Working with external data

Introduction to Data Visualization

Table of Contents

1 Overview

Introduction to some useful tools to get external data not yet loaded in one of the many R packages, into our R session, as well as saving results to external data files that can be used by others.

2 External data and R

- Data used in an interactive R session is stored in RAM and is volatile (disappears at `quit`)
- External data can be read from or written to with `read.table`
- Data rendered in HTML on web pages can be extracted

3 File management in R

- All functions to read external data assume these files exist in our *working directory*

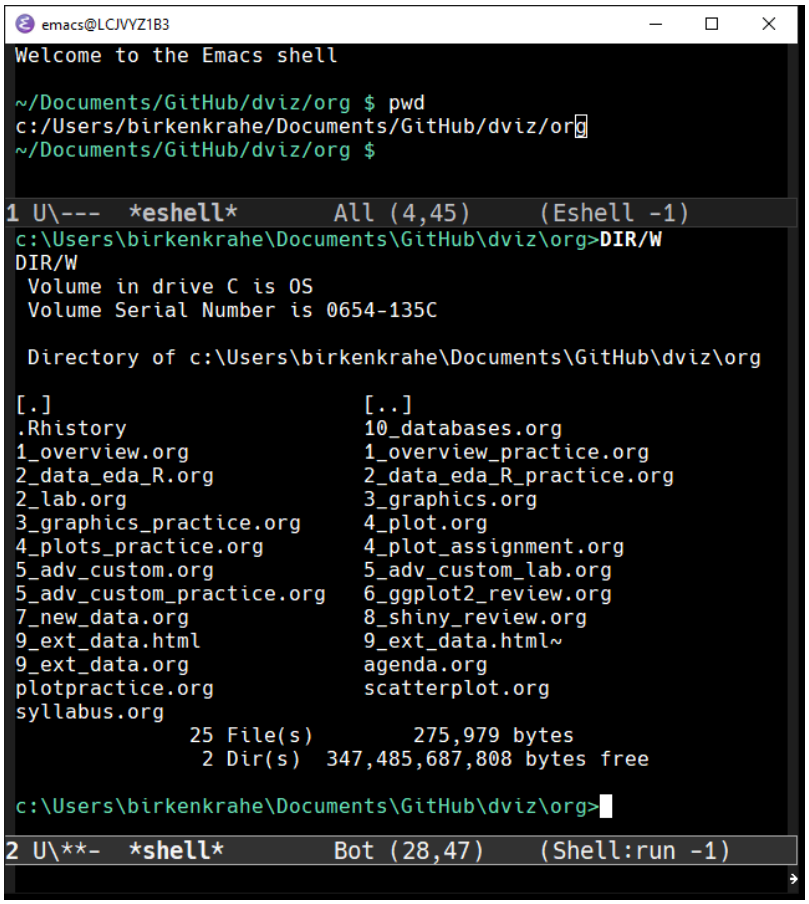
```
getwd()  # return working directory
```

```
[1] "C:/Users/birkenkrahe"
```

- To code along, open your R console `*R*` in an Emacs window

The slash

- Notice the use of `/` in the `PATH` by R (Linux, MacOS, Emacs) vs. `\` by Windows
- Where are you? Use `M-x shell` (`DIR/W`) and `M-x eshell` (`pwd`)



Set working directory

- `setwd("d")` changes the current working directory to `d`
- If `d` does not exist, an error is returned
- `dir.create` can create a directory for you: `dir.create('test')`
- `shell` executes shell (Windows OS) commands: `shell('DIR/W')`

Relative and absolute path

- Path (re-) direction is OS navigation
- Relative path: `./` is here, `../` goes up
- Absolute path: start with disk location `C:/`

o* NEXT Practice: navigation

1. In Emacs, change to `~/` and open an R shell
2. Print working directory files: `shell('DIR/W')`
3. Get current working directory with `getwd`
4. Get current directory (Windows: `CD`) with `shell`
5. Set working directory to `..` (one level up)
6. Check current working directory with `getwd`, `shell`
7. Set working directory to `C:/`
8. Check current working directory with `getwd`, `shell`
9. Return to the original working directory `~/`
10. Create a new directory `navtest` with `dir.create`
11. Create an Org-file `navtest.org` that creates new R session `*R2*`
12. In the code block, save `plot(Nile)` to the file `nile1.png` in `navtest` using the relative path to `navtest`
13. In another code block, change to ~ and save `hist(Nile)` to the file `nile2.png` in `navtest` using the absolute path to `navtest`
14. Print content of `navtest` with `shell` from the R shell

