Intro_R_2.5_2.6

HDS, Mike Kozlowski

July 24, 2018

updated 1/31/2023

Material Related to File and Workspace management

##Workspace

Your variable exist in the R workspace, in RStudio, you can click on the Environment tab and see what is defined in your workspace. Functions you create will appear here, as well as the data you have been working with. This is your workspace

You can see this in the console using the ls() command

you can remove things from the workspace using the rm() or remove() commands

```
y=1:10
y
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
rm(y)
#y
```

You can remove everything from your workspace using the rm(list=ls()) command- there will be no warning, variables will simply be removed.

You can save the entire workspace to disk, when you try to exit R or RStudio, it will ask if you want to do this. This can be helpful if you need to stop in the middle of a long calculation. Under the “session” tab you can load and save workspaces, which are all the variables and functions in use

Getting Data in and Out

We’ve talked about using read.table and read.csv to import data sets, we’ll talk a bit more about other types of general file I/O and importing JSON, XML, etc later, as well as connecting to databases and to APIs

R has a native data format .RDATA, you can save data in the RDATA format using the save function

Before we save, let’s check the location of the current working directory, where files will be saved to.

You can do this using the “Session” menu in RStudio, or by using the getwd() command. I could then set the working directory to my desktop using setwd(“/Users/hsheets/Desktop”)- alter the name of the directory in setwd() below to set it to your desktop- Hint: you can use file.choose() runing in your console window to figure out the correct file path to your own desktop

```
getwd()
```

```
## [1] "C:/Users/Mike/Documents/DAT511/2-1 class"
```

```
setwd("/Users/Mike/Desktop")
getwd()
```

```
## [1] "C:/Users/Mike/Desktop"
```

The default of the save function is to save into the working directory

Note you can also specify a full file path in the filename,

save(mydata, file="/Users/hsheets/Desktop/mydata.RData")

```
mydata=matrix(c(1,2,3,4),nrow=2,byrow=TRUE)
mydata
```

```
##      [,1] [,2]
## [1,]    1    2
## [2,]    3    4
```

```
save(mydata,file="mydata.RData")
```

We can then load this file, using the load command, if the file is in the current working directory, we don't have to specify the path name.

We could also use file.choose() here, obviously.

```
mydata=0
mydata
```

```
## [1] 0
```

```
load("mydata.RData")
mydata
```

```
##      [,1] [,2]
## [1,]    1    2
## [2,]    3    4
```

A couple of things to note:

save() can be used to save more than one R variable at a time, the data is saved along with all the class and structure information and the variable names- so this is a save of a portion of the working space of R, not simply the values of the variables.

When you load() an RData file, you are altering the workspace, loading all the information about the data you saved. Note that no variable is assigned to the result of the load() function, but the mydata variable has

obviously been changed.

The save.image(file="something.RData") command will save the entire contents of your workspace, all the variables plus all the functions. This can also be done using the "Session" menu in RStudio. When you exit R or RStudio, you will be asked if you want to save the workspace, by default it will be loaded the next time you start R or RStudio. There is a default .RData file used for this.

You can save the workspace image under any name you chose, I often add a time or date stamp to the name, this allows you to come back to an analysis at a later time if you need to.

The workspace image can be reloaded using the load("INSERT_FILENAME.RData") command

###Miscellaneous functions related to the file system

dir()- lists the contents of the current directory x=dir() is a handy why of loading the directory contnents into a list of strings, x, which you can then search, count, etc, etc

file.path()- merges a path name and a file name, but you can do this manually anyway

file.info()- gets basic info about a file

file.exists()- checks to see if a file exists- could be used to avoid a crash

Try tinkering with these commands a bit, just to see what use you might make of them.

##Specialized data files

There are many specialized file formats in the world, and many R packages meant to load them.

I generally don't use a lot of these, I tend to try to export data from other program as comma separated value files (.csv).

In some cases, though, the complexity of the data files is such that it make more sense to use a package to do the job- particularly in working with GIS files, for example.

van der Loo and de Jonge mention some of these packages such as "foreign" that load and says data for other statistical packages such as Minitab, SPSS, SAS,Stata, Systat, dBase etc.

the readxl package can read Excel files, in the xtml format that Excel uses. This approach can be helpful for some really fussy databases.

###Reading from DataBases

There are several package available that allow you to send queries and other data base commands from R to a database, and to retrieve the results of the queries or other commands.

You must have the access rights to the database and the server the database is located on. If I can get access to a simple database on campus, we will look at this.

We can access database files from SQLite3 easily enough. This is a simple database system that doesn' require a username and password system, we can take a quick look at it.

