intermediate-importing-data

Teodor Chakarov

2022-03-24

Contents

Tutorium in R

Intermediate Importing data with R

Exercise Intermediate to Importing Data in R - Number 2

By: Teodor Chakarov 12141198

Initialize MySQL

con is MySql connection object

```
library(DBI)
perl<- "C:\\Users\\tedoc\\OneDrive\\Dokumente\\R\\win-library\\4.1\\rtools42\\usr\\bin\\perl5.32.1.exe"

con <- dbConnect(RMySQL::MySQL(),
                 dbname = "tweater",
                 host = "courses.csrrinzqubik.us-east-1.rds.amazonaws.com",
                 port = 3306,
                 user = "student",
                 password = "datacamp")

tables <- dbListTables(con)

str(tables)  # view all tables
```

```
##  chr [1:3] "comments" "tweats" "users"
```

To import user table we use:

```
users <- dbReadTable(con, "users")
print(users)
```

```
##   id      name     login
## 1  1 elisabeth  elismith
```

```
## 2  2      mike     mikey
## 3  3      thea   teatime
## 4  4    thomas tomatotom
## 5  5    oliver olivander
## 6  6      kate  katebenn
## 7  7    anjali    lianja
```

To import all tables at once:

```
table_names <- dbListTables(con)
tables <- lapply(table_names, dbReadTable, conn = con)
print(table_names)
```

```
## [1] "comments" "tweats"   "users"
```

```
print(tables)
```

```
## [[1]]
##      id tweat_id user_id            message
## 1  1022       87       7              nice!
## 2  1000       77       7             great!
## 3  1011       49       5            love it
## 4  1012       87       1   awesome! thanks!
## 5  1010       88       6              yuck!
## 6  1026       77       4      not my thing!
## 7  1004       49       1  this is fabulous!
## 8  1030       75       6           so easy!
## 9  1025       88       2             oh yes
## 10 1007       49       3           serious?
## 11 1020       77       1 couldn't be better
## 12 1014       77       1       saved my day
##
## [[2]]
##   id user_id
## 1 75       3
## 2 88       4
## 3 77       6
## 4 87       5
## 5 49       1
## 6 24       7
##                                                                post
## 1                                      break egg. bake egg. eat egg.
## 2                          wash strawberries. add ice. blend. enjoy.
## 3                       2 slices of bread. add cheese. grill. heaven.
## 4                open and crush avocado. add shrimps. perfect starter.
## 5 nachos. add tomato sauce, minced meat and cheese. oven for 10 mins.
## 6                                 just eat an apple. simply and healthy.
##         date
## 1 2015-09-05
## 2 2015-09-14
## 3 2015-09-21
## 4 2015-09-22
## 5 2015-09-22
## 6 2015-09-24
##
## [[3]]
```

```
##   id      name     login
## 1  1 elisabeth  elismith
## 2  2      mike     mikey
## 3  3      thea   teatime
## 4  4    thomas tomatotom
## 5  5    oliver olivander
## 6  6      kate  katebenn
## 7  7    anjali    lianja
```

Queries in MySQL

Import tweat_id column of comments where user_id is 1: elisabeth

```
elisabeth <- dbGetQuery(con, "SELECT tweat_id FROM comments WHERE user_id = 1")
print(elisabeth)
```

```
##   tweat_id
## 1       87
## 2       49
## 3       77
## 4       77
```

Import tweets latest than 2015-09-21

```
latest <- dbGetQuery(con, "SELECT post FROM tweats WHERE date > '2015-09-21'")
print(latest)
```

```
##                                                                  post
## 1               open and crush avocado. add shrimps. perfect starter.
## 2 nachos. add tomato sauce, minced meat and cheese. oven for 10 mins.
## 3                             just eat an apple. simply and healthy.
```

Select specific information of an object

```
specific <- dbGetQuery(con, "Select message From comments Where tweat_id = 77 AND user_id > 4")
print(specific)
```

```
##   message
## 1  great!
```

Check len of chars with CHAR_LENGTH()

```
short <- dbGetQuery(con, "Select id, name From users Where CHAR_LENGTH(name)< 5")
print(short)
```

```
##   id name
## 1  2 mike
## 2  3 thea
## 3  6 kate
```

