Data Input and Output with R

Armand Tossou

9/4/2021

CSV Input and Output

CSV stands for 'comma-separated values'.

```
# create a sample CSV file using the built-in 'mtcars' dataset
write.csv(mtcars, file = 'my_example.csv')

# reading in a .csv file
ex <- read.csv('my_example.csv')

# preview the file
head(ex)
```

```
##                   X  mpg cyl disp  hp drat    wt  qsec vs am gear carb
## 1         Mazda RX4 21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## 2     Mazda RX4 Wag 21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## 3        Datsun 710 22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## 4    Hornet 4 Drive 21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## 5 Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## 6           Valiant 18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

```
tail(ex)
```

```
##                 X  mpg cyl  disp  hp drat    wt qsec vs am gear carb
## 27  Porsche 914-2 26.0   4 120.3  91 4.43 2.140 16.7  0  1    5    2
## 28   Lotus Europa 30.4   4  95.1 113 3.77 1.513 16.9  1  1    5    2
## 29 Ford Pantera L 15.8   8 351.0 264 4.22 3.170 14.5  0  1    5    4
## 30   Ferrari Dino 19.7   6 145.0 175 3.62 2.770 15.5  0  1    5    6
## 31  Maserati Bora 15.0   8 301.0 335 3.54 3.570 14.6  0  1    5    8
## 32     Volvo 142E 21.4   4 121.0 109 4.11 2.780 18.6  1  1    4    2
```

Check system info

There is a new library called readxl and writexl which are really easy to install and use, you can check them out here: https://readxl.tidyverse.org/

The older library shown in the video requires Java to be installed, also with the RJava library.

```r
# Check whether R is 32/64 bit with sessionInfo(). Check Platform.
sessionInfo()
```

```
## R version 4.1.1 (2021-08-10)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19043)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United States.1252
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## loaded via a namespace (and not attached):
##  [1] compiler_4.1.1  magrittr_2.0.1  fastmap_1.1.0   tools_4.1.1
##  [5] htmltools_0.5.2 yaml_2.2.1      stringi_1.7.4   rmarkdown_2.10
##  [9] knitr_1.33      stringr_1.4.0   xfun_0.25       digest_0.6.27
## [13] rlang_0.4.11    evaluate_0.14
```

```r
# Download the specific 32/64 bit of Java. This is really important. R and Java must have the same memo

# Download Java JDK for 32/64 bit. For 64-bit I had to download the Windows version from here: https://

# Install rJava with:

install.packages("rJava")
```

```
## Installing package into 'C:/Users/adtos/Documents/R/win-library/4.1'
## (as 'lib' is unspecified)

## Error in contrib.url(repos, "source"): trying to use CRAN without setting a mirror
```

```r
# Set your JAVA_HOME environment with Sys.setenv(JAVA_HOME="C:/Program Files/Java/jdk-10.0.1/") so that
# Note on JDK (Java SE Development Kit): For Java Developers. Includes a complete JRE plus tools for de

Sys.setenv(JAVA_HOME="C:/Program Files/Java/jdk-10.0.1/")

# check for errors after running this

library(rJava)
```

```
## java.home option:

## JAVA_HOME environment variable: C:/Program Files/Java/jdk-10.0.1/

## Warning in fun(libname, pkgname): Java home setting is INVALID, it will be ignored.
## Please do NOT set it unless you want to override system settings.
```

Excel Files with R

We'll install packages that allow us to work with Excel within 'R' and 'Rstudio'.

```
# install the 'readxl' library

install.packages("readxl")
```

```
## Installing package into 'C:/Users/adtos/Documents/R/win-library/4.1'
## (as 'lib' is unspecified)
```

```
## Error in contrib.url(repos, "source"): trying to use CRAN without setting a mirror
```

```
# load the package
library(readxl)


# import a sample sales data Excel file
## Note: the excel file was downloaded from Kaggle at this address: https://www.kaggle.com/kyanyoga/sam
# we manually split its 25 columns into 2 sheets, with 'ORDERNUMBER' as the unique identifier
excel_sheets('Sample_Sales_Data_Kaggle.xlsx')
```

```
## [1] "Sheet1" "Sheet2"
```

```
df <- read_excel('Sample_Sales_Data_Kaggle.xlsx', sheet = 'Sheet1')
head(df)
```