Make sure the package "RSLite" and "DBI" are installed

Set up the libraries

```
require(DBI)
```

```
## Loading required package: DBI
```

```
require(RSQLite)
```

```
## Loading required package: RSQLite
```

The first step in loading from a database is to make a connection to the database, that we pass commands from R to the database with, and which the database uses to send data back

SQLite has a very simple connector, with no usernames or passwords, setting up most database connectors requires a password

We are connecting to a SQLite database called “chinook” that is the database for a fictional online media company

```
con<-dbConnect(SQLite(),"C:\\Users\\Mike\\Documents\\DAT511\\2-1 class\\chinook.db")
```

This is a SQL database, it has data organized into tables, which are organized like data.frames are in R

We will first get a list of all the tables in this database, and put them into an R dataframe

```
as.data.frame(dbListTables(con))
```

```
##    dbListTables(con)
## 1    ManagementRoles
## 2             albums
## 3            artists
## 4          customers
## 5          employees
## 6             genres
## 7      invoice_items
## 8           invoices
## 9        media_types
## 10    playlist_track
## 11         playlists
## 12             ranks
## 13   sqlite_sequence
## 14      sqlite_stat1
## 15            tracks
```

Now we can load an entire table from the database into R to work with it in R

```
my_artists=dbReadTable(con,"artists")
```

```
head(my_artists)
```

```
##   ArtistId                 Name
## 1        1                AC/DC
## 2        2               Accept
## 3        3            Aerosmith
## 4        4    Alanis Morissette
## 5        5      Alice In Chains
## 6        6 Antônio Carlos Jobim
```

We can look at the columns (fields) in a table

```
as.data.frame(dbListFields(con,"employees"))
```

```
##    dbListFields(con, "employees")
## 1                      EmployeeId
## 2                        LastName
## 3                       FirstName
## 4                           Title
## 5                       ReportsTo
## 6                       BirthDate
## 7                        HireDate
## 8                         Address
## 9                            City
## 10                          State
## 11                        Country
## 12                     PostalCode
## 13                          Phone
## 14                            Fax
## 15                          Email
```

We can send SQL (sequential query language commands) from R to the database

It is helpful to use SQL to select only what we want from the database, which can save us time and memory space in R when working with a big database

```
# we send the query

rs=(dbSendQuery(con,"SELECT FirstName,Title FROM employees"))

#then fetch the result

as.data.frame(fetch(rs))
```

```
##   FirstName               Title
## 1    Andrew     General Manager
## 2     Nancy       Sales Manager
## 3      Jane Sales Support Agent
## 4  Margaret Sales Support Agent
## 5     Steve Sales Support Agent
## 6   Michael          IT Manager
## 7    Robert            IT Staff
## 8     Laura            IT Staff
```

When we are done working with a database, we need to close the connection

```
dbDisconnect(con)
```

```
## Warning in connection_release(conn@ptr): There are 1 result in use. The
## connection will be released when they are closed
```

In summary, to access a database from within R

a.) Load the R libraries needed b.) Set up the connection, which may need a username and password c.) Use the connection to -See what is in the database -Upload whole tables -Run SQL commands- these can store data into a database as well d.) Disconnect from the database

###Obtaining Data Via APIs

API- Application Programming Interface- many website provide APIs in different languages to allow you to pull data of the website (from the underlying data base or data files) from within your code.

Here will will use the RSocrata package to access data on OpenDataBuffalo.

It turns out that Socrata supplies the web services for many OpenData projects for different cities, including Chicago. Socrata has APIs for many languages

```
require("RSocrata")
```

```
## Loading required package: RSocrata
```

