RcodeforUKB-Zhu

Xiwen Liu

2023-09-11

Contents

This R markdown file is for the students who chose Prof. Zhu's projects and have never used R before. The code in this file should be enough for you to do data investigation and data preparation with UKB data.

1 Download R and Rstudio

Click here to download R and Rstudio

RStudio is an integrated development environment for R, a programming language for statistical computing and graphics.

2 Get to know Rstudio

When you first open it, click the plus "+" button at the top left to create a new R script. A R script is a file where all your code is stored. R script can also be created as follow: File –> New –> R Script.

The RStudio interface now is divided into four "Panes" as described below. Check Figure 1.

- Top LEFT: the source editor, where you edit scripts, documents and can "send" code to run in the console
- Bottom LEFT: the R console, where the code is run.
  - You can also type code directly into the console
  - or you can send it to the console by running it from the source editor
- Top RIGHT: your environment/history panes, where you can see variables you've created and a full history of functions / commands you have run

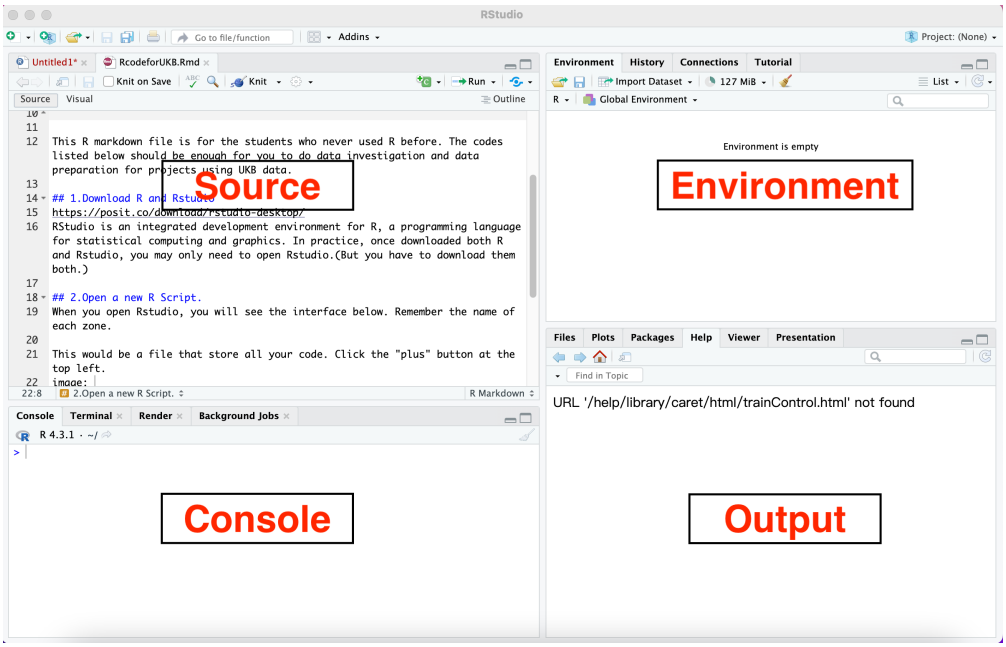

Figure 1: R Studio Panes

- Bottom RIGHT: Output Pane, containing several panes including:
  - files, where you can explore files on your computer like you would in windows explorer or finder on a mac
  - plots, where you will see plots that you create
  - packages/help/viewer.

3 Set up your working directory

The working directory is a folder where R reads and saves files. Use the menu to change your working directory under Session > Set Working Directory > Choose Directory. See Figure 2. Or use the code:

```
setwd("type your folder path here")
```

4 Load the data

You will have to replace the path in green with your own file path. A file path tells Rstudio where your data file is located in your computer. Here I named the sample data as "rawdata", and the dictionary data as "dicdata".

```
rawdata <- read.csv("~/Desktop/ZhuRiskModel/ukb673329.csv", header = T, check.names = F)
dicdata <- read.csv("~/Desktop/ZhuRiskModel/1Data_Dictionary_Showcase.csv", header = F)
```

Once loaded the data, you will see in the Environment pane that now you have 2 datasets, one called "rawdata" and the other "dicdata", and their dimensions. Check Figure 3.