```
## # A tibble: 6 x 13
##   ORDERNUMBER QUANTITYORDERED PRICEEACH ORDERLINENUMBER SALES
##         <dbl>           <dbl>     <dbl>           <dbl> <dbl>
## 1       10107              30      95.7               2 2871
## 2       10121              34      81.4               5 2766.
## 3       10134              41      94.7               2 3884.
## 4       10145              45      83.3               6 3747.
## 5       10159              49     100                14 5205.
## 6       10168              36      96.7               1 3480.
## # ... with 8 more variables: ORDERDATE <dttm>, STATUS <chr>, QTR_ID <dbl>,
## #   MONTH_ID <dbl>, YEAR_ID <dbl>, PRODUCTLINE <chr>, MSRP <dbl>,
## #   PRODUCTCODE <chr>
```

```
# check total sales
sum(df$MSRP)
```

```
## [1] 284320
```

```
# summary statistics
summary(df)
```

```
##   ORDERNUMBER    QUANTITYORDERED   PRICEEACH      ORDERLINENUMBER
##  Min.   :10100   Min.   : 6.00   Min.   : 26.88   Min.   : 1.000
```

```
##  1st Qu.:10180   1st Qu.:27.00   1st Qu.: 68.86   1st Qu.: 3.000
##  Median :10262   Median :35.00   Median : 95.70   Median : 6.000
##  Mean   :10259   Mean   :35.09   Mean   : 83.66   Mean   : 6.466
##  3rd Qu.:10334   3rd Qu.:43.00   3rd Qu.:100.00   3rd Qu.: 9.000
##  Max.   :10425   Max.   :97.00   Max.   :100.00   Max.   :18.000
##      SALES           ORDERDATE                     STATUS
##  Min.   :  482.1   Min.   :2003-01-06 00:00:00   Length:2823
##  1st Qu.: 2203.4   1st Qu.:2003-11-06 12:00:00   Class :character
##  Median : 3184.8   Median :2004-06-15 00:00:00   Mode  :character
##  Mean   : 3553.9   Mean   :2004-05-11 00:16:49
##  3rd Qu.: 4508.0   3rd Qu.:2004-11-17 12:00:00
##  Max.   :14082.8   Max.   :2005-05-31 00:00:00
##      QTR_ID         MONTH_ID         YEAR_ID     PRODUCTLINE
##  Min.   :1.000   Min.   : 1.000   Min.   :2003   Length:2823
##  1st Qu.:2.000   1st Qu.: 4.000   1st Qu.:2003   Class :character
##  Median :3.000   Median : 8.000   Median :2004   Mode  :character
##  Mean   :2.718   Mean   : 7.092   Mean   :2004
##  3rd Qu.:4.000   3rd Qu.:11.000   3rd Qu.:2004
##  Max.   :4.000   Max.   :12.000   Max.   :2005
##       MSRP       PRODUCTCODE
##  Min.   : 33.0   Length:2823
##  1st Qu.: 68.0   Class :character
##  Median : 99.0   Mode  :character
##  Mean   :100.7
##  3rd Qu.:124.0
##  Max.   :214.0
```

```
## Let's download the entire workbook into R, with both of its sheets

entire.workbook <- lapply(excel_sheets('Sample_Sales_Data_Kaggle.xlsx'), read_excel, path='Sample_Sales_

head(entire.workbook)
```

```
## [[1]]
## # A tibble: 2,823 x 13
##    ORDERNUMBER QUANTITYORDERED PRICEEACH ORDERLINENUMBER SALES
##          <dbl>           <dbl>     <dbl>           <dbl> <dbl>
##  1       10107              30      95.7               2 2871
##  2       10121              34      81.4               5 2766.
##  3       10134              41      94.7               2 3884.
##  4       10145              45      83.3               6 3747.
##  5       10159              49     100                14 5205.
##  6       10168              36      96.7               1 3480.
##  7       10180              29      86.1               9 2498.
##  8       10188              48     100                 1 5512.
##  9       10201              22      98.6               2 2169.
## 10       10211              41     100                14 4708.
## # ... with 2,813 more rows, and 8 more variables: ORDERDATE <dttm>,
## #   STATUS <chr>, QTR_ID <dbl>, MONTH_ID <dbl>, YEAR_ID <dbl>,
## #   PRODUCTLINE <chr>, MSRP <dbl>, PRODUCTCODE <chr>
##
## [[2]]
## # A tibble: 2,823 x 13
##    ORDERNUMBER CUSTOMERNAME    PHONE   ADDRESSLINE1    ADDRESSLINE2 CITY   STATE
```

