Getting data

Xijin Ge and Tom Brandenburg @ SDSU

We will demonstrate how to read data from keyboard, files, API, and SQL databases.

Required data files: <http://statland.org/R/RC/heartatk4R.txt>

Required R libraries: *jsonlite,* *ggplot2, ggmap,* and *rentrez*. These can be installed from **Packages 🡪 Install Packages** from main menu of R. Or by running this commands: **install.packages(“jsonlite”)**

Required software: text editor software like Notepad++, and Microsoft Excel

# **A: Enter data manually**

**x = c( 2.1, 3.1, 3.2, 5.4)**

**sum(x)**

**x = scan ( )** # Enter values separated by Return key. Can paste a column of numbers from Excel.

**A = matrix( c(2, 4, 3, 1, 5, 7),** # the data elements

**nrow=2,** # number of rows

**ncol=3 )** # number of columns

**A**  # show the matrix

[,1] [,2] [,3]

[1,] 2 3 5

[2,] 4 1 7

# **B: Read data from files**

General procedure:

1. Compressed? Unzip using 7-zip, WinRAR, Winzip, gzip. Any of these will do.
2. Text file (CSV, txt, …) or Binary file (XLS, XLSX, …)? Convert binary to text file using corresponding application. Comma separated values (CSV) files, use comma to separate the columns. Another common type is tab-delimitated text files, which uses the tab or \t as it is invisible character.
3. Open with text editor (TexPad, NotePad++) to have a look.
   1. Rows and columns? Row and column names? **row.names=1 header=T**
   2. Deliminaters?(space, comma, tab…) **sep=“\t”**
   3. Missing values? NA, na, NULL, blank, NaN, 0 **missingstring=**
4. Optional: Open as text file in Excel, choose appropriate deliminater while importing, or use the **Text to Column** under **Data**. Beware of the annoying automatic conversion in Excel “OCT4” 🡪 “4-OCT”. Edit column names, save as CSV, or Tab-deliminated text file for reading in R.
5. Change working directory to where the file was saved. Main menu: **File 🡪 Change dir…**
6. read.table ( ), or read.csv( )

**x = read.table(“somefile.txt”, sep=“\t”, header=TRUE, missingstring=“NA”)**

1. Double check the data with **fix(),** click on the column name to make sure each column is recognized correctly as “**character**” or “**numeric**”.

## **Activity:**

1. Install Notepad++
2. Download file from <http://statland.org/R/RC/heartatk4R.txt> and save it locally (**Downloads** folder)
3. Open data file in NotePad++ to have a look at the data to examine the data. Open the file in Excel. It is tab-delimited text file. Do we see rows and columns? Column headers, Row names? Missing values?
4. In R, change working directory to where the file was saved. For example it could be C:/users/John.Smith/Downloads. From R main menu: **File 🡪 Change dir…**
5. Read the data, your folder can be different from C:/users/John.Smith/Downloads.

**x = read.table("heartatk4R.txt",header=TRUE)**

Since this file is available online, you can actually access it directly from the internet:

**x = read.table("http://statland.org/R/RC/heartatk4R.txt", header=TRUE)**

**fix(x)** # make sure CHARGES is numeric.

**head(x)** # take a look at the first few lines

**attach(x)** # attach the heartatk dataset so that the columns are directly accessible by name.

**hist(age)**

# **C: Read data directly from database using SQL (structural query language)**

SQL databases are behind most of web servers. It is possible to access directly from R.

**library(DBI)**

**library(RMySQL)**

**m <- dbDriver("MySQL");**

**con <- dbConnect(m, user='mcm', password='welcome323', host='omega', dbname='yeast');**

**res <- dbSendQuery(con, "select \* from yeast\_genes\_07")**

Note that this is the SQL query and it can be very complicated with many different combinations and conditions: **select \* from yeast\_genes\_07 where …**

# **D: Reading data directly from internet via API**

Application program interface (**API**) is a set of routine definitions, protocols, and tools for building software and applications. It enables computers to send data to each other over the internet. Data is often sent in formats like **JSON** (JavaScript Object Notation) or **XML** (Extensible Markup Language). Many web servers provide free API access.

## **B1. Bike Sharing in New York City**

A public API that shows location, status and current availability for all stations in the New York City bike sharing imitative. Visit this page to see more: <https://www.citibikenyc.com/>

**bike <- fromJSON("http://citibikenyc.com/stations/json")**

**stations <- bike$stationBeanList**

**colnames(stations)**

**head(stations)**

**hist(stations$totalDocs)**

We can see there is location and status info for 507 stations, including location code for each. Below we will plot a map with the locations highlighted in red. We use the *ggmap* package to show them on a map.

**install.packages("ggplot2") # an extensive R plotting system**

**install.packages("ggmap") # for plotting maps**

**library(ggplot2)**

**library(ggmap)**

**qmplot( longitude,latitude, data = stations, colour =I('red'), darken =0.3)**

We can see that these stations are strategically placed across the NYC. We can also retrieve google maps, and display location-specific information on it by adding layers.

