Reading data | R programming language

Reading a file from a disk

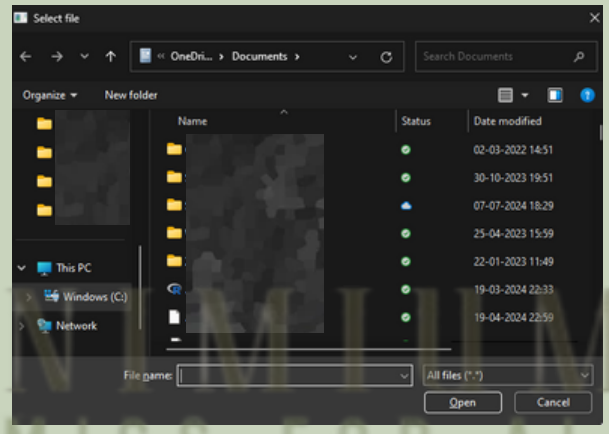
- *R looks for data in the default directory, the command for the same is* getwd(). *Running this will give* C:/Users/abcd/OneDrive/Documents *for windows.*
- *You can set your own default (say desktop) use* setwd('desktop').
- *To view directory-* dir('documents') *or* list.files.()
- *To see all files* (invisible) dir(all.files+TRUE)

Better methods

But without changing the directory one can read a file using file.choose()

Running this will give you a dialogue box from where you can choose the file needed to read and the path for the same will be visible in the console.

Dialogue box



Script Editor

```
30  file.choose()
```

Another way is using the command scan(file= pathname)

After getting the path, copy paste the path in the command as scan(file= "C:\\Users\\abcd\\OneDrive\\Desktop\\x.csv")

Running this will give error.Why?

Script Editor

```
144  scan(file= "C:\\Users\\     \\OneDrive\\Desktop\\x.csv")
```

Console

```
> scan(file= "C:\\Users\\     \\OneDrive\\Desktop\\x.csv")
Error in file(file, "r") : cannot open the connection
In addition: Warning messages:
1: In check_dep_version() : ABI version mismatch:
lme4 was built with Matrix ABI version 1
Current Matrix ABI version is 0
Please re-install lme4 from source or restore original 'Matrix' package
2: In file(file, "r") :
  cannot open file 'C:\Users\     \OneDrive\Desktop\x.csv': No such file or directory
```

Remember to also add instructions like what= and sep= to read the file

In this example x.csv file was chosen after running the command file.choose *and the file is comma seperated value (as we know for a csv file) and hence we must add* sep=','

Also since it has words we must add what='char' *as well after the pathname.*

Script Editor

```
157  scan(file= "C:\\Users\\     \\OneDrive\\Desktop\\x.csv", what= 'char', sep = ',')
```

Reading data | R programming language

Reading bigger data files

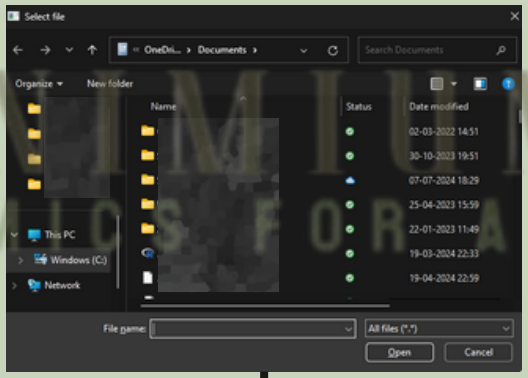
- *One of the most basic and convenient modes of file types for econometric analysis is csv (comma delimited) file type.*
- *The most common command used is* read.csv
- *Since its already a command specifically for csv type, there is no need for the instruction* sep=','

Script Editor

```
163  read.csv(file= "C:\\Users\\    \\OneDrive\\Desktop\\x.csv")
```

- *Instead of typing the entire pathname in the command, the command* read.csv(file.choose()) *can also be used to read the same file.*
- *Running this will make a dialogue box to appear where we will choose the file that we like to read. The chosen file will appear in the console as shown.*
- *The instruction* header=TRUE *reads the first row of the csv file and sets this as the name for each rows. Use this if needed. if you have column headings(which is rare) use the instruction* header= FALSE, row.names=n *where n is the number of rows.*

Dialogue box



Script Editor

```
170  read.csv(file.choose())
```

Note:- When you have missing values R substitutes the blank with 'NA'

