UNIT - 3

23-2-17

sqldf - package in R

An 'R' package for running SQL statements on data frames

```
# Load the package
library(sqldf)

Titanic      # data set in R
```

This dataset provides information on the fate of passengers on the fatal maiden voyage of the ocean liner 'Titanic', summarized according to economic status (class), sex, age & survival.

Format

A 4 dimensional array resulting from cross-tabulating 2201 observations on 4 variables. The variables & their levels are as follows

No	Name	Levels
1	class	1st, 2nd, 3rd, crew
2	sex	Male, Female
3	Age	Child, Adult
4	survived	No, yes.

Examples:

These examples show how to query a variety of data frame manipulations in R without SQL and then again with SQL.

```
warpbreaks                    # dataframe in R
   breaks wool tension
1      26    A       L
2      30    A       L
3      54    A       L
4      25    A       L
5      70    A       L
6      52    A       L

# head                        returns first 6 observations

alr <- head(warpbreaks)
als <- sqldf("select * from warpbreaks
              limit 6");

identical(alr, als)
[1] TRUE
```

outer function

```
a2r <- subset(CO2, grepl("^Qn", Plant))
a2s <- sqldf("select * from CO2 where
              Plant like 'Qn%'")
all.equal(as.data.frame(a2r), a2s)
[1] TRUE
```

```
CO2  # data frame
```

	Plant	Type	Treatment	conc	uptake
1	Qn1	Quebec	nonchilled	95	16.0
2	Qn1	Quebec	"	175	30.4
3	Qn1	"	"	250	34.8
4	Qn1	"	"	350	37.1
5	Qn1	"	"	500	35.3
6	Qn1	"	"	675	39.2
7	Qn1	"	"	1000	39.7
8	Qn2	"	"	95	13.6

③

data(fasms, package = "MASS")

a3r ← subset(fasms, Manag %in%
c("BF", "HF"))

a3s ← saldf("select * from fasms where
Manag in ('BF', 'HF')")

row.names(a3r) ← NULL

identical(a3r, a3s)

[1] TRUE

a3r ← subset(fasms, Manag %in%
c("BF", "HF"))

a3r

	Mois	Manag	Use	Manure
2	M1	BF	U2	C2
5	M1	HF	U1	C2
6	M1	HF	U2	C2
7	M1	HF	U3	C3
8	M5	HF	U3	C3
9	M4	HF	U1	C1
10	M2	BF	U1	C1
10	M1	BF	U3	C1

```
asr ← subset(farms, Mois = 'M1')
ass ← subset("select * from farms where
      saldf
                Mois = 'M1' ", row.names = TRUE
identical(asr, ass)
[1] FALSE
```

asr

	Mois	Manag	Use	Manure
1	M1	SF	U2	C4
2	M1	BF	U2	C2
5	M1	HF	U1	C2
6	M1	HF	U2	C2
7	M1	HF	U3	C3
11	M1	BF	U3	C1
18	M1	NM	U1	C0

should be identical but getting false

```
                                        Row.names = TRUE
aGr ← subset(farms, Mois = 'M2')
aGs ← sqldf("select * from farms where
        Mois = 'M2', Row.names = TRUE)
identical(aGr, aGs)
```

aGr

	Mois	Manag	Use	Manure
3	M2	SF	U2	C4
4	M2	SF	U2	C4
10	M2	BF	U1	C1
17	M2	NM	U1	C0

aGs

	Mois	Manag	Use	Manure
3	M2	SF	U2	C4
4	M2	SF	U2	C4
10	M2	BF	U1	C1
17	M2	NM	U1	C0

[1] FALSE ? should be identical

Rbind & Union all

a7r ← rbind(a5r, a6r)

a7s ← saldf("select * from ass Union all select * from a6s")

a7r

	Mois	Manag	Use	Maune
1	M1	SF	U2	C4
2	M1	BF	U2	C2
5	M1	HF	U1	C2
6	M1	Hf	U2	C2
7	M1	HF	U3	C3
11	M1	BF	U3	C3
18	M1	NM BF	U1	C0
3	M2	SF NM	U2	C4
4	M2	SF	U2	C4
10	M2	BF	U1	C4
17	M2	NM	U1	C0

a5r = 7
a6r = 4
} 11 rows retrieved

Note - sqldf drops the unused levels of Mois but rbind does not ; however,

all data is the same and the other columns are identical

```
row.names(a7r) ← NULL        # set it to NULL row names
identical (a7r[-1], a7s[-1])
[1] [TRUE] TRUE
```

