Introduction: Statistical Computing with R

Reading Data in and out of R

Allan Clark (adapted from B. Erni slides)

Department of Statistical Sciences, University of Cape Twon

28 February 2021

Contents

- How to read in different kinds of data.
- How to store results.
- Work through 'ReadWrite.r'

- NOTE: When copying the scripts below, ensure that the data files are in your working directory.
- If you don't have a particular library installed, you might encounter an error when using 'library(name of library)'. In this case, you will have to first install the package.
  i.e. 'install.package("name of package", dependencies=TRUE)'.

Reading text data

An extract of the data (fuel-frame.txt) that will be read in:

```
row.names,Weight,Disp.,Mileage,Fuel,Type
Eagle.Summit.4,2560,.97,33,3.030303,Small
Ford.Escort.4,2345,114,33,3.030303,Small
Ford.Festiva.4,1845,.81,37,2.702703,Small
Honda.Civic.4,2260,.91,32,3.125000,Small
Mazda.Protege.4,2440,113,32,3.125000,Small
Mercury.Tracer.4,2285,.97,26,3.846154,Small
Nissan.Sentra.4,2275,.97,33,3.030303,Small
Pontiac.LeMans.4,2350,.98,28,3.571429,Small
Subaru.Loyale.4,2295,109,25,4.000000,Small
Subaru.Justy.3,1900,.73,34,2.941176,Small
```

- Notice that the data is comma separated.
- Variables have column names.
- 'read.table' function can be used to import data.

Reading text data

```
#read in the file using read.table
#only the first 6 rows are displayed here

head(read.table("fuel-frame.txt", header=TRUE, sep=","))
```

```
##          row.names Weight  Disp. Mileage     Fuel  Type
## 1   Eagle.Summit.4   2560   0.97      33 3.030303 Small
## 2    Ford.Escort.4   2345 114.00      33 3.030303 Small
## 3   Ford.Festiva.4   1845   0.81      37 2.702703 Small
## 4    Honda.Civic.4   2260   0.91      32 3.125000 Small
## 5  Mazda.Protege.4   2440 113.00      32 3.125000 Small
## 6 Mercury.Tracer.4   2285   0.97      26 3.846154 Small
```

Reading text data

```
ff <- read.table("fuel-frame.txt", header=TRUE, sep=",")
```

- "fuel-frame.txt" : the name of the file imported.
- "header=T" : or "header=TRUE" : you do have column names, so read them in as well
- "sep=" : how is the data separated?
- Here the elements are separated by commmas.

Reading *.csv data

```
ff_csv <- read.table("fuel-frame.csv",
                     header=TRUE, sep=",")

head(ff_csv) #look at 'top' of ff_csv
```

```
##          row.names Weight  Disp. Mileage     Fuel  Type
## 1   Eagle.Summit.4   2560   0.97      33 3.030303 Small
## 2    Ford.Escort.4   2345 114.00      33 3.030303 Small
## 3   Ford.Festiva.4   1845   0.81      37 2.702703 Small
## 4    Honda.Civic.4   2260   0.91      32 3.125000 Small
## 5  Mazda.Protege.4   2440 113.00      32 3.125000 Small
## 6 Mercury.Tracer.4   2285   0.97      26 3.846154 Small
```

```
colnames(ff_csv) #the column names
```

```
## [1] "row.names" "Weight"    "Disp."     "Mileage"   "Fuel"      "Type"
```

Reading *.csv data (with missing values)

Lets read in data that contains missing values! These are denoted as NA.

```
ff2 <- read.table("fuel-frame2.csv", header=TRUE,
                  sep=",")
head(ff2)
```

```
##          row.names Weight  Disp. Mileage     Fuel  Type
## 1   Eagle.Summit.4     NA   0.97      33 3.030303 Small
## 2    Ford.Escort.4   2345 114.00      33       NA Small
## 3   Ford.Festiva.4   1845   0.81      NA 2.702703 Small
## 4    Honda.Civic.4   2260     NA      32 3.125000 Small
## 5  Mazda.Protege.4   2440 113.00      32 3.125000 Small
## 6 Mercury.Tracer.4   2285   0.97      26 3.846154 Small
```

Reading *.csv data (without column names)

Two different methods of adding column names to data!