Console

```
> read.csv(file.choose())
     Id   low    up age gender marital workscheme state  api  ahi children earning whours
1   DEL1 0e+00     0   0      1       1          0     1  7.5  7.5        3       1    8.0
2   DEL2 0e+00     0   0      0       1          1     1 25.0 25.0        1       1    7.0
3   DEL3 0e+00     0   0      1       1          1     1  7.5  7.5        2       1    6.0
4   DEL4 0e+00     0   0      0       1          1     1 12.5 12.5        2       2    8.0
5   DEL5 4e+03  6000   0      1       1          1     1  7.5 12.5        3       2    6.0
6   DEL6 0e+00     0   0      0       1          0     1 25.0 17.5        1       2    6.0
7   DEL7 4e+03  6000   0      0       1          0     1  5.0  5.0        2       3    9.0
8   DEL8 0e+00     0   0      0       1          1     1  5.0  5.0        2       3    7.5
```

Viewing previously loaded named-objects: the ls() command

- *Running this will give output as shown below as these are the objects named so far.*
- *When you have a large dataset named before unlike here and if you want particular ones to be viewed then use* ls(pattern=) *commands*
- ls(pattern='h')*means R will show all objects created that contain the letter h*
- *ls(pattern='fe') means R will show all objects created that contain the letters 'fe' together*
- ls(pattern='^k') *means R will show all objects created that start with the letter k*
- ls(pattern='^be') *means R will show all objects created that start with 'be'*
- ls(pattern='[zy]')*or* ls(pattern='z|y') *means R will show all objects created that contain the letters z or y or both*
- ls(pattern='g$') *means R will show all objects created that end with the letter g.*
- ls(pattern='l.z') *means R will show all objects created that start with letters l and z one letter apart (2 dots for two letters apart and so on.*

Reading data | R programming language

Script Editor

```
ls()
ls(pattern='h')
ls(pattern='fe')
ls(pattern='^k')
ls(pattern='[zy]')
ls(pattern='z|y')
ls(pattern='g$')
ls(pattern='l.z')
```

Console *Note:- When you have missing values R substitutes the blank with 'NA'*

```
> ls()
  [1] "a"                    "a1"                   "actual"                "adf_test"
  [5] "alpha"                "B"                    "b1"                    "C"
  [9] "c1"                   "censored_data"        "chi_square_statistic"  "cloglog_model"
 [13] "co2_emissions"        "coefficients"         "criteria"              "curve_data"
 [17] "curve_plot"           "D"                    "d1"                    "data"
 [21] "data_index"           "data_normalized"      "df"                    "dh_model"
> ls(pattern='h')
 [1] "alpha"                "chi_square_statistic" "dh_model"              "dh_modell"
 [5] "h"                    "hausman_test"         "heteroscedasticity_test" "population_growth"
 [9] "shapiro_test_result"  "threshold"            "variables_with_interaction" "weights"
[13] "within"               "youthunempolyment"
> ls(pattern='fe')
[1] "effect_size"          "model_fixed_effects"  "model_random_effects"
> ls(pattern='^k')
[1] "kuznets_curve_data"
> ls(pattern='[zy]')
 [1] "data_normalized"      "dummyvariable"        "effect_size"           "energy_consumption"
 [5] "frequency"            "heteroscedasticity_test" "kuznets_curve_data"  "outliers_z"
 [9] "poverty_gap"          "proxy"                "standardized_data"     "unemployment"
[13] "y"                    "your_data"            "youthunempolyment"     "z_scores"
> ls(pattern='z|y')
 [1] "data_normalized"      "dummyvariable"        "effect_size"           "energy_consumption"
 [5] "frequency"            "heteroscedasticity_test" "kuznets_curve_data"  "outliers_z"
 [9] "poverty_gap"          "proxy"                "standardized_data"     "unemployment"
[13] "y"                    "your_data"            "youthunempolyment"     "z_scores"
> ls(pattern='g$')
[1] "simplereg"
> ls(pattern='l.z')
[1] "data_normalized"
```

To remove the variable from the environment

- rm(a) - *This command removes object a*
- rm(list=ls(pattern='a..e')) - *This command will remove those objects created in the environment whose name has the letters a and e two letters apart.*

To *remove all the variables from the environment* *the following command is used (which makes the environment panel empty as shown)*

rm(list=ls())