1-8

```
setwd('~')      # go home
                                        #  shell('DIR/W')  # print directory content
getwd()
shell('CD')
setwd('..')
getwd()
shell('CD')
setwd('c:/')
getwd()
shell('CD')
```

```
[1] "C:/Users/birkenkrahe"
C:\Users\birkenkrahe
[1] "C:/Users"
C:\Users
[1] "c:/"
c:\
```

9-10

```
setwd('~')
getwd()
shell('CD')
dir.create('navtest')
getwd()
shell('CD')
```

```
[1] "C:/Users/birkenkrahe"
C:\Users\birkenkrahe
Warning message:
```

```
In dir.create("navtest") : 'navtest' already exists
[1] "C:/Users/birkenkrahe"
C:\Users\birkenkrahe
```

11-14

```
setwd('~')
plot(Nile)
```

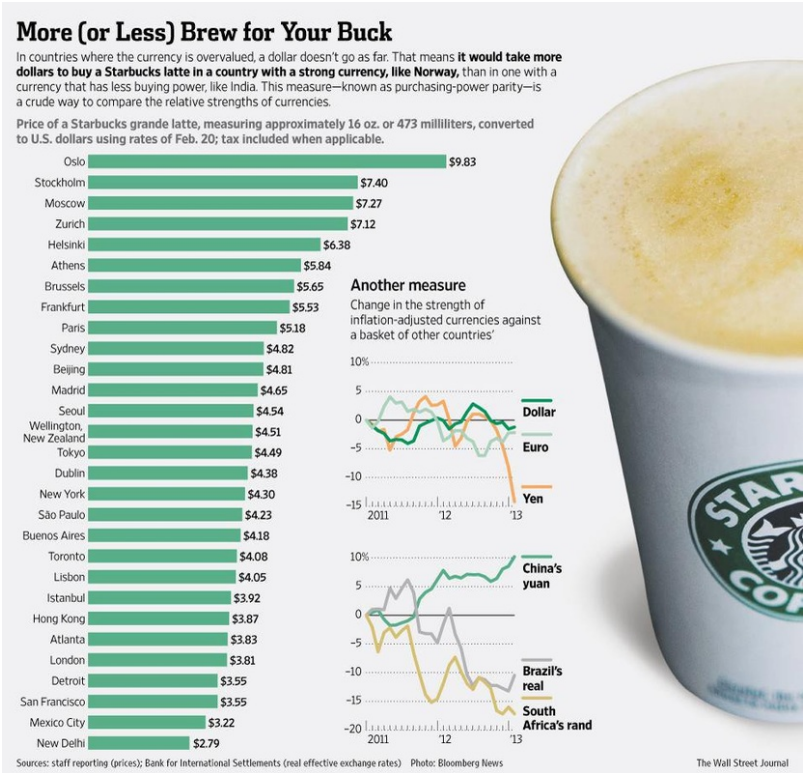
nile1.png

```
shell('CD')
shell('DIR/W navtest')
```