```
##          <dbl> <chr>           <chr>   <chr>           <chr>        <chr>  <chr>
##  1       10107 Land of Toys I~ 212555~ 897 Long Airpo~ <NA>         NYC    NY
##  2       10121 Reims Collecta~ 26.47.~ 59 rue de l'Ab~ <NA>         Reims  <NA>
##  3       10134 Lyon Souveniers +33 1 ~ 27 rue du Colo~ <NA>         Paris  <NA>
##  4       10145 Toys4GrownUps.~ 626555~ 78934 Hillside~ <NA>         Pasad~ CA
##  5       10159 Corporate Gift~ 650555~ 7734 Strong St. <NA>         San F~ CA
##  6       10168 Technics Store~ 650555~ 9408 Furth Cir~ <NA>         Burli~ CA
##  7       10180 Daedalus Desig~ 20.16.~ 184, chausse d~ <NA>         Lille  <NA>
##  8       10188 Herkku Gifts    +47 22~ Drammen 121, P~ <NA>         Bergen <NA>
##  9       10201 Mini Wheels Co. 650555~ 5557 North Pen~ <NA>         San F~ CA
## 10       10211 Auto Canal Pet~ (1) 47~ 25, rue Lauris~ <NA>         Paris  <NA>
## # ... with 2,813 more rows, and 6 more variables: POSTALCODE <chr>,
## #   COUNTRY <chr>, TERRITORY <chr>, CONTACTLASTNAME <chr>,
## #   CONTACTFIRSTNAME <chr>, DEALSIZE <chr>
```

```
entire.workbook
```

```
## [[1]]
## # A tibble: 2,823 x 13
##    ORDERNUMBER QUANTITYORDERED PRICEEACH ORDERLINENUMBER SALES
##          <dbl>           <dbl>     <dbl>           <dbl> <dbl>
##  1       10107              30      95.7               2 2871
##  2       10121              34      81.4               5 2766.
##  3       10134              41      94.7               2 3884.
##  4       10145              45      83.3               6 3747.
##  5       10159              49     100                14 5205.
##  6       10168              36      96.7               1 3480.
##  7       10180              29      86.1               9 2498.
##  8       10188              48     100                 1 5512.
##  9       10201              22      98.6               2 2169.
## 10       10211              41     100                14 4708.
## # ... with 2,813 more rows, and 8 more variables: ORDERDATE <dttm>,
## #   STATUS <chr>, QTR_ID <dbl>, MONTH_ID <dbl>, YEAR_ID <dbl>,
## #   PRODUCTLINE <chr>, MSRP <dbl>, PRODUCTCODE <chr>
##
## [[2]]
## # A tibble: 2,823 x 13
##    ORDERNUMBER CUSTOMERNAME    PHONE   ADDRESSLINE1    ADDRESSLINE2 CITY   STATE
##          <dbl> <chr>           <chr>   <chr>           <chr>        <chr>  <chr>
##  1       10107 Land of Toys I~ 212555~ 897 Long Airpo~ <NA>         NYC    NY
##  2       10121 Reims Collecta~ 26.47.~ 59 rue de l'Ab~ <NA>         Reims  <NA>
##  3       10134 Lyon Souveniers +33 1 ~ 27 rue du Colo~ <NA>         Paris  <NA>
##  4       10145 Toys4GrownUps.~ 626555~ 78934 Hillside~ <NA>         Pasad~ CA
##  5       10159 Corporate Gift~ 650555~ 7734 Strong St. <NA>         San F~ CA
##  6       10168 Technics Store~ 650555~ 9408 Furth Cir~ <NA>         Burli~ CA
##  7       10180 Daedalus Desig~ 20.16.~ 184, chausse d~ <NA>         Lille  <NA>
##  8       10188 Herkku Gifts    +47 22~ Drammen 121, P~ <NA>         Bergen <NA>
##  9       10201 Mini Wheels Co. 650555~ 5557 North Pen~ <NA>         San F~ CA
## 10       10211 Auto Canal Pet~ (1) 47~ 25, rue Lauris~ <NA>         Paris  <NA>
## # ... with 2,813 more rows, and 6 more variables: POSTALCODE <chr>,
## #   COUNTRY <chr>, TERRITORY <chr>, CONTACTLASTNAME <chr>,
## #   CONTACTFIRSTNAME <chr>, DEALSIZE <chr>
```

```
## writing to Excel files

# we first need to install the 'xlsx' package
install.packages('xlsx')
```

```
## Installing package into 'C:/Users/adtos/Documents/R/win-library/4.1'
## (as 'lib' is unspecified)
```

```
## Error in contrib.url(repos, "source"): trying to use CRAN without setting a mirror
```

```
library(xlsx) # call the library

# let's write the built-in 'mtcars' dataframe to Excel
head(mtcars)
```

```
##                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

```
write.xlsx(mtcars, "output_example.xlsx")
```

SQL with R

This will actually be a brief lecture, because connecting R to a SQL database is completely dependent on the type of database you are using (MYSQL, Oracle, etc.).