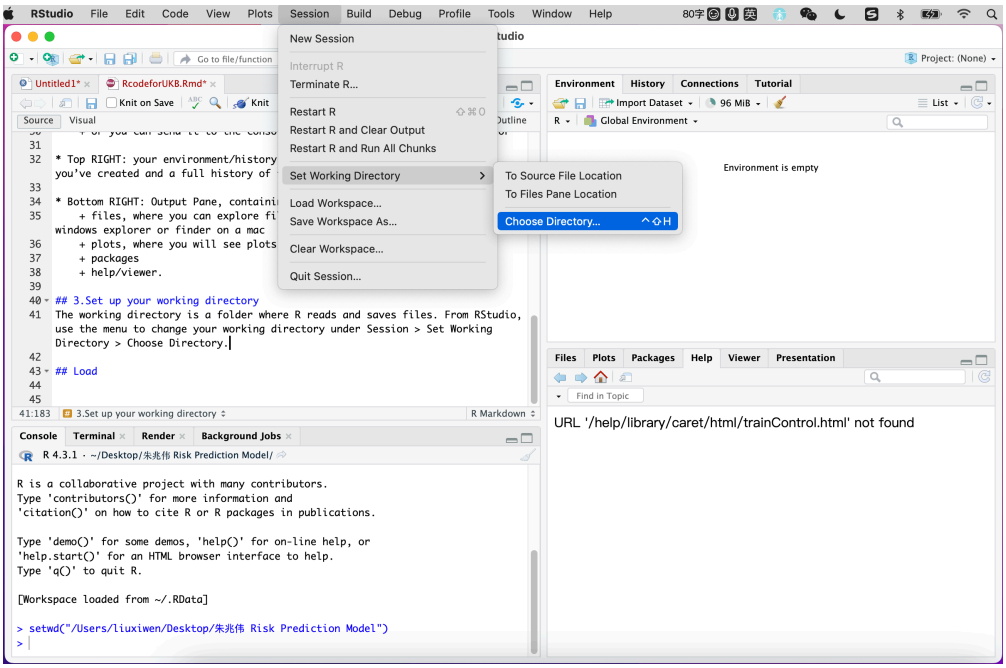

Figure 2: Set up the working directory

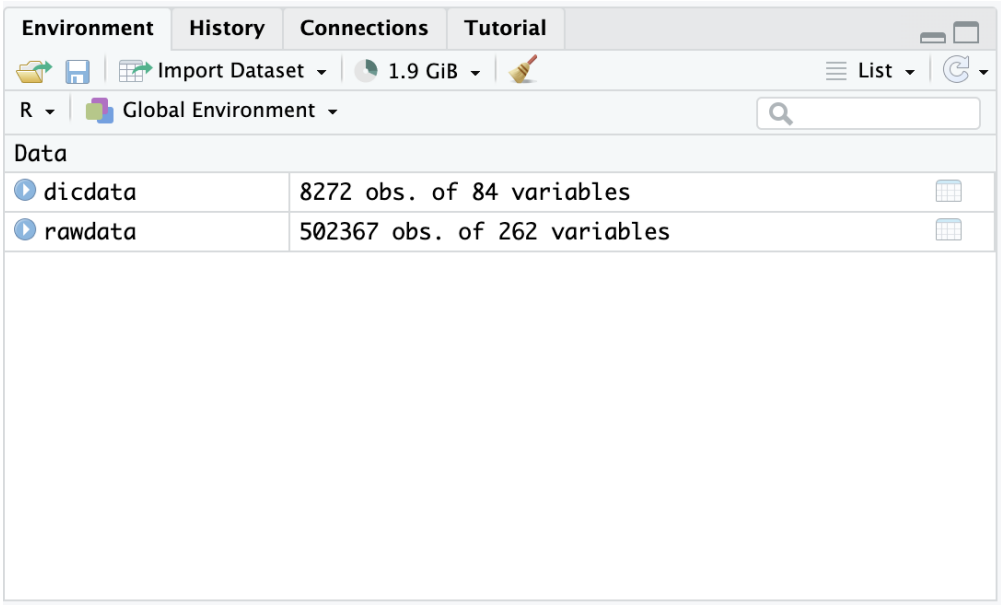

Figure 3: rawdata and dicdata have been loaded

4.1 View the data

The rawdata is a very wide and long data frame (262 variables/columns and 502367 observations/rows), so we will just to view the first 9 columns and the first 5 rows to have an idea of the data.

```
rawdata[1:5, 1:9]
```

```
##       eid 31-0.0 34-0.0 46-0.0 46-1.0 46-2.0 46-3.0 47-0.0 47-1.0
## 1 1000017      0   1951     24     NA     NA     NA     30     NA
## 2 1000025      1   1944     38     NA     NA     NA     38     NA
## 3 1000038      0   1942     18     NA     NA     NA     18     NA
## 4 1000042      0   1960     12     NA     NA     NA     25     NA
## 5 1000056      0   1968     23     NA     NA     NA     26     NA
```