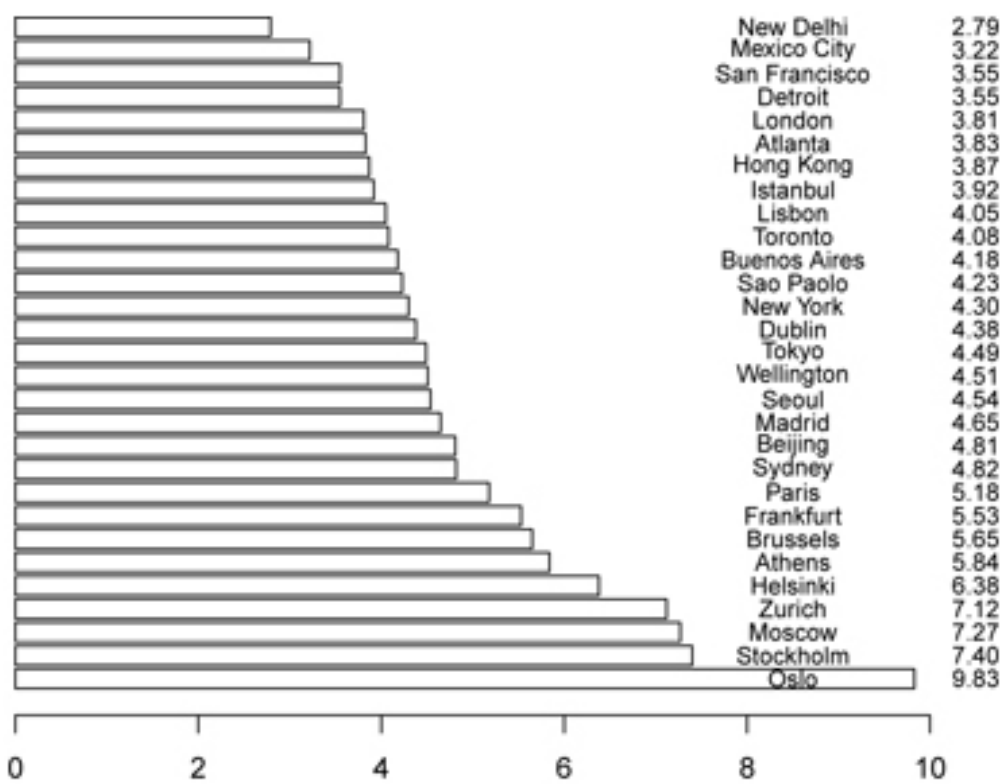
```
C:\Users\birkenkrahe
 Volume in drive C is OS
 Volume Serial Number is 0654-135C

 Directory of C:\Users\birkenkrahe\navtest

[.]         [..]        nile1.png   nile2.png   nile3.png
               3 File(s)         12,522 bytes
               2 Dir(s)  345,621,860,352 bytes free
```

4 Manual data entry

- Manual data entry is tedious and error-prone and should be avoided
- You have to do it e.g. when data are embedded in images (source)



- Example: City names and Grande Latte prices via link (Wall Street Journal and Bloomberg News on Feb 27, 2013).



- Constructing the data frame begins with a one-row data frame.
  - `fix` pops up a spreadsheet-style data entry window
  - `write.csv` writes the data to a CSV file
  - Enter the next row (`Stockholm 7.40 Sweden`) and close the app

```
LatteIndexFrame <- data.frame(
  city = "Oslo",
  price = 9.83,
  country = "Norway")
fix(LatteIndexFrame)
write.csv(x = LatteIndexFrame,
          file = "../data/LatteIndexFrame.csv",
          row.names=FALSE)
```

- Show the dataframe in R and also print it from the file using the Windows shell command `notepad [file]`

```
LatteIndexFrame
shell("notepad ../data/LatteIndexFrame.csv")
```

```
  city price country
1 Oslo  9.83  Norway
```

5 Interacting with the Internet

- An examples of Internet data download and use: text data!
- The Internet is a potentially volatile data source: web pages can be changed, moved, taken down - apply good citation habits!

6 Getting the data set

- An automobile gas mileage data set is available from Univ of Calif at Irvine's Machine Learning Repository - used for benchmarking ML algorithms.

- You can open a browser to a URL (Universal Resource Locator) with `browseURL` (you have to manually navigate back to R or Emacs):

```
browseURL("http://archive.ics.uci.edu/ml")
```

  1. Open the dataset list on the page with `*View ALL Data Sets*`
  2. Open the `Data Folder` belonging to `Auto MPG`
  3. Copy link address of `auto-mpg.data` (mouse right click)

- To get the data:

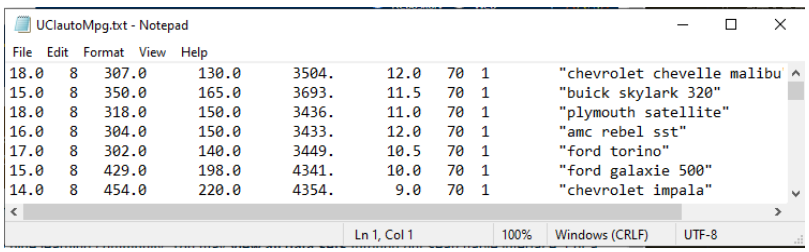
  1. feed the copied URL to `download.file`
  2. check that the file was downloaded with `shell`

```
shell('DEL UCIautoMpg.txt')
URL <- "http://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data"
download.file(URL, "UCIautoMpg.txt")
shell('DIR/W UCIautoMpg.txt')
```

```
trying URL 'http://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.dat
Content type 'application/x-httpd-php' length 30286 bytes (29 KB)
downloaded 29 KB
 Volume in drive C is OS
 Volume Serial Number is 0654-135C

 Directory of C:\Users\birkenkrahe

UCIautoMpg.txt
               1 File(s)         30,684 bytes
               0 Dir(s)  345,621,594,112 bytes free
```