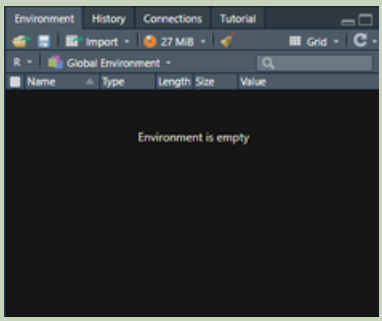
Script Editor

```
rm(a)
rm(list=ls(pattern='a..e'))
rm(list=ls())
```

Console

```
> rm(list=ls())
```

Environment



Note:- To clear console use ctrl+l

Reading data | R programming language

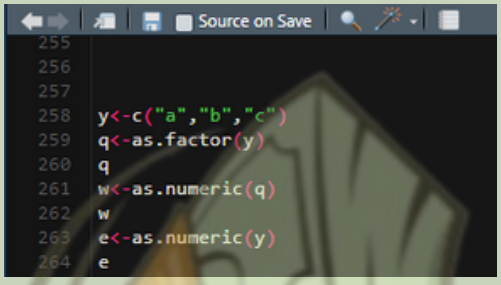
Converting class of data/datasets

Types of data

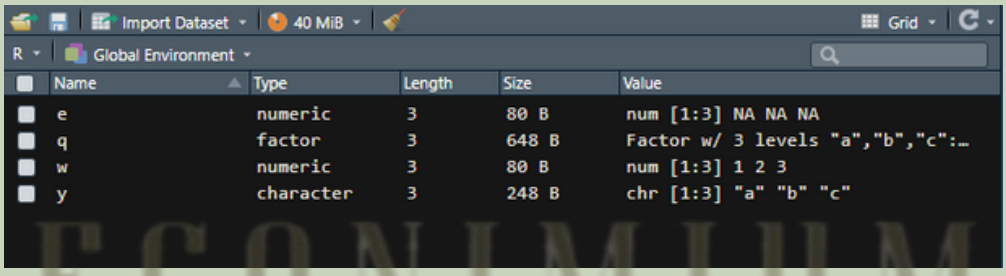
- *1 2 3 4 8 5 3 2 4 9- this data set is called integer*
- *1.2 2 3 6 5.8 4 3.8 9.5 7- this data set is called numeric*
- *"a" "b" "c"- this data set is called character*
- *a b c - this data set is called factor*
- *y<-c("a","b","c") is a data set which is called character*

To convert this dataset y into factor we use the command as.factor *and to revert it back* as.character. Same goes for all other classes, *but conversion of character to number is not possible in which case the output will be NA NA NA*