So instead of trying to cover all of these (since each requires a different package), we'll use this lecture to point you in the right direction for various database types. Once you've downloaded the correct library, actually connecting is usually quite simple. It's then just a matter of passing SQL queries through R.

We'll show a general version using the `DBI` package, then point to more specific resources.

RODBC - General Use

The RODBC (https://cran.r-project.org/web/packages/RODBC/RODBC.pdf) library is one way of connecting to databases. Regardless of what you decide to use, I highly recommend a Google search consisting of 'your database of choice + R' . Here's an example use of RODBC.

```
# install the RODBC package
install.packages("RODBC")
```

```
## Installing package into 'C:/Users/adtos/Documents/R/win-library/4.1'
## (as 'lib' is unspecified)
```

```
## Error in contrib.url(repos, "source"): trying to use CRAN without setting a mirror
```

```r
# RODBC example of syntax
library(RODBC)

myconn <- odbcConnect("Database_name", uid="User_ID", pws="password")
```

```
## Error in RODBC::odbcDriverConnect("DSN=Database_name;UID=User_ID", pws = "password"): unused argumen
```

```r
dat <- sqlFetch(myconn, "Table_Name")
```

```
## Error in odbcValidChannel(channel): object 'myconn' not found
```

```r
querydat <- sqlQuery(myconn, "SELECT * FROM table")
```

```
## Error in odbcValidChannel(channel): object 'myconn' not found
```

MySQL

The RMySQL (https://cran.r-project.org/web/packages/RMySQL/index.html) package provides an interface to MySQL.

Oracle

The ROracle (https://cran.r-project.org/web/packages/ROracle/index.html) package provides an interface for Oracle.

JDBC

The RJDBC (https://cran.r-project.org/web/packages/RJDBC/index.html) package provides access to databases through a JDBC interface.

PostgreSQL

The RPostgreSQL (https://cran.r-project.org/web/packages/RPostgreSQL/index.html) package provides access to databases through a PostgreSQL interface..

Again, Google is the best way to go for your personal situation, since databases and your permissions can differ a lot!

For instance, you can check out the website http://www.r-bloggers.com for tutorials. Here's an example on how to connect to SQL through RPostgreSQL: https://www.r-bloggers.com/2015/05/getting-started-with-postgresql-in-r/.

Web Scraping with R

NOTE: TO FULLY UNDERSTAND THIS LECTURE, YOU WILL NEED TO KNOW HTML AND CSS. YOU WILL ALSO NEED TO KNOW THE PIPE OPERATOR IN R (%>%). COME BACK TO THIS LECTURE AFTER REVIEWING THAT MATERIAL.

Web Scraping in general is almost always going to be unique to your personal use case. This is because every website is different. Updates occur and things can change. To fully understand webscraping in R, you'll need to understand HTML and CSS in order to know what you are trying to grab off the website.

If you don't know HTML or CSS, you may be able to use an auto-web-scrape tool, like import.io (https://www.import.io/). Check it out; it will auto scrape and create a .csv file for you.

`rvest` library

Below is a simple example of using `rvest`, but the best way to see a good demo of `rvest` is through the built-in demos by using:

```
demo(package = 'rvest')
```

Now, if you are familiar with HTML and CSS, a very useful library is `rvest`. Below, we'll go over a simple example from RStudio:

```
# Will also install dependencies
install.packages('rvest')
```

```
## Installing package into 'C:/Users/adtos/Documents/R/win-library/4.1'
## (as 'lib' is unspecified)
```

```
## Error in contrib.url(repos, "source"): trying to use CRAN without setting a mirror
```

Imagine we'd like to scrape some information about The Lego Movie from IMDB. We start by downloading and parsing the file with `html()`:

```
library(rvest)
lego_movie <- read_html("http://www.imdb.com/title/tt1490017/")
```

To extract the rating, we start with `SelectGadget` to figure out which CSS selector matches the data we want: strong span. If you haven't heard of `SelectGadget`, make sure to read about it at https://selectorgadget.com/. It's the easiest way to determine which selector extracts the data that you're interested in.

We use `html_node()` to find the first node that matches that selector, extract its contents with `html_text()`, and convert it to numeric with `as.numeric()`:

```
lego_movie %>%
  html_node("strong span") %>%
  html_text() %>%
  as.numeric()
```

```
## [1] NA
```

We use a similar process to extract the cast, using `html_nodes()` to find all nodes that match the selector.

```
lego_movie %>%
  html_node("#titleCast .itemprop span") %>%
  html_text()
```

```
## [1] NA
```

Alright, hopefully this lecture gives you some good resources and ideas in case you want to webscrape with R in the future!