Method 1

```
fuel<-read.csv("fuel-frame3.csv", header=FALSE,

                col.names=c("row.names","Weight",
                            "Disp.","Mileage",
                            "Fuel","Type"), sep=",")
```

Method 2

```
fuel<-read.csv("fuel-frame3.csv", sep=",",
                header=FALSE)

names(fuel)=c("row.names","Weight","Disp.","Mileage",
               "Fuel","Type")
```

Reading *.csv data (without column names)

```
fuel<-read.csv("fuel-frame3.csv", sep=",", header=FALSE)
```

```
##                  V1   V2     V3 V4       V5    V6
## 1   Eagle.Summit.4   NA   0.97 33 3.030303 Small
## 2    Ford.Escort.4 2345 114.00 33       NA Small
## 3   Ford.Festiva.4 1845   0.81 NA 2.702703 Small
## 4    Honda.Civic.4 2260     NA 32 3.125000 Small
## 5  Mazda.Protege.4 2440 113.00 32 3.125000 Small
## 6 Mercury.Tracer.4 2285   0.97 26 3.846154 Small
```

```
names(fuel)=c("row.names","Weight","Disp.","Mileage",
              "Fuel","Type")
```

```
##          row.names Weight  Disp. Mileage     Fuel  Type
## 1   Eagle.Summit.4     NA   0.97      33 3.030303 Small
## 2    Ford.Escort.4   2345 114.00      33       NA Small
## 3   Ford.Festiva.4   1845   0.81      NA 2.702703 Small
## 4    Honda.Civic.4   2260     NA      32 3.125000 Small
## 5  Mazda.Protege.4   2440 113.00      32 3.125000 Small
## 6 Mercury.Tracer.4   2285   0.97      26 3.846154 Small
```

Problems you will/could encounter

- can't find file - ensure that the file being read in is in your working directory!
- number of elements in row x is wrong.
- converted variables to wrong type.
- **0 or missing value?**
- and many others.

Create some objects in workspace

```
#lists the elements in your workspace
ls()
```

```
## [1] "ff_csv" "ff2"    "fuel"
```

```
#be careful before you run the next line!!!
#remove all of the objects in the workspace
rm(list=ls())
ls()
```

```
## character(0)
```

```
X <- matrix(rnorm(10000), nrow = 100); y <- 3:50

library(gapminder); data(gapminder)

ls()
```

```
## [1] "gapminder" "X"           "y"
```

Writing data to a file

Write data frame to a .txt/.csv (comma separated) file.

```
write.table(gapminder, file = "f1.txt")
write.csv(gapminder, file = "f1.csv")
```

The csv file looks like this!

,"country","continent","year","lifeExp","pop","gdpPercap"

1,"Afghanistan","Asia",1952,28.801,8425333,779.4453145

2,"Afghanistan","Asia",1957,30.332,9240934,820.8530296

3,"Afghanistan","Asia",1962,31.997,10267083,853.10071

4,"Afghanistan","Asia",1967,34.02,11537966,836.1971382

5,"Afghanistan","Asia",1972,36.088,13079460,739.9811058

6,"Afghanistan","Asia",1977,38.438,14880372,786.11336

7,"Afghanistan","Asia",1982,39.854,12881816,978.0114388

8,"Afghanistan","Asia",1987,40.822,13867957,852.3959448

9,"Afghanistan","Asia",1992,41.674,16317921,649.3413952

10,"Afghanistan","Asia",1997,41.763,22227415,635.341351

Dumping output in a file

. . . instead of to the console. You can send outputs to an external txt file.

```
sink("Routput.txt")
rnorm(10)
plot(rnorm(10))     # only text, not plots
sink()      # switch back to console
```

Saving your R data objects: Save/Load Workspace

```r
save(X, y, file = "mystuff.RData")

save.image()  # saves current workspace into .RData

load("mystuff.RData")    # load into workspace
```

Same R objects as before.

Importing .xls, .xlsx data

1. In Excel copy data to clipboard, in R type ‘df <- read.table("clipboard", header = TRUE)‘. Not good, because not reproducible.
2. Save .xls file as a tab delimited .txt or .csv file, in R use one of

```
df <- read.table("file.txt", header = TRUE, sep = "\t")
df <- read.table("file.txt", header = TRUE, sep = ",")
df <- read.csv("file.csv")
```

Importing .xls, .xlsx data

3. Use the R package `xlsx`.

```
library(xlsx)

df <- read.xlsx("CO2.xlsx", sheetIndex = 1)
head(df)
dim(df)
```

Importing .xls, .xlsx data

4. Use the R package 'XLConnect'.

```
library(XLConnect) #install the XLConnect
```

```
## XLConnect 1.0.2 by Mirai Solutions GmbH [aut],
##   Martin Studer [cre],
##   The Apache Software Foundation [ctb, cph] (Apache POI),
##   Graph Builder [ctb, cph] (Curvesapi Java library)

## https://mirai-solutions.ch
## https://github.com/miraisolutions/xlconnect
```

```
df <- readWorksheetFromFile("CO2.xlsx", sheet = 1)
head(df)
```

```
##   Year Concentration
## 1 1000        277.00
## 2 1001        277.01
## 3 1002        277.02
## 4 1003        277.03
## 5 1004        277.04
```