Aggregate -

```
a8r ← aggregate [1:2], iris[5], mean)
a8s ← sqldf ('select species avg("sepal.
  length") 'sepal. Length',
avg("sepal. Width") 'sepal. Width' from
iris group by species')

all. equal (a8r, a8s)
  [TRUE]
```

a8r

	species	sepal. Length	sepal. Width
1	setosa	5.006	3.428
2	Vericolor	5.936	2.770
3	Virginica	6.588	2.974

ass <- sweep

ass

Species	sepal.Length	Sepal.width
setosa	5.006	3.428
versicolor	5.936	2.770
virginica	6.588	2.974

all.equal(~~ass, ass~~) (asr, ass)

[1] TRUE

[1] "Attributes: < component "row.names": Modes: character, numeric >"

[2] "Attributes: < component "row.names": target is character, current is numeric >"

by - avg conc and total uptake by Plant and Type

```
aqr ← do.call(rbind, by(iris, iris[5], function(x)
    with(x, data.frame(Species = Species[1],
        mean.Sepal.Length = mean(Sepal.Length),
        mean.Sepal.Width = mean(Sepal.Width),
        mean.Sepal.ratio = mean(Sepal.Length/Sepal.Width)))))

row.names(aqr) ← NULL

aqs ← sqldf('select Species, avg("Sepal.Length")
    'mean.Sepal.Length',
    avg("Sepal.Width") 'mean.Sepal.Width',
    avg("Sepal.Length"/"Sepal.Width")
    'mean.Sepal.ratio' From iris
    group by Species')

all.equal(aqr, aqs)
```

aqs

Species	mean.Sepal.Length	m.S.W	Sepal.ratio
setosa	5.006	3.428	1.470188
versicolor	5.936	2.770	2.160402
virginica	6.588	2.974	2.230453

Species	m.S.L	m.S.W	se
Setosa ↦ same	5.006	3.428	1.47
Versicolor "	5.936	2.770	2.16
Virginica "	6.588	2.974	2.23

all.equal (a9s, a9s)

"Attributes"

"Attributes"

Reading excel files in R

3 packages

```
install.packages("XLConnect")
install.packages("xlsx")
install.packages("gdata")
```

Using XLConnect

```
getwd()
```

```
setwd() → for setting the path of your file
```

```
install.packages("rJava")
```

```
any(grepl("rJava", installed.packages()))
[1] TRUE
```

```
any(grepl("methods", installed.packages()))
[1] TRUE
```

```
library("rJava")
```

Error: .onLoad failed in loadNamespace()
for 'rJava' details:
call: fun(libname, pkgname)
error: JAVA_HOME cannot be determined from the Registry

Error: package or namespace load failed for 'rJava'

```
Sys.setenv(JAVA_HOME = 'C:\\Program Files(x8
                \\Java\\jre7')

library(rJava)
```

Error: onLoad failed in loadNamespace() for 'rJava', details:

call: inDL(x, as.logical(local), as.logical(now), ...)

error: unable to load shared object

'C:/users/sridhar/Documents/R/win-library/3.3/rJava/libs/i386/rJava.dll':

LoadLibrary failure: The specified module could not be found

Error: package or namespace load failed for 'rJava'

Environment Variable
Path setting

```
Sys.setenv(JAVA_HOME='C:\\Program Files\\
    Java\\jre1.8.0_121')

library(rJava)      = no quotes
library("xlconnect")

wb <- loadWorkbook("simple.xlsx")
myDf <- readWorksheet(wb, sheet="sheet1",
                      header=TRUE)
```