Otherwise, we can randomly select some small numbers of columns and rows to view.

```
rawdata[sample(1:502367, 5, replace=FALSE), sample(1:262, 7, replace=FALSE)]
```

```
##        23104-1.0 3160-1.0 40002-0.3 6153-1.1 48-3.0 40002-0.14 6177-1.1
## 198576        NA       NA                NA     NA                  NA
## 274424      25.8       NA                NA     NA                  NA
## 266611        NA       NA                NA     NA                  NA
## 465398        NA       NA                NA     NA                  NA
## 390138        NA       NA                NA     NA                  NA
```

Check the data type. That is, whether variables are numeric, characters, factors. Most common types are:

- Numeric (1.2, 5, 7, 3.14159)
- Integer (1, 2, 3, 4, 5)
- Complex (i + 4)
- Logical (TRUE / FALSE)
- Character (“a”, “apple”)

```
unique(sapply(rawdata, class))
```

```
## [1] "integer"   "numeric"   "character" "logical"
```

```
str(colnames(rawdata))
```

```
##  chr [1:262] "eid" "31-0.0" "34-0.0" "46-0.0" "46-1.0" "46-2.0" "46-3.0" ...
```

4.2 View the dictionary

Let’s have a look of the dictionary.

```
dicdata[1:5, 1:7]
```

```
##                                                          V1     V2 V3
## 1      Population characteristics > Baseline characteristics 100094 31
## 2      Population characteristics > Baseline characteristics 100094 33
## 3      Population characteristics > Baseline characteristics 100094 34
## 4 Assessment Centre > Physical measures > Hand grip strength 100019 46
## 5 Assessment Centre > Physical measures > Hand grip strength 100019 47
##                           V4     V5 V6     V7
## 1                        Sex 502413  1 502413
## 2              Date of birth 502413  1 502413
```

```
## 3              Year of birth 502413  1 502413
## 4  Hand grip strength (left) 499191  1 574280
## 5 Hand grip strength (right) 499260  1 574364
```

It seems that V3 corresponds with the column names in the rawdata, and V4 is their explaination. Let's create a new dictionary that contains only these two columns.

```
dict <- dicdata[3:4]
colnames(dict) <- c('feature', 'meaning')
head(dict, 5)
```

```
##   feature                    meaning
## 1      31                        Sex
## 2      33              Date of birth
## 3      34              Year of birth
## 4      46  Hand grip strength (left)
## 5      47 Hand grip strength (right)
```

5 Search important words in the dictionary and find the columns in data

grep() function can help you find the certain character in a string vector. For example I want to find if there is any "meaning" contains "grip strength":

```
grep("Grip strength", dict$meaning, ignore.case = T,value = TRUE)
```

```
## [1] "Hand grip strength (left)"
## [2] "Hand grip strength (right)"
## [3] "Reason for skipping grip strength (right)"
## [4] "Reason for skipping grip strength (left)"
```

```
#I have already tuned the function so it is not case-sensitive
```

Now let's create a function to form a sample data set that contains the keywords we put in.

```
sampledata <- function(keywords, dict, rawdata) {
  meaningindex<-grep(as.character(keywords), dict$meaning, ignore.case = T,value = F)
  featureindex<-dict$feature[meaningindex]
  prefix.feature<-sub("\\-.*", "", colnames(rawdata))
  sampledata<- cbind(rawdata[1],rawdata[which(prefix.feature %in% featureindex)])
  return(sampledata)
}
```

```
sample1 <- sampledata(keywords = "hand", dict = dict, rawdata = rawdata)
sample1[1:5, 1:5]
```

```
##       eid 46-0.0 46-1.0 46-2.0 46-3.0
## 1 1000017     24     NA     NA     NA
## 2 1000025     38     NA     NA     NA
## 3 1000038     18     NA     NA     NA
## 4 1000042     12     NA     NA     NA
## 5 1000056     23     NA     NA     NA
```