- View the file using the Windows `notepad` app:

```
shell('notepad UCIautoMpg.txt')
```



7 Reading the data records

- The format of the dataset requires us to develop a *parsing* strategy to extract the fields we want from each record
- The function for reading text data (including HTML) is `readLines`. It reads the extracted data into a `character` vector

- To see what the raw data format looks like, we analyze the first record of the data set using

  1. `readLine` to read the data into R
  2. `head` to show a couple of lines
  3. `nchar` to count characters,
  4. `substr` to split record in two components

```
autoMpgRecords <- readLines("UCIautoMpg.txt")
head(autoMpgRecords,1)
x <- autoMpgRecords[1]  # store first record in x
nchar(x)
substr(x, start=1, stop=56) # numbers
substr(x, start=57, stop=84) # text
```

```
[1] "18.0   8   307.0      130.0      3504.      12.0   70  1\t\"chevrolet chevelle malibu
[1] 84
```

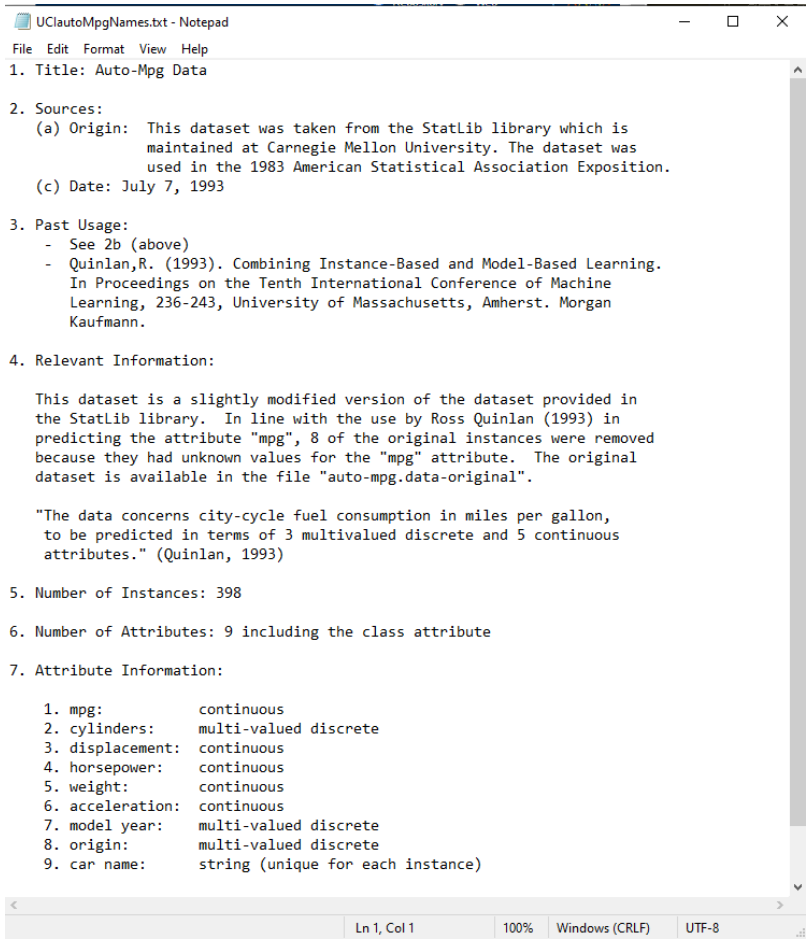
```
[1] "18.0   8   307.0      130.0      3504.      12.0   70  1"
[1] "\t\"chevrolet chevelle malibu\""
```

8 Reading the data variable names

- The names of the variables of our data set are stored separately
- Obtain it using the `download.file` function on the resp. URL and check that the file exists with `shell` - when opening this as a string without `DIR`, Windows brings up the Notepad app automatically

```
URL1 <- "http://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.names"
download.file(URL1, "UCIautoMpgNames.txt")
shell("DIR/W UCIautoMpgNames.txt")
```

```
trying URL 'http://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.nam
Content type 'application/x-httpd-php' length 1660 bytes
downloaded 1660 bytes
 Volume in drive C is OS
 Volume Serial Number is 0654-135C

 Directory of C:\Users\birkenkrahe

UCIautoMpgNames.txt
               1 File(s)          1,705 bytes
               0 Dir(s)  345,621,590,016 bytes free
```



