Importing data from different formats

RECAP: Data Types / Classes

Data Types	Stores
real	floating point numbers
integer	integers
complex	Complex numbers
factor	categorical data
character	strings
logical	TRUE or FALSe
NA	Missing
NULL	Empty
Function	Function type

Vector

- A vector can only contain objects of the same class

```
a <- c(1,2,5.3,6,-2,4) # numeric vector
b <- c("one","two","three") # character vector
c <- c(TRUE,TRUE,TRUE,FALSE,TRUE,FALSE) #logical vector
```

Matrices

- All columns in a matrix must have the same class(numeric, character, etc.) and the same length. The general format is

```
#mymatrix <- matrix(vector, nrow=r, ncol=c, byrow=FALSE,
#dimnames=list(char_vector_rownames, char_vector_colnames))
#byrow=TRUE indicates that the matrix should be filled by rows
```

Factors

- Used to represent categorical data.
- Can be unordered or ordered. -A factor is like an integer vector where each integer has a label.

```
x <- factor(c("yes", "yes", "no", "yes", "no"))
x
```

```
## [1] yes yes no  yes no
## Levels: no yes
```

Missing Values

- Missing values are represented by the symbol **NA** (not available)
- Impossible values (e.g., dividing by zero) are represented by the symbol NaN (not a number)
- Can be unordered or ordered. -A factor is like an integer vector where each integer has a label.

```
  x <- NA
 # is.na(x) # returns TRUE of x is missing
# mean(x, na.rm=TRUE) # exclude missing in functions
# complete.cases() #returns the number of complete cases
```

Data Frames

- More general than a matrix, has different columns and can have different modes (numeric, character, factor, etc.)
- Used to store tabular data
- Can store data of different classes
- *read.table()* or *read.csv()* – used to load dataframes

Create Data Frames

```
data.frame(foo = 1:4, bar = c(T, T, F, F))
```

```
##   foo   bar
## 1   1  TRUE
## 2   2  TRUE
## 3   3 FALSE
## 4   4 FALSE
```

```
x <- c(1, 2,3,4,5,6,7,8,9)
y <- c("a","b","c","d","e","f","g","h","i")
df <- data.frame(x=x, y=y)
```

```
print(df)
```

```
##   x y
## 1 1 a
## 2 2 b
## 3 3 c
## 4 4 d
## 5 5 e
## 6 6 f
## 7 7 g
## 8 8 h
## 9 9 i
```

```
class(df)
```

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

Datasets

- R works with different types of datasets
- Base R functions *read.table* , *read.csv* and *read.delim* can read in data stored as text files, delimited by *almost anything*
- Data from other stat packages can be read using *foreign package?* and *Hmisc package*
  read.xlsx(file, sheetIndex=1) #excel files
  read.dta(file)# stata files

.RDA Data

- R Data type
- Can be created from other data sets -data <- load("profit.rda")
  -Saving a data frame as an rda
    - Save(data.frame, "dataset.rda")

Examples: 1

```
# Reading data from SPSS using the package "foreign"
library(foreign)
data1<-read.spss("D:/F-STAR/data/experim.sav", to.data.frame=T
```

```
## re-encoding from CP1252
```

```
data1
```

```
##    id    sex age                group fost1 confid1 depress1
## 1   4   male  23 confidence building    50       15        44
## 2  10   male  21 confidence building    47       14        42
## 3   9   male  25       maths skills    44       12        40
## 4   3   male  30       maths skills    47       11        43
## 5  12   male  45 confidence building    46       16        44
## 6  11   male  22       maths skills    39       13        43
## 7   6   male  22 confidence building    32       21        37
## 8   5   male  26       maths skills    44       17        44
```

Examples: 2

```
# Reading data from STATA using the package "foreign"
data2<-read.dta("D:/F-STAR/data/cr4.dta")
data2
```

```
##      y a order
## 1    4 1     1
## 2    6 1     2
## 3    3 1     3
## 4    3 1     4
## 5    1 1     5
## 6    3 1     6
## 7    2 1     7
## 8    2 1     8
## 9    4 2     1
## 10   5 2     2
## 11   4 2     3
```

Export to csv

- Use *write.csv* to export data frames

```
write.csv(Your DataFrame,"Path to export the DataFrame\\
File Name.csv", row.names = FALSE)
```

```
write.csv(data2,"D:/F-STAR/data/cr4.csv", row.names = FALSE)
```

Importing data from excel, csv

-This is the most common format we use -Data files saved in excel can be saved in the several formats:

```
.xlsx #Excel format -
.csv  #comma seperate values
.txt #tab delimited
```

- data must always first be organized in the right format see- Broman and Woo, 2017
  Link: https:
  //www.tandfonline.com/doi/full/10.1080/00031305.2017.1375989