We need an address from a data set at open data buffalo

We can get the public art data set for Buffalo

I got this by going to open data buffalo, finding the public art data set and then using the API button to get the address of the data

```
api_target="https://data.buffalony.gov/resource/6xz2-syui.csv"

artdf=read.socrata(api_target)

head(artdf)
```

```
##                            title           category       type
## 1    ISOCHRONIC MOUNTAIN BUFFALO          SCULPTURE FIGURATIVE
## 2              BUFFALO STREET MAP           GRAPHICS        MAP
## 3 WAR MEMORIAL AUDITORIUM CHAIRS  DECORATIVE OBJECT  FURNITURE
## 4                         CHARITY           PAINTING      MURAL
## 5         MAYOR ANTHONY MASIELLO          PAINTINGS   PORTRAIT
## 6                      CITY SEAL DECORATIVE OBJECTS     PLAQUE
##                                 medium frame
## 1                              CERAMIC false
## 2               CLOTH MAP IN WOOD  CASE  true
## 3                         WOODEN CHAIRS false
## 4 OIL ON CANVAS WITH HIGH RELIEF PLASTER false
## 5                          OIL ON CANVAS  true
## 6                        BRONZE ON WOOD false
##                                                                                                photo_url_link
## 1                                                                                                     UNKNOWN
## 2                                                                                                     UNKNOWN
## 3                                                                                                     UNKNOWN
## 4 HTTP://WWW.CI.BUFFALO.NY.US/FILES/1_2_1/PUBLIC%20ART%20WEBSITE/WEB%20PAGES/CHARITY.HTML
## 5                                                                                                     UNKNOWN
## 6                                                                                                     UNKNOWN
##                     artist contractor founder architect carver designer
## 1            JOSHUA G STEIN
## 2                  UNKNOWN
## 3                  UNKNOWN
## 4 WILLIAM DE LEFTWICH DODGE
## 5           NATHAN NAETZKER
## 6                  UNKNOWN
##         date object_height_in object_width_in object_depth_in
## 1 2019-06-04               NA              NA              NA
## 2 1928-01-01               NA              NA              NA
## 3       <NA>               NA              NA              NA
## 4 1931-01-01              108             132              NA
## 5       <NA>               NA              NA              NA
## 6       <NA>               NA              NA              NA
##   object_diameter_in base_height_in base_width_in base_depth_in
## 1                 NA             NA            NA            NA
## 2                 NA             NA            NA            NA
## 3                 NA             NA            NA            NA
## 4                 NA             NA            NA            NA
## 5                 NA             NA            NA            NA
## 6                 NA             NA            NA            NA
##   base_diameter_in plaque_inscription                site    street_address
## 1               NA                    28TH FLOOR CITY HALL 65 NIAGARA SQUARE
## 2               NA                     1ST FLOOR CITY HALL 65 NIAGARA SQUARE
## 3               NA                           201 CITY HALL 65 NIAGARA SQUARE
## 4               NA                     1ST FLOOR CITY HALL 65 NIAGARA SQUARE
## 5               NA                           201 CITY HALL 65 NIAGARA SQUARE
## 6               NA                           201 CITY HALL 65 NIAGARA SQUARE
##      city zip_code state latitude longitude         geocoded_column
## 1 BUFFALO    14202    NY 42.88664 -78.87935 (42.886641, -78.879354)
```

```
## 2 BUFFALO    14202    NY 42.88664 -78.87935 (42.886641, -78.879354)
## 3 BUFFALO    14202    NY 42.88664 -78.87935 (42.886641, -78.879354)
## 4 BUFFALO    14202    NY 42.88664 -78.87935 (42.886641, -78.879354)
## 5 BUFFALO    14202    NY 42.88664 -78.87935 (42.886641, -78.879354)
## 6 BUFFALO    14202    NY 42.88664 -78.87935 (42.886641, -78.879354)
```

##“Raw” file I/O

There are general purpose file I/O routines in R, that will let you load or save pretty much any format you want by writing your own code. This will be covered in a different RMD file.

If we want to get a quick look at a text file, to see what is in it, we can use a readline function to read one line at a time

We first create a file connection to the file (just like a connection to a database)

We are opening the file to read (“r”) text (“t)

```
infile="C:\\Users\\Mike\\Documents\\DAT511\\2-1 class\\2020-2021_Assessment_Roll.csv"

filecon=file(infile,"rt")

# Read the first 10 lines and store them in temp

temp=readLines(filecon, n=10)
```

we can then print successive lines of the file, this lets us see delimiters, for example

```
temp[1]
```

```
## [1] "SBL,TAX DISTRICT,PRINT KEY,FRONT,DEPTH,PROPERTY CLASS,PROP CLASS DESCRIPTION,PREVIOUS
PROPERTY CLASS,OWNER1,OWNER2,PREVIOUS OWNER,MAIL1,MAIL2,MAIL3,MAIL4,HOUSE NUMBER,STREET,ADDRE
SS,CITY,STATE,ZIP CODE (5-DIGIT),ZIP CODE (4-DIGIT),DEED BOOK,DEED PAGE,DEED DATE,ROLL,LAND V
ALUE,TOTAL VALUE,SALE PRICE,YEAR BUILT,FIRST STORY AREA,SECOND STORY AREA,TOTAL LIVING AREA,O
VERALL CONDITION,BUILDING STYLE,HEAT TYPE,BASEMENT TYPE,# OF STORIES,# OF FIREPLACES,# OF BED
S,# OF BATHS,# OF KITCHENS,COUNCIL DISTRICT,POLICE DISTRICT,CENSUS TRACT,CENSUS BLOCK GROUP,C
ENSUS BLOCK,NEIGHBORHOOD,LATITUDE,LONGITUDE,LOCATION"
```

```
temp[2]
```

```
## [1] "1008100010016000,147003,100.81-10-16,30,100,311,RESIDENTIAL VACANT LAND,311,BACKEY QU
INTELLA,,,,,167 ORANGE ST,\"BUFFALO, NY\",144,PEACH,144 PEACH,BUFFALO,NY,14204,NA,9041,102,09
/09/9999 12:00:00 AM,1,4000,4000,0,,0,0,0,NA,NA,NA,NA,0,0,0,0,0,ELLICOTT,District B,31003,3,3
017,Fruit Belt,42.8991088078,-78.8576067349,POINT (-78.8576067349 42.8991088078)"
```

Close the connection to the file

```
close(filecon)
```

Question/Exercise

Open Excel, create a small data table, save it as an excel file (not a csv)

Import readxl

use the read_excel(filename) function to load the excel file

you will need to use google to figure out how to do this

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

```
library(readxl)
read_excel("C:\\Users\\Mike\\Documents\\DAT511\\2-1 class\\Book1.xlsx")
```

```
## # A tibble: 1 × 3
##    test test2 test3
##   <dbl> <dbl> <dbl>
## 1     3     4     5
```