Fetching the data

```
res <- dbSendQuery(con, "SELECT * FROM comments WHERE user_id > 4")
print(res)
```

```
## <MySQLResult:334651504,0,10>
```

```
dbFetch(res, n = 2)
```

```
##     id tweat_id user_id message
## 1 1022       87       7   nice!
```

```
## 2 1000       77       7  great!
```

```
dbFetch(res)
```

```
##     id tweat_id user_id  message
## 1 1011       49       5  love it
## 2 1010       88       6    yuck!
## 3 1030       75       6 so easy!
```

```
print(res)
```

```
## <MySQLResult:334651504,0,10>
```

```
#dbClearResult(res)
```

HTTP requests

Get the data via HTTP GET request

```
library("readr")

# Import the csv file: pools
url_csv <- "http://s3.amazonaws.com/assets.datacamp.com/production/course_1478/datasets/swimming_pools.
pools <- read_csv(url_csv)
```

```
## Rows: 20 Columns: 4
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): Name, Address
## dbl (2): Latitude, Longitude
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# Import the txt file: potatoes
url_delim <- "http://s3.amazonaws.com/assets.datacamp.com/production/course_1478/datasets/potatoes.txt"
potatoes <- read_tsv(url_delim)
```

```
## Rows: 160 Columns: 8
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## dbl (8): area, temp, size, storage, method, texture, flavor, moistness
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
print(pools)
```

```
## # A tibble: 20 x 4
##    Name                                Address          Latitude Longitude
##    <chr>                               <chr>               <dbl>     <dbl>
##  1 Acacia Ridge Leisure Centre         1391 Beaudesert~    -27.6      153.
##  2 Bellbowrie Pool                     Sugarwood Stree~    -27.6      153.
##  3 Carole Park                         Cnr Boundary Ro~    -27.6      153.
##  4 Centenary Pool (inner City)         400 Gregory Ter~    -27.5      153.
##  5 Chermside Pool                      375 Hamilton Ro~    -27.4      153.
##  6 Colmslie Pool (Morningside)         400 Lytton Road~    -27.5      153.
##  7 Spring Hill Baths (inner City)      14 Torrington S~    -27.5      153.
```

```
##  8 Dunlop Park Pool (Corinda)                794 Oxley Road,~    -27.5      153.
##  9 Fortitude Valley Pool                     432 Wickham Str~    -27.5      153.
## 10 Hibiscus Sports Complex (upper MtGravatt) 90 Klumpp Road,~    -27.6      153.
## 11 Ithaca Pool ( Paddington)                 131 Caxton Stre~    -27.5      153.
## 12 Jindalee Pool                             11 Yallambee Ro~    -27.5      153.
## 13 Manly Pool                                1 Fairlead Cres~    -27.5      153.
## 14 Mt Gravatt East Aquatic Centre            Cnr wecker Road~    -27.5      153.
## 15 Musgrave Park Pool (South Brisbane)       100 Edmonstone ~    -27.5      153.
## 16 Newmarket Pool                            71 Alderson Str~    -27.4      153.
## 17 Runcorn Pool                              37 Bonemill Roa~    -27.6      153.
## 18 Sandgate Pool                             231 Flinders Pa~    -27.3      153.
## 19 Langlands Parks Pool (Stones Corner)      5 Panitya Stree~    -27.5      153.
## 20 Yeronga Park Pool                         81 School Road,~    -27.5      153.
```

```
print(potatoes)
```

```
## # A tibble: 160 x 8
##     area  temp  size storage method texture flavor moistness
##    <dbl> <dbl> <dbl>   <dbl>  <dbl>   <dbl>  <dbl>     <dbl>
##  1     1     1     1       1      1     2.9    3.2       3
##  2     1     1     1       1      2     2.3    2.5       2.6
##  3     1     1     1       1      3     2.5    2.8       2.8
##  4     1     1     1       1      4     2.1    2.9       2.4
##  5     1     1     1       1      5     1.9    2.8       2.2
##  6     1     1     1       2      1     1.8    3         1.7
##  7     1     1     1       2      2     2.6    3.1       2.4
##  8     1     1     1       2      3     3      3         2.9
##  9     1     1     1       2      4     2.2    3.2       2.5
## 10     1     1     1       2      5     2      2.8       1.9
## # ... with 150 more rows
```