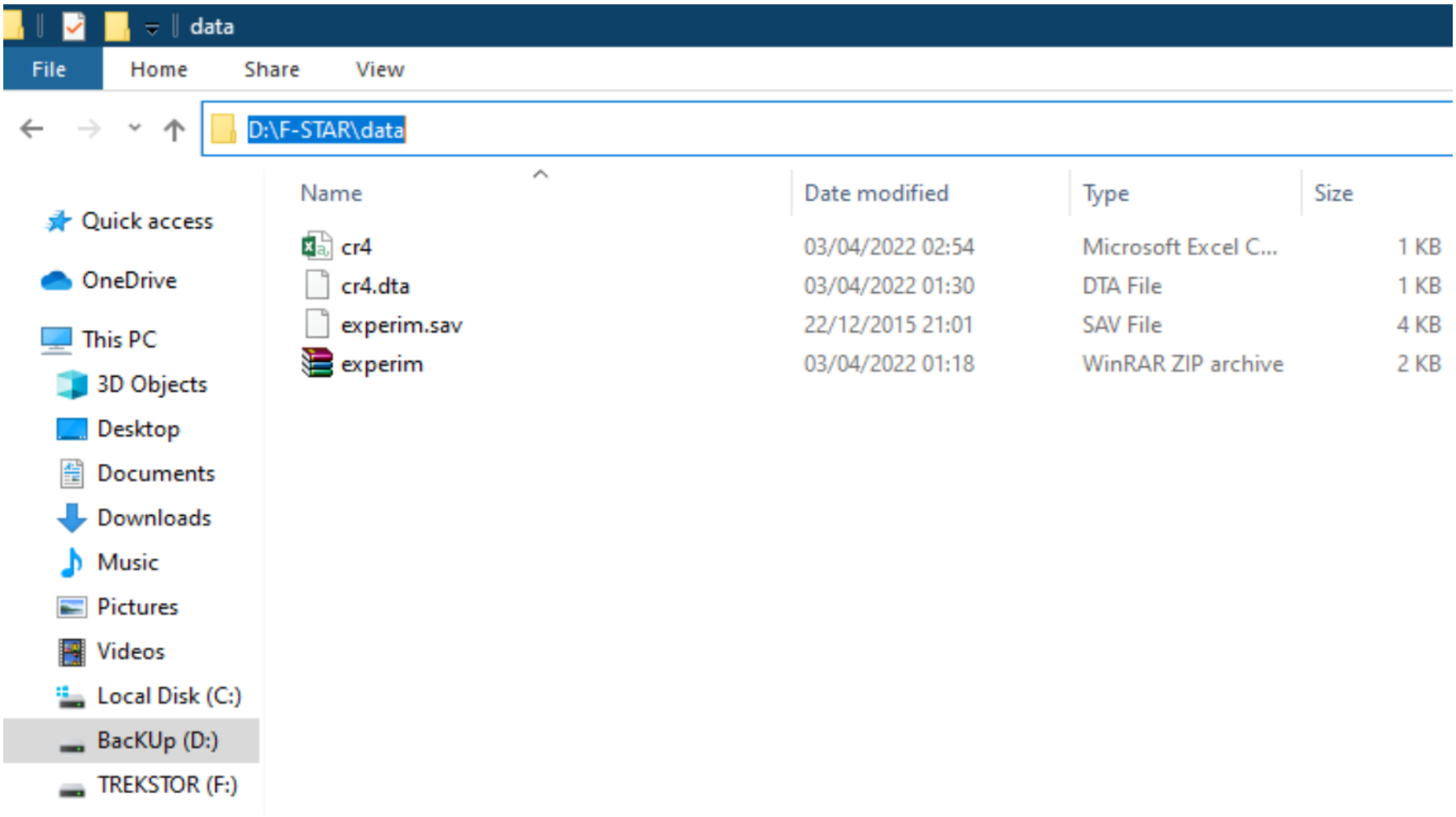
Example

```
read.csv("D:/F-STAR/data/cr4.csv")
```

```
##      y a order
## 1    4 1     1
## 2    6 1     2
## 3    3 1     3
## 4    3 1     4
## 5    1 1     5
## 6    3 1     6
## 7    2 1     7
## 8    2 1     8
## 9    4 2     1
## 10   5 2     2
## 11   4 2     3
## 12   3 2     4
## 13   2 2     5
```

Paths

- Note that when importing your data you must know where the file is
- Helps you tell the computer where to find the data



- Absolute path

Paths

- Types of paths
  - Absolute path -fixed and things must always be in the location specified
  - Relative path- allows you to move entire folders and retain folder structure
- It is best to create workspace/ projects using Rstudio
- This allows you to move folders and retain folder structure

Importing using Absolute path

```
# reading a txt file
# read.table("D:/F-STAR/data/cr4.txt")
```

```
# reading a excel file
library(readxl)
read_xlsx("D:/F-STAR/data/cr4.xlsx",sheet=1, col_names = TRUE)
```

```
## # A tibble: 32 x 3
##         y     a order
##     <dbl> <dbl> <dbl>
##  1      4     1     1
##  2      6     1     2
##  3      3     1     3
##  4      3     1     4
##  5      1     1     5
##  6      3     1     6
```

Importing using relative path

- Here use either the Rstudio Projects or set working directory *setwd* approach

```
# Set working directory
# read_xlsx("D:/F-STAR/data/cr4.xlsx",sheet=1)
setwd("D:/F-STAR")
read_xlsx("data/cr4.xlsx",sheet=1,col_names = TRUE)
```

```
## # A tibble: 32 x 3
##         y     a order
##     <dbl> <dbl> <dbl>
##  1      4     1     1
##  2      6     1     2
##  3      3     1     3
##  4      3     1     4
##  5      1     1     5
##  6      3     1     6
```