Example 1:

mtcars .xlsx file from demoFiles subfolder of package XLConnect.

```
demoExcelFile ← system.file("demoFiles/mtcars.xlsx", package = "XLConnect")

# Load workbook
wb ← loadWorkbook(demoExcelFile)

# Read worksheet 'mtcars' (providing no specific area bounds;
# with default header: TRUE)

data ← readWorksheet(wb, sheet = "mtcars")
```

Example 2:

```
demoExcelFile ← system.file("demoFiles/mtcars.xlsx",
  Package = "XLConnect")

wb ← loadWorkbook(demoExcelFile)

data ← readWorksheet(wb, sheet = "mtcars",
    startRow = 2,  startCol = 3,
    endRow = 15,  endCol = 8)
```

Providing area bounds using the region argument, with default header = TRUE

```
data ← readWorksheet(wb, sheet = "mtcars",
                     region = "C1:H15")
```

Example 4

```
df ← readWorksheet(wb, sheet = "Conversion",
     header = TRUE,
  colTypes = c(XLC$DATA_TYPE.NUMERIC,
               XLC$DATA_TYPE.DATETIME,
               XLC$DATA_TYPE.BOOLEAN),
  forceConversion = TRUE,
  dateTimeFormat = "%Y-%m-%d %H:%M:
```

Read worksheet 'Conversion' with the Pre specified column types.

Note. in the worksheet all data was entered as strings.

forceConversion = TRUE is used to force conversion from String into less generic data types Numeric, DateTime & Boolean.

Example 5

```
wb ← loadWorkbook(demoExcelFile)

data ← readWorksheet(wb, sheet = "mtcars",
                     keep = c(1,3,5))

# Read the columns 1, 3 and 5 from the
  sheet 'mtcars' (with default header =
                  TRUE)
```

Eg.

```
excelFile ← system.file("demofiles/conversion.xlsx",
                        package = "XLConnect")

wb ← loadWorkbook(excelFile)
```

Package : gdata

```
read.xls(xls, sheet=1, verbose=FALSE, pattern,
  na.strings=c("NA", "#DIV/0!"),
  method=c("csv", "tsv", "tab"), perl="perl")
```

xls: Path to the Microsoft Excel file - supports "http://", "https://", and "ftp://" URLs.

sheet: name or number of the worksheet to read

verbose: logical flag indicating whether details should be printed as the file is processed.

pattern: if specified, then skip all lines before the first containing this string

perl: name of the perl executable to be called.

method: intermediate file format, "csv" for comma-separated and "tab" for tab-separated.

na.strings: a character vector of strings which are to be interpreted as 'NA' values.

blank.lines.skip: logical flag indicating whether blank lines in the original file should be ignored.

read.xls = returns a data frame

Note: Either a working version of Perl must be present in the executable search path or the exact path of the perl executable must be provided via the perl argume

Examples:

```
xlsfile ← file.path(path.package('gdata'),
          'xls', 'iris.xls')

# iris.xls is included in the gregmisc
  package for use as an example.

xlsfile

iris ← read.xls(xlsfile)  # defaults to csv
                            format
iris ← read.xls(xlsfile, method = "csv")
        # specify csv format
iris ← read.xls(xlsfile, method = "tab")
        # specify tab format
```

Example1:

```
mydf ← read.xls("simple.xlsx", sheet = 2,
                header = TRUE)
```

mydf

	sid	sname	marks	age
1	1	A	89	21
2	2	B	98	22
3	3	C	78	23
4	4	D	88	24
5	5	E	68	25

```
df
```

Example 2:

```
iris <- read.xls(xlsfile, perl = "C:/
                 Perl/bin/perl.exe")
```

Example specifying exact perl path for default MS-Windows install of ActiveState perl.