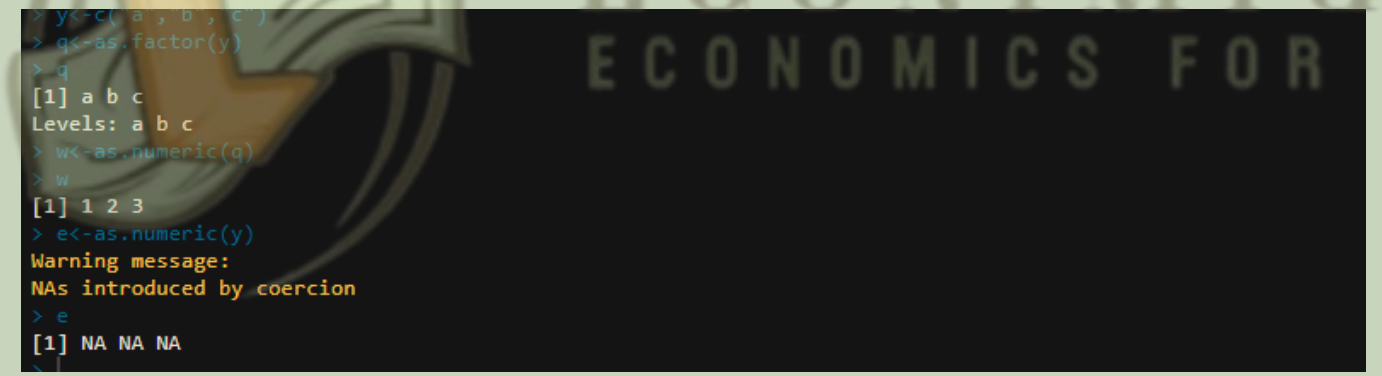
Script Editor



Environment

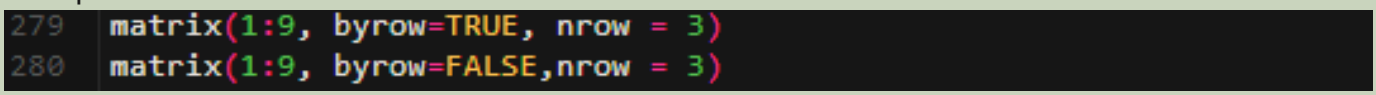


Console

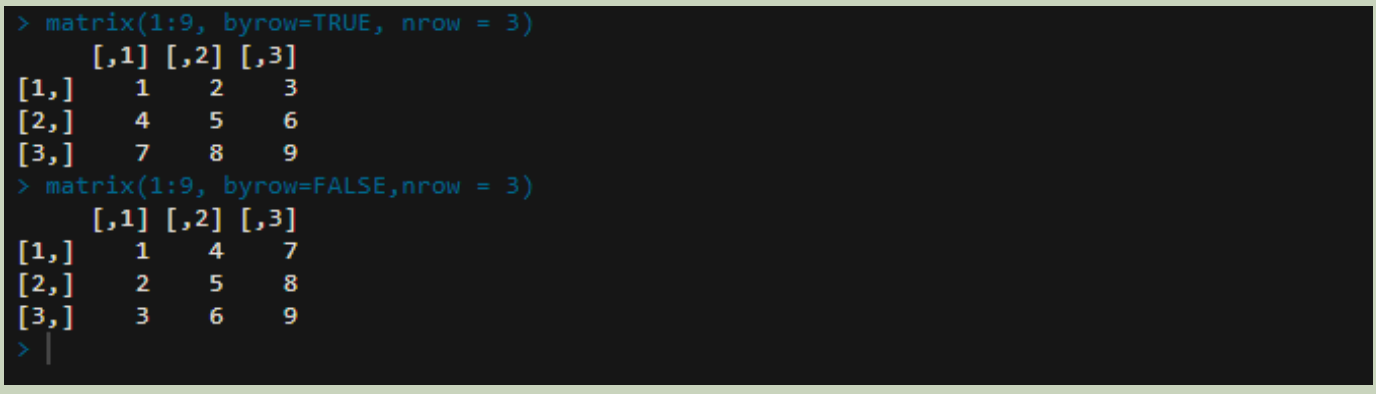


Creating Matrices

Script Editor



Console



Reading data | R programming language

History

- *To view the entire history-* history()
- *To view a few of the lines (say 13)-* history(max.show = 13)
- *To save history-* savehistory(file='name.Rhistory')
- *To load a list of instructions-* loadhistory(file="name.Rhistory")

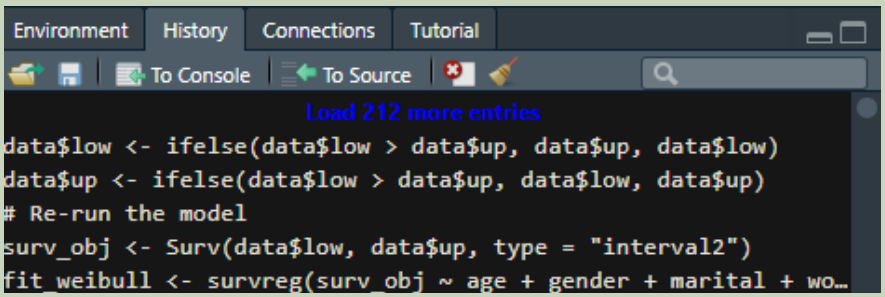
Script Editor

```
294  history()
```

Console

```
> history()
```

History Panel



Script Editor

```
history(max.show = 13)
```

Console

```
> history(max.show = 13)
```

History Panel

```
rm(list=ls())
y<-c("a","b","c")
q<-as.factor(y)
q
w<-as.numeric(q)
w
e<-as.numeric(y)
e
matrix(1:9, byrow=TRUE, nrow = 3)
matrix(1:9, byrow=FALSE,nrow = 3)
history()
history(max.show = 13)
history(max.show = 13)
```

Script Editor

```
savehistory(file='name.Rhistory')
loadhistory(file="name.Rhistory")
```