For secure connection to **https** we use **read.csv()**

To dowlnoad data use:

```
library(readxl)
library(gdata)
```

```
## gdata: Unable to locate valid perl interpreter
## gdata:
## gdata: read.xls() will be unable to read Excel XLS and XLSX files
## gdata: unless the 'perl=' argument is used to specify the location of a
## gdata: valid perl intrpreter.
## gdata:
## gdata: (To avoid display of this message in the future, please ensure
## gdata: perl is installed and available on the executable search path.)

## gdata: Unable to load perl libaries needed by read.xls()
## gdata: to support 'XLX' (Excel 97-2004) files.

##

## gdata: Unable to load perl libaries needed by read.xls()
## gdata: to support 'XLSX' (Excel 2007+) files.

##

## gdata: Run the function 'installXLSXsupport()'
## gdata: to automatically download and install the perl
```

```
## gdata: libaries needed to support Excel XLS and XLSX formats.
##
## Attaching package: 'gdata'
## The following object is masked from 'package:stats':
##
##     nobs
## The following object is masked from 'package:utils':
##
##     object.size
## The following object is masked from 'package:base':
##
##     startsWith
```

```
url_xls <- "http://s3.amazonaws.com/assets.datacamp.com/production/course_1478/datasets/latitude.xls"
excel_gdata <- read.xls(url_xls, perl = perl)
```

```
## Warning in system(cmd, intern = intern, wait = wait | intern,
## show.output.on.console = wait, : running command 'C:\Windows\system32\cmd.exe /c
## ftype perl' had status 2

## Warning in system(cmd, intern = intern, wait = wait | intern,
## show.output.on.console = wait, : running command 'C:\Windows\system32\cmd.exe /c
## ftype perl' had status 2
```

```
print(excel_gdata)
```

```
##                            country       X1700
## 1                      Afghanistan  34.5650000
## 2              Akrotiri and Dhekelia  34.6166667
## 3                          Albania  41.3120000
## 4                          Algeria  36.7200000
## 5                   American Samoa -14.3070000
## 6                          Andorra  42.5460000
## 7                           Angola  -8.8430000
## 8                         Anguilla  18.2500000
## 9               Antigua and Barbuda  17.0720000
## 10                        Argentina -36.6760000
## 11                          Armenia  40.2540000
## 12                            Aruba  12.5130000
## 13                        Australia -32.2190000
## 14                          Austria  48.2310000
## 15                       Azerbaijan  40.3520000
## 16                          Bahamas  24.7000000
## 17                          Bahrain  26.0240000
## 18                       Bangladesh  23.8800000
## 19                         Barbados  13.1790000
## 20                          Belarus  53.5470900
## 21                          Belgium  50.8370000
## 22                           Belize  17.8430000
## 23                            Benin   6.3640000
## 24                          Bermuda  32.2170000
## 25                           Bhutan  27.4790000
## 26                          Bolivia -15.1900000
```

```
## 27            Bosnia and Herzegovina  44.1750100
## 28                          Botswana -21.5360000
## 29                            Brazil -19.5570000
## 30            British Virgin Islands  18.5000000
## 31                            Brunei   4.5010000
## 32                          Bulgaria  42.0730000
## 33                      Burkina Faso  12.0490000
## 34                           Burundi  -3.3650000
## 35                          Cambodia  12.0260000
## 36                          Cameroon  10.7300000
## 37                            Canada  43.7270000
## 38                        Cape Verde  15.0910000
## 39                     Cayman Islands  19.3190000
## 40               Central African Rep.   4.3310000
## 41                              Chad  10.3770000
## 42                    Channel Islands  49.2170000
## 43                             Chile -33.5540000
## 44                             China  29.5610000
## 45                  Christmas Island -10.5000000
## 46                      Cocos Island -12.5000000
## 47                          Colombia   4.7880000
## 48                          Comoros -11.6710000
## 49                  Congo, Dem. Rep.  -2.9202760
## 50                       Congo, Rep.  -3.6840000
## 51                      Cook Islands -21.2333333
## 52                        Costa Rica   9.9410000
## 53                     Cote d'Ivoire   5.4960000
## 54                           Croatia  45.1100800
## 55                              Cuba  23.0840000
## 56                            Cyprus  35.0810000
## 57                        Czech Rep.  49.7792900
## 58                           Denmark  55.7180000
## 59                          Djibouti  11.5050000
## 60                          Dominica  15.4330000
## 61                    Dominican Rep.  18.5610000
## 62                      East Germany          NA
## 63                           Ecuador  -2.0620000
## 64                             Egypt  29.9960000
## 65                       El Salvador  13.7750000
## 66                  Equatorial Guinea   2.3260000
## 67                           Eritrea  15.3120000
## 68                           Estonia  58.6850000
## 69                          Ethiopia   9.0070000
## 70                    Faeroe Islands  62.0000000
## 71      Falkland Islands (Malvinas) -51.7500000
## 72                              Fiji -17.8270000
## 73                           Finland  60.2120000
## 74                            France  48.8570000
## 75                     French Guiana   3.9880000
## 76                  French Polynesia -17.6660000
## 77                             Gabon   0.3720000
## 78                            Gambia  13.2570000
## 79                           Georgia  42.0340000
## 80                           Germany  48.1610000
```

```
## 81                        Ghana   6.6940000
## 82                    Gibraltar  36.1333333
## 83                       Greece  38.0580000
## 84                    Greenland  64.2170000
## 85                      Grenada  12.1150000
## 86                   Guadeloupe  16.1630000
## 87                         Guam  13.4400000
## 88                    Guatemala  14.6220000
## 89                     Guernsey  49.4666667
## 90                       Guinea  11.6710000
## 91                Guinea-Bissau  12.2620000
## 92                       Guyana   5.7610000
## 93                        Haiti  18.9320000
## 94                     Holy See  41.9000000
## 95                     Honduras  14.1940000
## 96            Hong Kong, China  22.7040000
## 97                      Hungary  47.4190000
## 98                      Iceland  63.8920000
## 99                        India  25.2740000
## 100                   Indonesia  -6.5620000
## 101                        Iran  35.3800000
## 102                        Iraq  33.3140000
## 103                     Ireland  54.6110000
## 104                  Isle of Man  54.2250000
## 105                      Israel  32.0840000
## 106                       Italy  45.4150000
## 107                     Jamaica  18.0550000
## 108                       Japan  35.7120000
## 109                      Jersey  49.2500000
## 110                      Jordan  31.6020000
## 111                  Kazakhstan  44.3080000
## 112                       Kenya  -0.5130000
## 113                    Kiribati   1.8470000
## 114          Korea, Dem. Rep.  39.5270000
## 115               Korea, Rep.  37.5530000
## 116                     Kosovo  42.5833333
## 117                     Kuwait  29.3260000
## 118                   Kyrgyzstan  41.4894000
## 119                        Laos  16.5470000
## 120                      Latvia  56.8580000
## 121                     Lebanon  34.1110000
## 122                     Lesotho -29.5950000
## 123                     Liberia   6.3850000
## 124                       Libya  32.6070000
## 125               Liechtenstein  47.1410000
## 126                   Lithuania  55.3180000
## 127                  Luxembourg  49.7800000
## 128               Macao, China  22.5060000
## 129             Macedonia, FYR  41.5738100
## 130                 Madagascar -18.9580000
## 131                     Malawi -15.8110000
## 132                   Malaysia   3.2690000
## 133                   Maldives   3.2500000
## 134                       Mali  12.5080000
```

```
## 135                           Malta  35.8870000
## 136                 Marshall Islands   9.0000000
## 137                       Martinique  14.6540000
## 138                       Mauritania  17.9250000
## 139                        Mauritius -20.2320000
## 140                          Mayotte -12.8333333
## 141                           Mexico  16.7590000
## 142           Micronesia, Fed. Sts.   7.3570000
## 143                          Moldova  47.1660000
## 144                           Monaco  43.7460000
## 145                         Mongolia  47.4930000
## 146                       Montenegro  42.7889900
## 147                       Montserrat  16.7500000
## 148                          Morocco  33.5930000
## 149                       Mozambique -18.4990000
## 150                          Myanmar  17.6790000
## 151                          Namibia -17.9790000
## 152                            Nauru  -0.5333333
## 153                            