- View the file with `notepad` by passing the filename only to `shell`

```
shell("UCIautoMpgNames.txt")
```

- Read the text file with `readLines`. Lines 32 to 44 of this file contain the the variable names we want:

```
autoMpgNames <- readLines("UCIautoMpgNames.txt")
autoMpgNames[32:44]
```

```
 [1] "7. Attribute Information:"
 [2] ""
 [3] "    1. mpg:           continuous"
 [4] "    2. cylinders:     multi-valued discrete"
 [5] "    3. displacement:  continuous"
 [6] "    4. horsepower:    continuous"
 [7] "    5. weight:        continuous"
 [8] "    6. acceleration:  continuous"
 [9] "    7. model year:    multi-valued discrete"
[10] "    8. origin:        multi-valued discrete"
[11] "    9. car name:      string (unique for each instance)"
[12] ""
[13] "8. Missing Attribute Values:  horsepower has 6 missing values"
```

9 Creating the data frame

- To turn the *raw* data records into an R data frame:
  1. Remove internal quotation marks from every record with `gsub`
  2. Split element on the tab character with `strsplit` into character vectors with the numerical variables and the car names
  3. Split the numerical variable vector on any occurrence of whitespace giving a character vector with eight elements (one for each variable in the data record) again using `strsplit`
  4. Convert the eight elements to numeric variables with `as.numeric`
  5. Combine resulting variables and car names into a data frame

- All these commands are contained in the function `ConvertAutoMpgRecords`. It is saved as a file with that name. You can load it into your R session after downloading the file:

```
URL2 <- "https://tinyurl.com/yc65uauv"
download.file(URL2, "ConvertAutoMpgRecords")
load(file="ConvertAutoMpgRecords")
autoMpgFrame <- ConvertAutoMpgRecords(autoMpgRecords)
head(autoMpgFrame)
```

```
trying URL 'https://tinyurl.com/yc65uauv'
downloaded 72 KB
Error in load(file = "ConvertAutoMpgRecords") :
  bad restore file magic number (file may be corrupted) -- no data loaded
In addition: Warning message:
file 'ConvertAutoMpgRecords' has magic number '<!DOC'
  Use of save versions prior to 2 is deprecated
Warning messages:
1: In ConvertAutoMpgRecords(autoMpgRecords) : NAs introduced by coercion
2: In ConvertAutoMpgRecords(autoMpgRecords) : NAs introduced by coercion
3: In ConvertAutoMpgRecords(autoMpgRecords) : NAs introduced by coercion
4: In ConvertAutoMpgRecords(autoMpgRecords) : NAs introduced by coercion
5: In ConvertAutoMpgRecords(autoMpgRecords) : NAs introduced by coercion
6: In ConvertAutoMpgRecords(autoMpgRecords) : NAs introduced by coercion
  mpg cylinders displacement horsepower weight acceleration modelYear origin
1  18         8          307        130   3504         12.0        70      1
2  15         8          350        165   3693         11.5        70      1
3  18         8          318        150   3436         11.0        70      1
4  16         8          304        150   3433         12.0        70      1
5  17         8          302        140   3449         10.5        70      1
6  15         8          429        198   4341         10.0        70      1
                    carName
1 chevrolet chevelle malibu
2         buick skylark 320
3        plymouth satellite
4             amc rebel sst
5               ford torino
6          ford galaxie 500
```

- The function definition and saving to file:

```
ConvertAutoMpgRecords <- function(rawRecords) {
  noQuotes <- gsub('\"', '', rawRecords)
  n <- length(noQuotes)
  outFrame <- NULL
  for (i in 1:n) {
    x <- noQuotes[i]
```

```
    twoParts <- unlist(strsplit(x,split="\t"))
    partOne <- unlist(strsplit(twoParts[1],split="[ ]+"))
    numbers <- as.numeric(partOne)
    upFrame <- data.frame(mpg = numbers[1],
                          cylinders = numbers[2],
                          displacement = numbers[3],
                          horsepower = numbers[4],
                          weight = numbers[5],
                          acceleration = numbers[6],
                          modelYear = numbers[7],
                          origin = numbers[8],
                          carName = twoParts[2])
    outFrame <- rbind.data.frame(outFrame, upFrame)
  }
  return(outFrame)
}
save(ConvertAutoMpgRecords, file="ConvertAutoMpgRecords")
```

10 References

- Pearson RK (2016). Exploratory Data Analysis. CRC Press.

Author: Marcus Birkenkrahe
Created: 2022-11-20 Sun 15:49