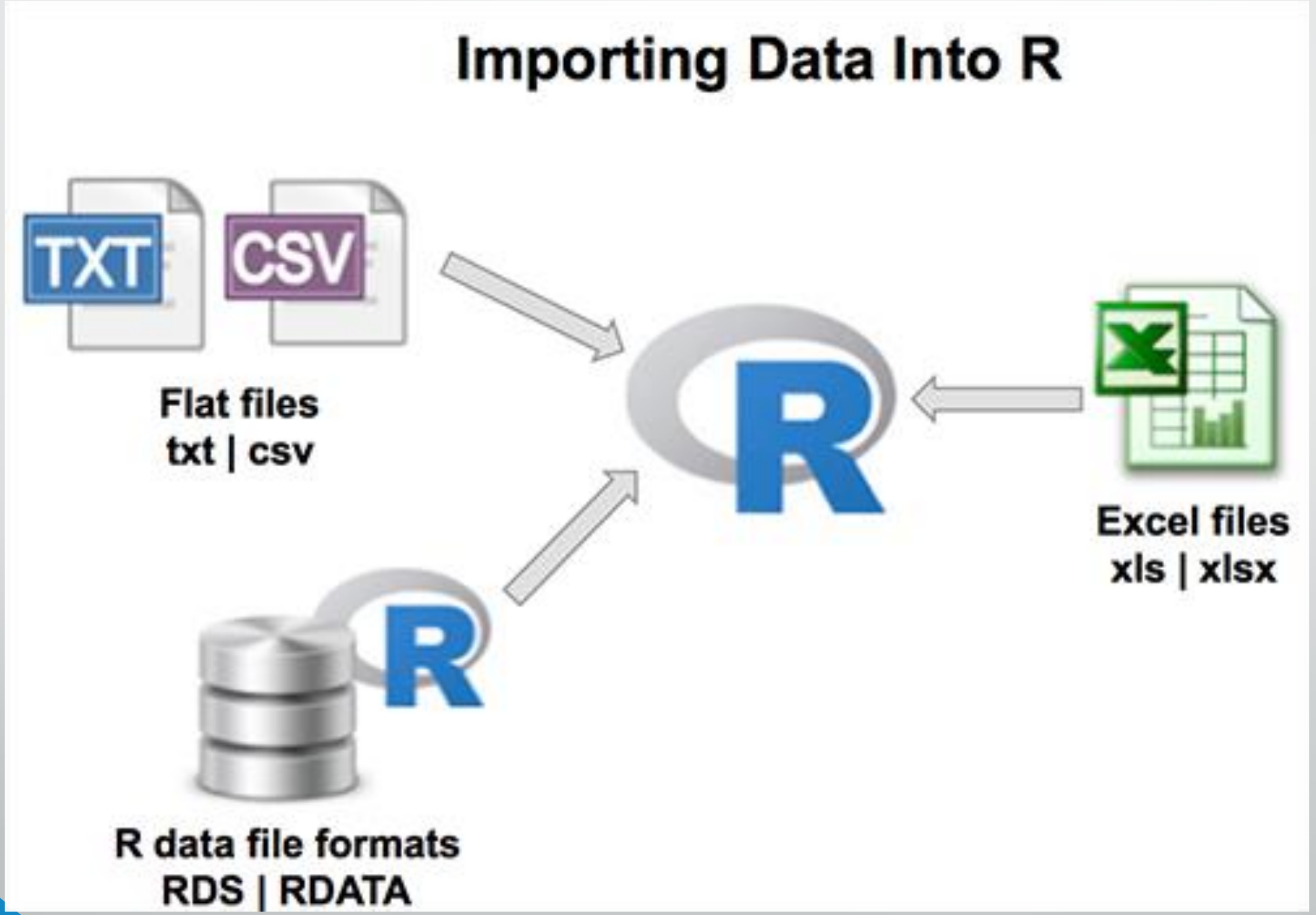
Big Data Analytics

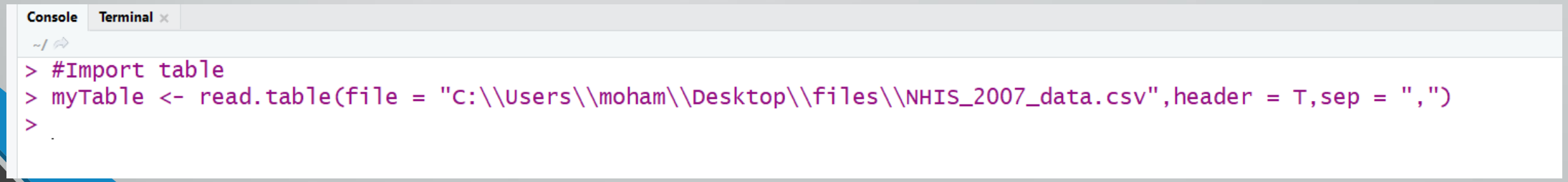

Importing Data Into R
TXT
CSV
Flat files
txt | csv
R
Excel files
xls | xlsx
R
R data file formats
RDS | RDATA

Reading Data Files with **read.table()**

The R base function **read.table()** is a general function that can be used to read a file in table format. The data will be imported as a data frame.

- The **read.table()** function has a few important arguments:

- **file**: the path to the file containing the data to be imported into R.
- **sep**: the field separator character. "\t" is used for tab-delimited file.
- **header**: logical value. If TRUE, **read.table()** assumes that your file has a header row, so row 1 is the name of each column. If that's not the case, you can add the argument **header = FALSE**.
- **dec**: the character used in the file for decimal points.



Variants of read.table()

- depending on the format of your file, several variants of read.table() are available including