Nepal  27.7120000
## 154                      Netherlands  51.8740000
## 155            Netherlands Antilles  12.1920000
## 156                    New Caledonia -21.3290000
## 157                      New Zealand -36.8920000
## 158                        Nicaragua  12.2110000
## 159                            Niger  13.8760000
## 160                          Nigeria   6.5430000
## 161                             Niue -19.0333333
## 162                   Norfolk Island -29.0333333
## 163        Northern Mariana Islands  15.1780000
## 164                           Norway  59.9770000
## 165                             Oman  20.4450000
## 166                         Pakistan  31.1730000
## 167                            Palau   7.5000000
## 168                           Panama   9.2060000
## 169                 Papua New Guinea  -6.6000000
## 170                         Paraguay -25.5830000
## 171                             Peru -11.7940000
## 172                      Philippines  13.9220000
## 173                         Pitcairn -25.0666667
## 174                           Poland  50.2440000
## 175                         Portugal  38.8160000
## 176                      Puerto Rico  18.2250000
## 177                            Qatar  25.3090000
## 178                          Reunion -20.9540000
## 179                          Romania  44.5260000
## 180                           Russia  55.6750000
## 181                           Rwanda  -2.0320000
## 182                Saint Barthélemy  17.9000000
## 183                    Saint Helena -15.9500000
## 184           Saint Kitts and Nevis  17.3270000
## 185                      Saint Lucia  13.8980000
## 186                     Saint Martin  18.0833333
## 187 Saint Vincent and the Grenadines  13.2540000
## 188         Saint-Pierre-et-Miquelon  46.8333333
```

```
## 189                         Samoa -13.6330000
## 190                    San Marino  43.7666667
## 191            Sao Tome and Principe   1.0000000
## 192                  Saudi Arabia  23.0690000
## 193                       Senegal  14.7720000
## 194                        Serbia  44.0467300
## 195                    Seychelles  -4.6640000
## 196                  Sierra Leone   8.7010000
## 197                     Singapore   1.3550000
## 198                Slovak Republic  48.7853800
## 199                      Slovenia  46.1279000
## 200                Solomon Islands  -9.6250000
## 201                       Somalia  10.6330000
## 202                  South Africa -29.1300000
## 203                         Spain  37.3980000
## 204                     Sri Lanka   6.8680000
## 205                         Sudan  14.0430000
## 206                      Suriname   5.6050000
## 207                      Svalbard  78.0000000
## 208                     Swaziland -26.5450000
## 209                        Sweden  59.2780000
## 210                   Switzerland  47.4080000
## 211                         Syria  33.4580000
## 212                        Taiwan  23.6448500
## 213                    Tajikistan  37.8060000
## 214                      Tanzania  -2.1540000
## 215                      Thailand  13.7700000
## 216                   Timor-Leste  -8.8333333
## 217                          Togo   6.1940000
## 218                       Tokelau  -9.0000000
## 219                         Tonga -21.1730000
## 220           Trinidad and Tobago  10.4180000
## 221                       Tunisia  36.8160000
## 222                        Turkey  41.2020000
## 223                  Turkmenistan  39.1293800
## 224       Turks and Caicos Islands  21.7500000
## 225                        Tuvalu  -8.0000000
## 226                        Uganda   0.2280000
## 227                       Ukraine  50.2810000
## 228          United Arab Emirates  23.3900000
## 229                United Kingdom  51.5100000
## 230                 United States  34.3600000
## 231                       Uruguay -34.8220000
## 232                          USSR          NA
## 233                    Uzbekistan  41.2720000
## 234               Wallis et Futuna -13.3000000
## 235                       Vanuatu -15.2330000
## 236                     Venezuela   9.8430000
## 237            West Bank and Gaza  31.4166667
## 238                  West Germany          NA
## 239                Western Sahara  24.6191300
## 240                       Vietnam  10.7980000
## 241          Virgin Islands (U.S.)  17.7360000
## 242                    Yemen, Rep.  15.2280000
```

```
## 243                       Yugoslavia          NA
## 244                           Zambia -12.9420000
## 245                         Zimbabwe -17.8760000
## 246                            Åland  60.0000000
```