Unix systems:

```
iris <- read.xls(xlsfile, perl = "/usr/bin/
                 perl")
```

Finding Perl

```
Perl <- gdata:::findPerl("perl")
iris <- read.xls(xlsfile, perl = Perl)
```

read.xls automatically calls findPerl so this is rarely needed.)

read xls file from net

```
nba.url <- "http://mgtclass.mgt.unm.edu/BOSE/
  Excel/Tutorial.05/Cases/NBA.xls"

nba <- read.xls(nba.url)
```

read xls file ignoring all lines prior to first containing state.

```
crime.url <- "http://www.jrsainfo.org/jabg/
  state_data2/Tribal_Data00.xls"

crime <- read.xls(crime.url, pattern = "state")
```

use of xls2csv - open con, print two lines, close con.

```
con <- xls2csv(crime.url)
print(readLines(con, 2))
file.remove(summary(con)$description)
```

Examples demonstrating

```
examplefile ← file.path(path.package('gdata
    'xls', 'ExampleExcelFile.xls')

examplefile2007 ← file.path(path.package('gd
    'xls', 'ExampleExcelFile.xlsx')

# Examples demonstrating selecting of
  specific sheets from the example
  XLS file 'ExampleExcelFile.xls'

Eg: see the no: of names of sheets

sheetCount(examplefile)
if('XLSX' %in% xlsFormats())
sheetCount(examplefile2007)

# Names of the sheets

sheetNames(example file)
if('XLSX' %in% xlsformat())
sheetNames(examplefile2007)
```

Examples:

```
data <- read.xls(exampleFile)
# default is first worksheet
data <- read.xls(exampleFile, sheet = 2)
# second worksheet by no.
data <- read.xls(exampleFile, sheet = "sheet
                 second",
  V=TRUE)
  ## and by name

# load the third worksheet, skipping the
  first two non-data lines
if('XLSX' %in% XLSFormats())
  # if XLSX is supported
data <- read.xls(exampleFile2007, sheet = sheet
  with initial text", skip=2)

load a file containing data & column
  names using latin-1 characters

latinFile <- file.path(path.package('gdata'),
  'xls', 'latin-1.xls')
latin1 <- read.xls(latinFile, fileEncoding =
                   "latin1")
colnames(latin1)
```

data = read.xls("emp.xlsx")

data

eid	ename	dep
501	Sasitha	CSE
502	Hasitha	IT
503	Vanitha	CSE
504	Hari	CIVIL
505	Veena	MECH

df = read.xls("emp.xlsx", sheet = 1,
Perl = "C:/Perl/bin/Perl.exe")

df = read.xls("emp.xlsx",
Perl = "C://Perl//bin//Perl.exe")

same: data o/p

Perl = "C:/Perl/bin/Perl.exe")

df = read.xls("emp.xlsx", 1,
Perl = "")

df = read.xls("emp.xlsx", sheet = 1)

read.xlsx function

```
data1 <- read.xlsx("emp.xlsx")
```

Error in read.xlsx("emp.xlsx") :
please provide a sheet name or index

```
data1 <- read.xlsx("emp.xlsx", sheet = 1)
```

Error in
argument 2 matches multiple formal arguments.

```
data1 <- read.xlsx("emp.xlsx", sheet = "sheet1")
```

same error.

```
data1 <- read.xlsx("emp.xlsx", 1)
```

data1

eid	ename	dep
501	Saritha	CSE
502	Haritha	IT
503	Vanitha	CSE
504	Hari	CIVIL
505	Veena	MECH

Read R output in excel

```
head(mtcars)

ht ← head(mtcars)
write.table(ht, "mt.csv", sep = ",",
            header = TRUE)
```

Error: unused argument header = TRUE

```
write.table(ht, "mt.csv", sep = ",")
write.table(ht, "mt.csv", sep = ",",
            row.names = FALSE)
```

using csv file:-

```
write.csv(ht, "mt2.csv", row.names =
          FALSE)
```

Examples:-

```
read.csv("      ")

write.csv(x, file = "foo.csv", row.names
          = FALSE)

write.csv(x, file = "foo.csv")
```

Textbook example:-

```
x ← cbind(rnorm(20), runif(20))
colnames(x) ← c("A", "B")
 x
write.table(x, file = "myfile.csv", sep = ",",
            row.names = FALSE)
```

```
a <- 1:10
b <- 10:1
cbind(a,b)
   a  b
1  1 10
2  2  9
3  3  8
```

```
write.csv(cbind(a,b), file = "mycolumn1.csv"
                      row.names = FALSE)
```

```
write.csv(cbind(a,b), file = mycolumn1.csv")
```