- **read.csv():** for reading **"comma separated value"** files (".csv").
- **read.csv2():** variant used in countries that use a comma "," as decimal point and a semicolon ";" as field separators.
- **read.delim():** for reading "*tab-separated value*" files (".txt"). By default, point (".") is used as decimal points.
- **read.delim2():** for reading "*tab-separated value*" files (".txt"). By default, comma (",") is used as decimal points.

Variants of read.table() (Examples)

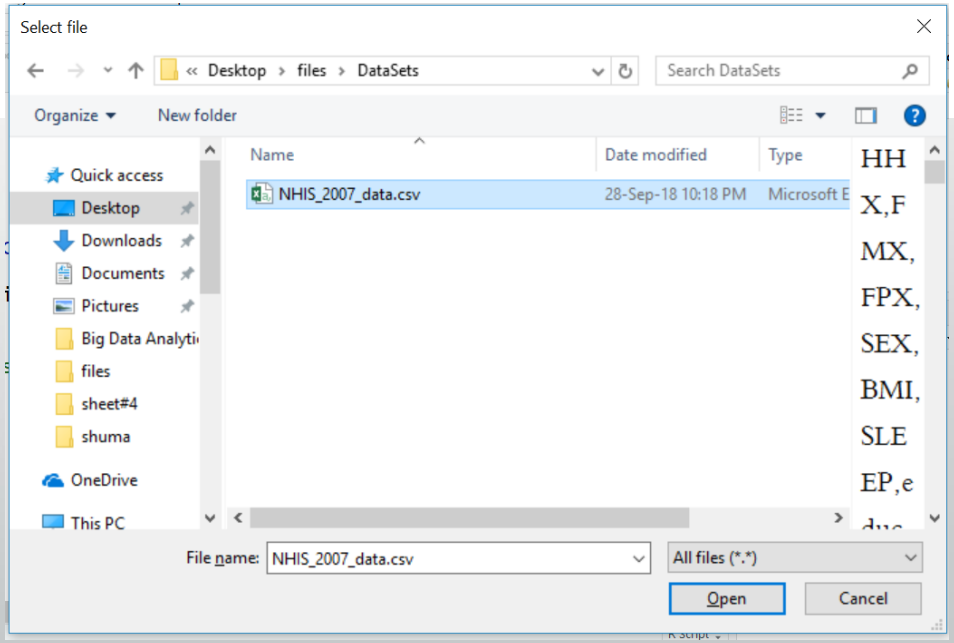
```
# Read tabular data into R
read.table(file, header = FALSE, sep = "", dec = ".")
# Read "comma separated value" files (".csv")
read.csv(file, header = TRUE, sep = ",", dec = ".", ...)
# Or use read.csv2: variant used in countries that
# use a comma as decimal point and a semicolon as field separator.
read.csv2(file, header = TRUE, sep = ";", dec = ",", ...)
# Read TAB delimited files
read.delim(file, header = TRUE, sep = "\t", dec = ".", ...)
read.delim2(file,header=TRUE,sep="\t",dec=",",...)
```