To workaround reading excel file with read_excel() we need to:

```
download.file(url_xls, destfile = "local_latitude.xls") #download file locally

#excel_readxl <- read_excel("local_latitude.xls") #read it
```

Download file and read it as RData file

```
url_rdata <- "https://s3.amazonaws.com/assets.datacamp.com/production/course_1478/datasets/wine.RData"
download.file(url_rdata, destfile = "wine_local.RData")
load("wine_local.RData")

summary(wine)
```

```
##     Alcohol        Malic acid        Ash        Alcalinity of ash
##  Min.   :11.03   Min.   :0.74   Min.   :1.360   Min.   :10.60
##  1st Qu.:12.36   1st Qu.:1.60   1st Qu.:2.210   1st Qu.:17.20
##  Median :13.05   Median :1.87   Median :2.360   Median :19.50
##  Mean   :12.99   Mean   :2.34   Mean   :2.366   Mean   :19.52
##  3rd Qu.:13.67   3rd Qu.:3.10   3rd Qu.:2.560   3rd Qu.:21.50
##  Max.   :14.83   Max.   :5.80   Max.   :3.230   Max.   :30.00
##    Magnesium      Total phenols     Flavanoids    Nonflavanoid phenols
##  Min.   : 70.00   Min.   :0.980   Min.   :0.340   Min.   :0.1300
##  1st Qu.: 88.00   1st Qu.:1.740   1st Qu.:1.200   1st Qu.:0.2700
##  Median : 98.00   Median :2.350   Median :2.130   Median :0.3400
##  Mean   : 99.59   Mean   :2.292   Mean   :2.023   Mean   :0.3623
##  3rd Qu.:107.00   3rd Qu.:2.800   3rd Qu.:2.860   3rd Qu.:0.4400
##  Max.   :162.00   Max.   :3.880   Max.   :5.080   Max.   :0.6600
##  Proanthocyanins Color intensity       Hue           Proline
##  Min.   :0.410   Min.   : 1.280   Min.   :1.270   Min.   : 278.0
##  1st Qu.:1.250   1st Qu.: 3.210   1st Qu.:1.930   1st Qu.: 500.0
##  Median :1.550   Median : 4.680   Median :2.780   Median : 672.0
##  Mean   :1.587   Mean   : 5.055   Mean   :2.604   Mean   : 745.1
##  3rd Qu.:1.950   3rd Qu.: 6.200   3rd Qu.:3.170   3rd Qu.: 985.0
##  Max.   :3.580   Max.   :13.000   Max.   :4.000   Max.   :1680.0
```

Get content of HTTP request

```
library("httr")
url <- "http://www.example.com/"
resp <- GET(url)

print(resp)
```

```
## Response [http://www.example.com/]
##   Date: 2022-03-26 11:44
##   Status: 200
##   Content-Type: text/html; charset=UTF-8
##   Size: 1.26 kB
## <!doctype html>
## <html>
## <head>
```

```
##     <title>Example Domain</title>
##
##     <meta charset="utf-8" />
##     <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
##     <meta name="viewport" content="width=device-width, initial-scale=1" />
##     <style type="text/css">
##     body {
## ...
```

```
raw_content <- content(as = "raw", resp)
head(raw_content)
```

```
## [1] 3c 21 64 6f 63 74
```

API and JSON

Read JSON data

```
library("jsonlite")
wine_json <- '{"name":"Chateau Migraine", "year":1997, "alcohol_pct":12.4, "color":"red", "awarded":fals

wine <- fromJSON(wine_json)
str(wine)
```

```
## List of 5
##  $ name       : chr "Chateau Migraine"
##  $ year       : int 1997
##  $ alcohol_pct: num 12.4
##  $ color      : chr "red"
##  $ awarded    : logi FALSE
```

```
url_sw4 <- "http://www.omdbapi.com/?apikey=72bc447a&i=tt0076759&r=json"
url_sw3 <- "http://www.omdbapi.com/?apikey=72bc447a&i=tt0121766&r=json"

sw4 <- fromJSON(url_sw4)
sw3 <- fromJSON(url_sw3)

# Print out the Title element of both lists
sw4$Title
```

```
## [1] "Star Wars"
```

```
sw3$Title
```

```
## [1] "Star Wars: Episode III - Revenge of the Sith"
```

```
# Is the release year of sw4 later than sw3?
sw4$Year > sw3$Year
```

```
## [1] FALSE
```

```
print(sw4)
```

```
## $Title
## [1] "Star Wars"
##
## $Year
## [1] "1977"
##
```

```
## $Rated
## [1] "PG"
##
## $Released
## [1] "25 May 1977"
##
## $Runtime
## [1] "121 min"
##
## $Genre
## [1] "Action, Adventure, Fantasy"
##
## $Director
## [1] "George Lucas"
##
## $Writer
## [1] "George Lucas"
##
## $Actors
## [1] "Mark Hamill, Harrison Ford, Carrie Fisher"
##
## $Plot
## [1] "Luke Skywalker joins forces with a Jedi Knight, a cocky pilot, a Wookiee and two droids to save
##
## $Language
## [1] "English"
##
## $Country
## [1] "United States"
##
## $Awards
## [1] "Won 7 Oscars. 63 wins & 29 nominations total"
##
## $Poster
## [1] "https://m.media-amazon.com/images/M/MV5BNzVlY2MwMjktM2E4OS00Y2Y3LWE3ZjctYzhkZGM3YzA1ZWM2XkEyXkF
##
## $Ratings
##                     Source  Value
## 1 Internet Movie Database 8.6/10
## 2         Rotten Tomatoes    92%
## 3              Metacritic 90/100
##
## $Metascore
## [1] "90"
##
## $imdbRating
## [1] "8.6"
##
## $imdbVotes
## [1] "1,312,386"
##
## $imdbID
## [1] "tt0076759"
##
```

```
## $Type
## [1] "movie"
##
## $DVD
## [1] "06 Dec 2005"
##
## $BoxOffice
## [1] "$460,998,507"
##
## $Production
## [1] "N/A"
##
## $Website
## [1] "N/A"
##
## $Response
## [1] "True"
```