Including path name (files in a different working directory)

- Know your working directory.
- . is for current working directory
- paste0 pastes 2 (or more) character strings into 1 string, without space

```
getwd()    # what is my working directory

filePath <- "~/Dropbox/TEACHING/
Honours StatisticalComputing/R/2017"

df2 <- readWorksheetFromFile(paste0(filePath,
                                    "/Data/CO2b.xlsx"),
                             sheet = 1)
head(df2)
dim(df2)
```

Including path name (files in a different working directory)

```
#getwd()   # what is my working directory

## relative to current working directory
df1 <- readWorksheetFromFile("./CO2b.xlsx",
                             sheet = 1)
head(df1)
```

```
##   Year Concentration
## 1 1000         277.00
## 2 1001         277.01
## 3 1002         277.02
## 4 1003         277.03
## 5 1004         277.04
## 6 1005         277.05
```

```
dim(df1)
```

```
## [1] 1013    2
```

Variable names

Sometimes you need to specify, or rename variables.

```
air <- read.table("air.txt")
head(air)
```

```
##   V1  V2   V3 V4 V5 V6
## 1 41 190  7.4 67  5  1
## 2 36 118  8.0 72  5  2
## 3 12 149 12.6 74  5  3
## 4 18 313 11.5 62  5  4
## 5 NA  NA 14.3 56  5  5
## 6 28  NA 14.9 66  5  6
```

Variable names

Sometimes you need to specify, or rename variables

```
vars <- names(airquality)     # or type in by hand
vars <- c("Ozone", "Solar.R", "Wind", "Temp", "Month",
          "Day")

names(air) <- vars
head(air)
```

```
##   Ozone Solar.R Wind Temp Month Day
## 1    41     190  7.4   67     5   1
## 2    36     118  8.0   72     5   2
## 3    12     149 12.6   74     5   3
## 4    18     313 11.5   62     5   4
## 5    NA      NA 14.3   56     5   5
## 6    28      NA 14.9   66     5   6
```

Data from data base (SQLite)

```r
library(DBI)

mydb <- dbConnect(RSQLite::SQLite(), "my-db.sqlite")
dbListTables(mydb)
```

```
## [1] "airquality"
```

```r
## Queries
dbGetQuery(mydb, 'SELECT * FROM airquality')
```

```
##     Ozone Solar.R Wind Temp Month Day
## 1      41     190  7.4   67     5   1
## 2      36     118  8.0   72     5   2
## 3      12     149 12.6   74     5   3
## 4      18     313 11.5   62     5   4
## 5      NA      NA 14.3   56     5   5
## 6      28      NA 14.9   66     5   6
## 7      23     299  8.6   65     5   7
## 8      19      99 13.8   59     5   8
## 9       8      19 20.1   61     5   9
## 10     NA     194  8.6   69     5  10
## 11      7      NA  6.9   74     5  11
```

Data from web

```
URL <- "http://earthquake.usgs.gov/earthquakes/feed/v1.0/su

#too long for the line!
#"http://earthquake.usgs.gov/earthquakes/feed/
#v1.0/summary/all_month.csv"

Earthquake_30Days <- read.table(URL, sep = ",",
                                header = T)
names(Earthquake_30Days)
```

```
##  [1] "time"            "latitude"        "longitude"
##  [5] "mag"             "magType"         "nst"
##  [9] "dmin"            "rms"             "net"
## [13] "updated"         "place"           "type"
## [17] "depthError"      "magError"        "magNst"
## [21] "locationSource"  "magSource"
```

```
#head(Earthquake_30Days)
```

Common data import problems

TIP: Don't change the files, use R to sort out any problems. Advantage: you will have code to document EXACTLY what you did.

1. Type conversions tries to guess data type of variable/column

as.is, colClasses in read functions.

2. special characters/separators (;, spaces)

- look at the data file

3. files with blanks instead of missing values

Prac

Check that you get all of the above to run, and update with your own comments.

Read the following data into R. Each time check that R has done the right thing. Leave the data as original as possible.

1. counts.xlsx (on Vula)
2. Tortoise data.xls, sheet 'Tortoise measurements' (on Vula)
3. Globular clusters:
   http://www.physics.mcmaster.ca/~harris/GCS_table.txt
4. Large-scale climatic index (MEI):
   https://www.esrl.noaa.gov/psd/enso/mei/table.html
5. Voice Data from Singing the Vowel 'ooh':
   http://www.statsci.org/data/general/ooh.txt
6. Air quality data: air2.dat (on Vula)
7. wader counts.xls (on Vula)