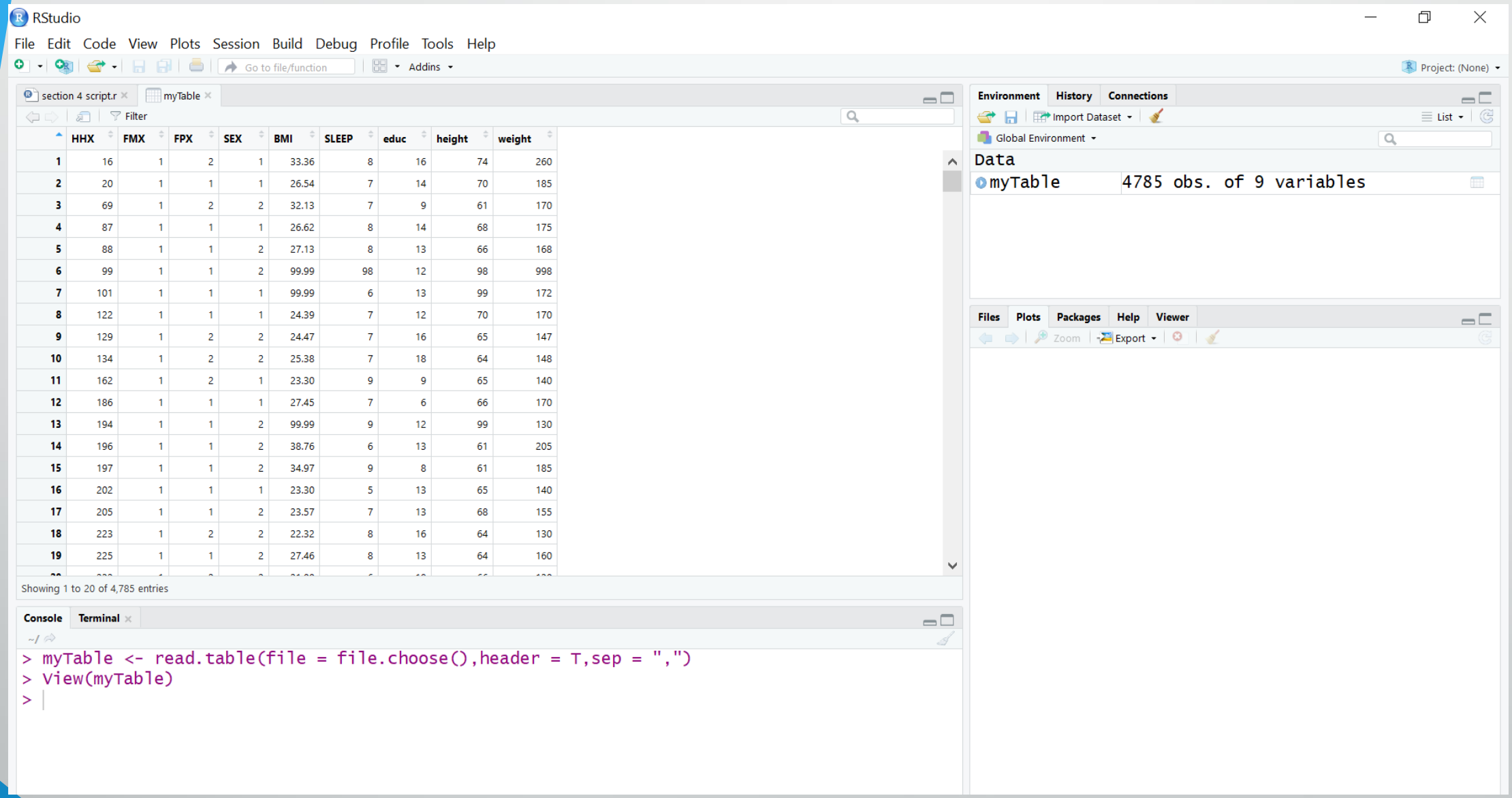
Function to open a file-choose dialogue file.choose()