Build from Json object

```
# Challenge 1
json1 <- '[1, 2, 3, 4, 5, 6]'
fromJSON(json1)
```

```
## [1] 1 2 3 4 5 6
```

```
# Challenge 2
json2 <- '{"a": [1, 2, 3], "b": [4, 5, 6]}'
fromJSON(json2)
```

```
## $a
## [1] 1 2 3
##
## $b
## [1] 4 5 6
```

```
# Challenge 1
json1 <- '[[1, 2], [3, 4]]'
fromJSON(json1)
```

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

```
# Challenge 2
json2 <- '[{"a": 1, "b": 2}, {"a": 3, "b": 4}, {"a": 5, "b": 6}]'
fromJSON(json2)
```

```
##   a b
## 1 1 2
## 2 3 4
## 3 5 6
```

Convert the data to JSON

```
url_csv <- "http://s3.amazonaws.com/assets.datacamp.com/production/course_1478/datasets/water.csv"

# Import the .csv file located at url_csv
water <- read.csv(url_csv, stringsAsFactors=FALSE)
```

```
# Convert the data file according to the requirements
water_json <- toJSON(water)
print(water_json)
```

```
## [{"water":"Algeria","X1992":0.064,"X2002":0.017},{"water":"American Samoa"},{"water":"Angola","X1992
```

Minify and prettify JSON

```
pretty_json <- toJSON(mtcars, pretty = TRUE)

# Print pretty_json
#pretty_json - style the json one above the other

# Minify pretty_json: mini_json
mini_json <- minify(pretty_json)

# Print mini_json
mini_json
```

```
## [{"mpg":21,"cyl":6,"disp":160,"hp":110,"drat":3.9,"wt":2.62,"qsec":16.46,"vs":0,"am":1,"gear":4,"carb
```

Importing data from other sources (SAS, STATA, SPSS)

With haven package

library("haven") 1. read_sas()

2. Import from STATA

```
library("haven")

sugar <- read_dta("http://assets.datacamp.com/production/course_1478/datasets/trade.dta")

# Structure of sugar
str(sugar)
```