**ggmap ( get\_googlemap("Brookings, SD") )**

## **B2. Articles and movie reviews at New York Times**



The New York Times has several APIs as part of the NYT developer network. These interface to data from various departments, such as news articles, book reviews, real estate, etc. Registration is required (but free) and a key can be obtained at [here](http://developer.nytimes.com/signup). The code below includes some example keys for illustration purposes.

*#movie reviews*

**key <- "&api-key=5a3daaeee6bbc6b9df16284bc575e5ba:0:68700045"**

**url <- "http://api.nytimes.com/svc/movies/v2/reviews/dvd-picks.json?order=by-date"**

**req <- fromJSON(paste0(url, key))**

**reviews <- req$results**

**colnames(reviews)**

**reviews[1:10, c("display\_title", "byline", "mpaa\_rating")]**

display\_title byline mpaa\_rating

1 Clown GLENN KENNY R

2 Bang Gang: A Modern Love Story STEPHEN HOLDEN

3 Cosmos A. O. SCOTT

4 Tickled MANOHLA DARGIS R

5 Raiders: The Story of the Greatest Fan Film Ever Made MANOHLA DARGIS

## **B3: Public legislation data**

The Sunlight Foundation is a non-profit that helps to make government transparent and accountable through data, tools, policy and journalism. Register a free key at [here](http://sunlightfoundation.com/api/accounts/register/). An example key is provided.

**key <- "&apikey=39c83d5a4acc42be993ee637e2e4ba3d"**

***#Find bills about drones***

**drone\_bills <- fromJSON(paste0("http://openstates.org/api/v1/bills/?q=drone", key))**

**drone\_bills$title <- substring(drone\_bills$title, 1, 40)**

**print(drone\_bills[1:5, c("title", "state", "chamber", "type")])**

title state chamber type

1 Relates to prohibiting civilian drone us ny upper bill

2 Relates to prohibiting civilian drone us ny lower bill

3 relative to the use of drones. nh lower bill

4 Omnibus supplemental budget bill. mn lower bill

5 Relates to conducting unlawful surveilla ny lower bill

There are many others, including twitter API. See this web site:

<https://cran.r-project.org/web/packages/jsonlite/vignettes/json-apis.html>

## **B4: Retrieve biological data from National Institutes of Health (NIH)**

Entrez is probably the largest repository of biological data. It is maintained by National Center for Biotechnology Information (NCBI), a part of NIH. These data are accessible through a web interface <http://www.ncbi.nlm.nih.gov/> . In addition, the databases can be accessed programmatically. For this example, we install a special R package called *Rentrez*.

**install.packages("rentrez")**

**library(rentrez)**

**entrez\_dbs( ) # list of all databases**

[1] "pubmed" "protein" "nuccore" "nucleotide" "nucgss" "nucest" "structure" "genome"

[9] "annotinfo" "assembly" "bioproject" "biosample" "blastdbinfo" "books" "cdd" "clinvar" ……

PubMed is a database for biomedical literature, with 26 million citations. Try it at [www.PubMed.gov](http://www.PubMed.gov) and search for “breast cancer” related literature. Note the number of papers.

Using rentrez package we can access this database through its API programmatically. We can search for published articles related to certain keyword:

**pubmed <- entrez\_search( db="pubmed", term="breast cancer", retmax=100)**

**pumed$count** # number of articles returned

**pubmed$ids** # extract PubMed IDs, each corresponds to a paper

**summ <- entrez\_summary(db="pubmed", id=pmid[1])**  # retrieve more info about a paper

**summ$title** # title of first article

As you can see above, the object returned by entrez\_search() includes the number of records matching a given search. This means you can learn a little about the composition of, or trends in, the records stored in the NCBI’s databases using only the search utility. For instance, let’s track the rise of the scientific research after the ebola outbreak, programmatically creating search terms:

# first define a function that runs the query and returns number of articles

**search\_year <- function(year, term) {**

**query <- paste(term, "AND (", year, "[PDAT])")**

**entrez\_search(db="pubmed", term=query, retmax=0)$count**

**}**

**year <- 2008:2016** # define time range

**papers <- sapply(year, search\_year, term="ebola", USE.NAMES=FALSE)**

**plot(year, papers, type='b', main="Ebola")**

We have a trend of the number of papers on this topic. We can see that scientific papers shoot up at 2014 and 2015, after the outbreak. As a comparison, we can go to Google Trends web site [www.Google.com/trends](http://www.Google.com/trends) to get a plot on internet search frequencies from the general public.

|  |  |
| --- | --- |
|  |  |

Figure . Left: Medical research on Ebola virus surges in 2014. Over 2000 papers are published in 2015. Right: Google searches on Ebola peaks and recedes quickly following the outbreak.

Activity: Change the search term and time interval in the example above to study the trend in medical research.