Console Terminal

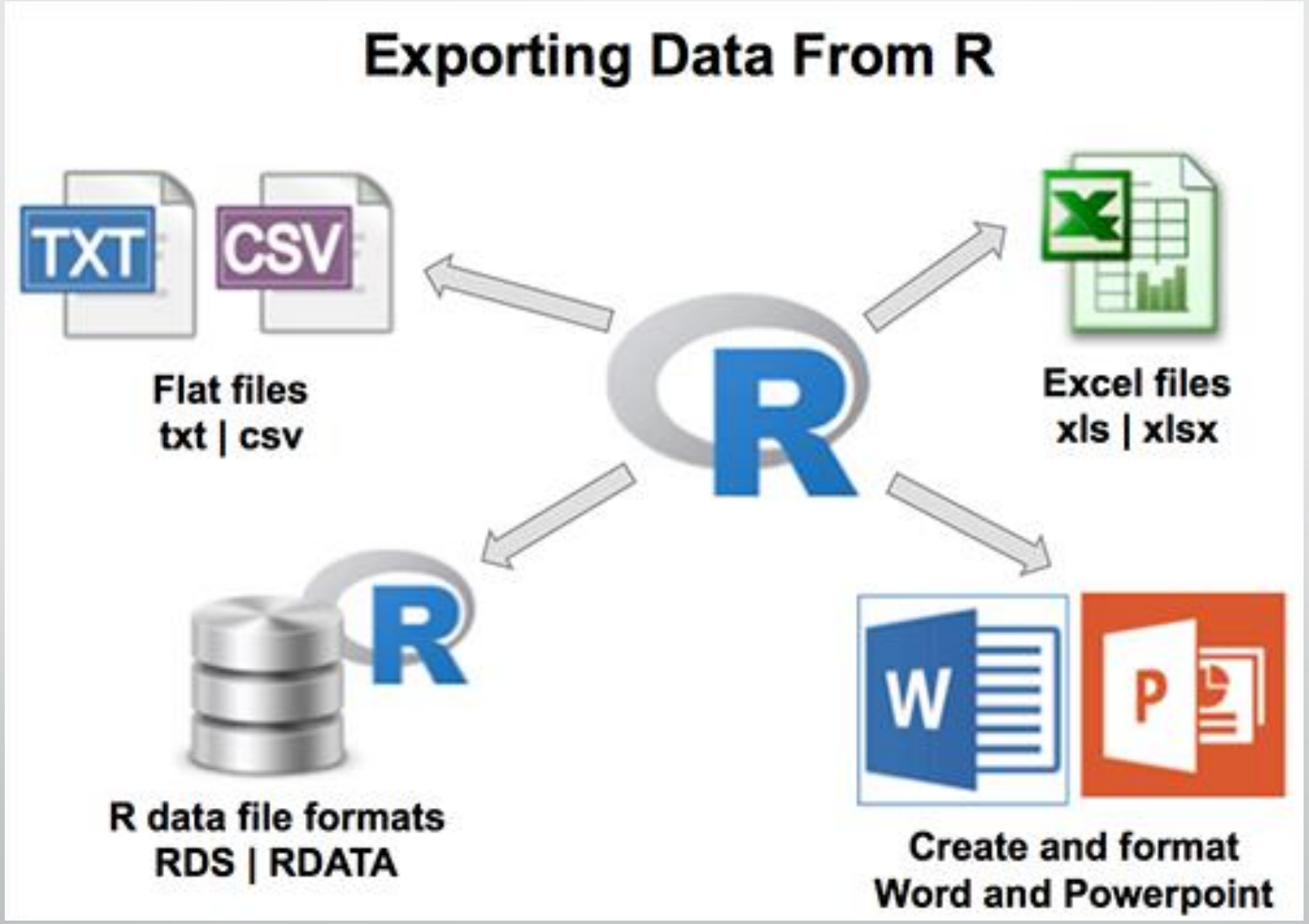
~/

```
> myTable <- read.table(file = file.choose(),header = T,sep = ",")
```




RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
section 4 script.r
myTable
Filter
HHX FMX FPX SEX BMI SLEEP educ height weight
1 16 1 2 1 33.36 8 16 74 260
2 20 1 1 1 26.54 7 14 70 185
3 69 1 2 2 32.13 7 9 61 170
4 87 1 1 1 26.62 8 14 68 175
5 88 1 1 2 27.13 8 13 66 168
6 99 1 1 2 99.99 98 12 98 998
7 101 1 1 1 99.99 6 13 99 172
8 122 1 1 1 24.39 7 12 70 170
9 129 1 2 2 24.47 7 16 65 147
10 134 1 2 2 25.38 7 18 64 148
11 162 1 2 1 23.30 9 9 65 140
12 186 1 1 1 27.45 7 6 66 170
13 194 1 1 2 99.99 9 12 99 130
14 196 1 1 2 38.76 6 13 61 205
15 197 1 1 2 34.97 9 8 61 185
16 202 1 1 1 23.30 5 13 65 140
17 205 1 1 2 23.57 7 13 68 155
18 223 1 2 2 22.32 8 16 64 130
19 225 1 1 2 27.46 8 13 64 160
Showing 1 to 20 of 4,785 entries
Console Terminal
~/
> myTable <- read.table(file = file.choose(),header = T,sep = ",")
> View(myTable)
>
Environment History Connections
Import Dataset
List
Global Environment
Data
myTable 4785 obs. of 9 variables
Files Plots Packages Help Viewer
Zoom
Export

Exporting Data From R



R base functions for writing data

- The R base function **write.table()** can be used to export a data frame or a matrix to a file.
- A simplified format is as follow:

```
write.table(x, file, append = FALSE, sep = " ", dec = ".", row.names = TRUE,
col.names = TRUE)
```

write.table() arguments

- **x**: a **matrix or a data frame** to be written.
- **file**: a character specifying the name of the result file.
- **sep**: the field separator string, e.g., sep = "\t" (for tab-separated value).
- **dec**: the string to be used as decimal separator. Default is "."
- **row.names:** either a logical value indicating whether the row names of x are to be written along with x, or a character vector of row names to be written.
- **col.names:** either a logical value indicating whether the column names of x are to be written along with x, or a character vector of column names to be written. If col.names = NA and row.names = TRUE a blank column name is added, which is the convention used for CSV files to be read by spreadsheets.

Variants of write.table()

- It's also possible to write **csv** files using the functions **write.csv()** and **write.csv2()**.

```
write.csv(my_data, file = "my_data.csv")
write.csv2(my_data, file = "my_data.csv")
```

Renaming columns with R base functions

- # get column names

colnames(myTable)

#Change column names

colnames(myTable) <- c("new col_1 name", " new col_2 name ")

Renaming columns (Cont.)