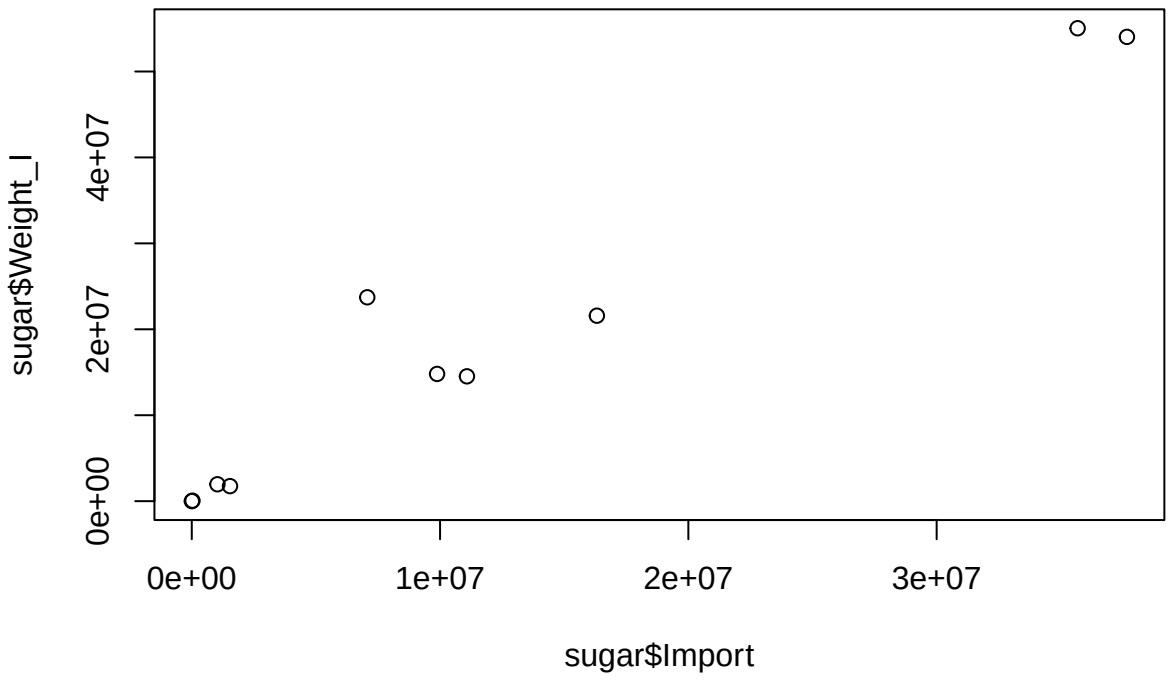
```
## tibble [10 x 5] (S3: tbl_df/tbl/data.frame)
##  $ Date    : dbl+lbl [1:10] 10,  9,  8,  7,  6,  5,  4,  3,  2,  1
##    ..@ label       : chr "Date"
##    ..@ format.stata: chr "%9.0g"
##    ..@ labels      : Named num [1:10] 1 2 3 4 5 6 7 8 9 10
##    .. ..- attr(*, "names")= chr [1:10] "2004-12-31" "2005-12-31" "2006-12-31" "2007-12-31" ...
##  $ Import  : num [1:10] 37664782 16316512 11082246 35677943 9879878 ...
##   ..- attr(*, "label")= chr "Import"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ Weight_I: num [1:10] 54029106 21584365 14526089 55034932 14806865 ...
##   ..- attr(*, "label")= chr "Weight_I"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ Export  : num [1:10] 5.45e+07 1.03e+08 3.79e+07 4.85e+07 7.15e+07 ...
##   ..- attr(*, "label")= chr "Export"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ Weight_E: num [1:10] 9.34e+07 1.58e+08 8.80e+07 1.12e+08 1.32e+08 ...
##   ..- attr(*, "label")= chr "Weight_E"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  - attr(*, "label")= chr "Written by R."
```

```r
# Convert values in Date column to dates
sugar$Date <- as.Date(as_factor(sugar$Date))

# Structure of sugar again
str(sugar)
```

```
## tibble [10 x 5] (S3: tbl_df/tbl/data.frame)
##  $ Date    : Date[1:10], format: "2013-12-31" "2012-12-31" ...
##  $ Import  : num [1:10] 37664782 16316512 11082246 35677943 9879878 ...
##   ..- attr(*, "label")= chr "Import"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ Weight_I: num [1:10] 54029106 21584365 14526089 55034932 14806865 ...
##   ..- attr(*, "label")= chr "Weight_I"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ Export  : num [1:10] 5.45e+07 1.03e+08 3.79e+07 4.85e+07 7.15e+07 ...
##   ..- attr(*, "label")= chr "Export"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ Weight_E: num [1:10] 9.34e+07 1.58e+08 8.80e+07 1.12e+08 1.32e+08 ...
##   ..- attr(*, "label")= chr "Weight_E"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  - attr(*, "label")= chr "Written by R."
```

```r
plot(sugar$Import, sugar$Weight_I)
```



3. Import data from SPSS with read_sav()

```
# Import SPSS data from the URL: work
work <- read_sav("http://s3.amazonaws.com/assets.datacamp.com/production/course_1478/datasets/employee.s

summary(work$GENDER)
```

```
##    Length     Class      Mode
##       474 character character
```

```
# Convert work$GENDER to a factor
work$GENDER <- as_factor(work$GENDER)
summary(work$GENDER)
```

```
## Female   Male
##    216    258
```

With foreign package

1. From STATA read.dta("florida.dta")

The arguments you will use most often are **convert.dates**, **convert.factors**, **missing.type** and **convert.underscore**

2. Form SPSS read.spss("international.sav", to.data.frame=TRUE) boxplot(demo$gdp)