- # Rename column where names is "Length"
- **names**(myTable)[names(myTable) == "Length"] <- " new col_1 name "
  names(myTable)[names(myTable) == "Width"] <- " new col_2 name “
- It’s also possible to rename by index in names vector as follow.
- **names**(myTable)[1] <- “new col_1 name "
- **names**(myTable)[2] <- " new col_2 name "

Extracting rows by criteria with R base functions: subset()

- Extract rows where Length > 7 and Width ≤ 3. You can use this:
- my_data[my_data$Length > 7 & my_data$Width <= 3,]
- OR
- **subset**(my_data, subset(my_data $Length > 7 & subset(my_data $Width <= 3)

Useful Functions

Console Terminal

~/

```
> head(myTable)
  HHX FMX FPX SEX   BMI SLEEP educ height weight
1  16   1   2   1 33.36     8   16     74    260
2  20   1   1   1 26.54     7   14     70    185
3  69   1   2   2 32.13     7    9     61    170
4  87   1   1   1 26.62     8   14     68    175
5  88   1   1   2 27.13     8   13     66    168
6  99   1   1   2 99.99    98   12     98    998
> head(myTable, n=3)
  HHX FMX FPX SEX   BMI SLEEP educ height weight
1  16   1   2   1 33.36     8   16     74    260
2  20   1   1   1 26.54     7   14     70    185
3  69   1   2   2 32.13     7    9     61    170
> tail(myTable)
       HHX FMX FPX SEX   BMI SLEEP educ height weight
4780 53929   1   2   2 28.69     8   12     70    200
4781 53939   1   1   2 17.12     8   13     69    116
4782 53949   1   1   1 27.47     7   14     69    186
4783 53950   1   2   2 29.16     7   12     64    170
4784 53953   1   1   2 23.68     8   16     64    138
4785 53955   1   2   2 20.12     8   16     62    110
> tail(myTable,n=2)
       HHX FMX FPX SEX   BMI SLEEP educ height weight
4784 53953   1   1   2 23.68     8   16     64    138
4785 53955   1   2   2 20.12     8   16     62    110
>
```

Useful Functions

Console Terminal

~/

```
> myTable[c(1,2,3), ]
  HHX FMX FPX SEX   BMI SLEEP educ height weight
1  16   1   2   1 33.36     8   16     74    260
2  20   1   1   1 26.54     7   14     70    185
3  69   1   2   2 32.13     7    9     61    170
> myTable[c(1,2,3), c(1,2)]
  HHX FMX
1  16   1
2  20   1
3  69   1
> myTable[, c(1,2)]
      HHX FMX
1      16   1
2      20   1
3      69   1
4      87   1
5      88   1
6      99   1
7     101   1
8     122   1
9     129   1
10    134   1
11    162   1
12    186   1
13    194   1
14    196   1
15    197   1
16    202   1
17    205   1
18    223   1
```

Control Structures in R

- **if** (<condition>){
- ##do something
- } **else** {
- ##do something
- }

- #initialize a variable
- **N <- 10**

- #check if this variable * 5 is > 40
- **if** (N * 5 > 40){
- **print**("This is easy!")
- } **else** {
- **print**} ("It's not easy!")

- [1] "This is easy!"

for Loop

- **for** (<search condition>){
- #do something
- }

```
#initialize a vector
y <- c(99,45,34,65,76,23)

#print the first 4 numbers of this vector
For (i in 1:4){
  print (y[i])
}
[1] 99
[1] 45
[1] 34
[1] 65
```

while Loop

- #initialize a condition
- **Age <- 12**

- #check if age is less than 17
- **while** (Age < 17){
- **print** (Age)
- Age <- Age + 1 #Once the loop is executed, this code breaks the loop
- }
- [1] 12
- [1] 13
- [1] 14
- [1] 15
- [1] 16