ICSSR DATA SERVICE

HTTP://WWW.ICSSRDATASERVICE.IN/

ICSSR Data Service

Indian Social Science Data Repository

"R": User Guide

Indian Council of Social Science Research

ICSSR Data Service

R 3.2.2: User Guide

Contents

1. Introduction

R is a language, a system environment to run statistical analysis, and represent the graphics. It was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues. R is a dialect of S language, as it falls under the categories of not only software but also in language. But interestingly, R is an interpreted language which means the users need not to build a programme like in other language (e.g. C). In R, the commands written by user directly get executed. The commands should be always written in parentheses (e.g., str ( )), then only the commands will be executed, otherwise R will display only the contents of function. The interesting feature of R is that while using R, the uploaded data, variables and functions are stored in the computer's memory in the form of objects, which are not visible at the front page of R.

2. Installation

Firstly, user should install the R language software which is available as free software. After the installation, you can open it by clicking the R short-cut icon located on the desktop. The R interface will appear on your system screen as shown in Fig. 1. Here, the window with the name of R Console is shown, where the commands are to be written. In the R Console window, you can see the symbol ">"which means R is asking to write the command of a function for its execution.

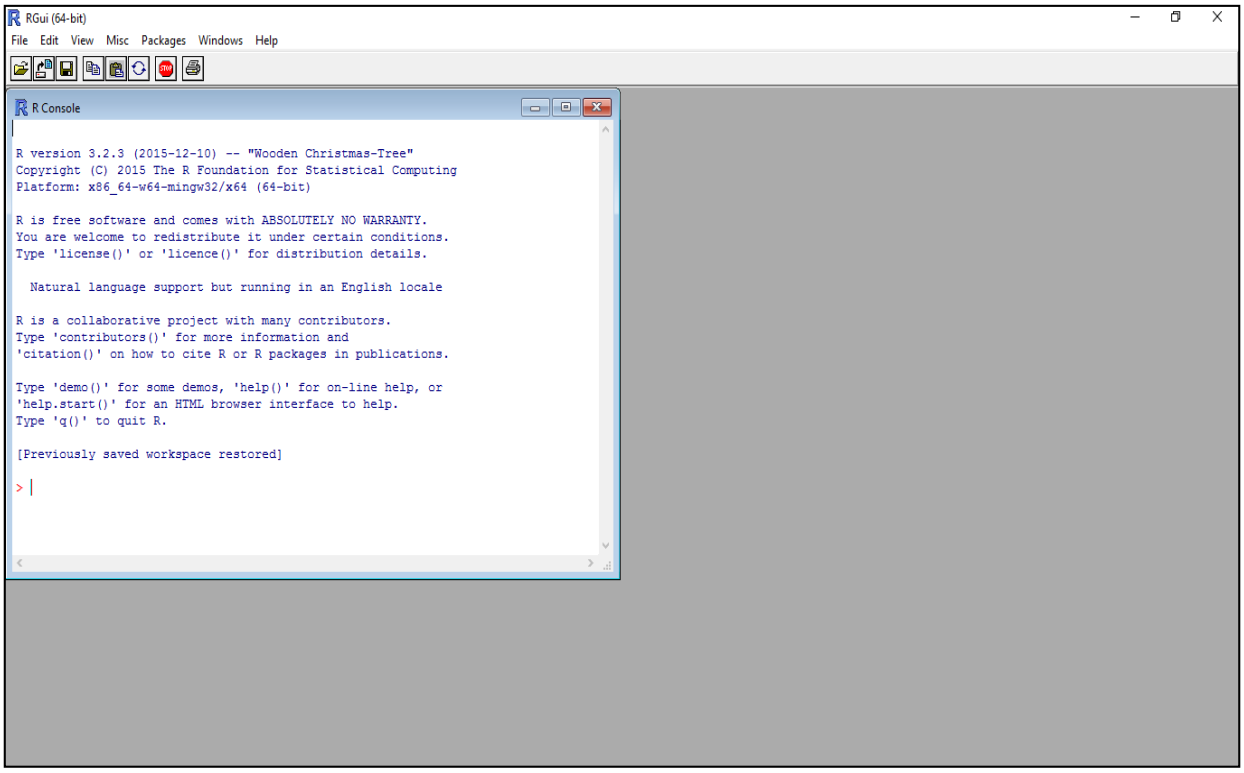


Fig. 1: R Interface

3. Assignment

Feeding the data in R is called assigning the data in R object. The process of assigning data in R object is very simple as shown in Fig. 2. First, write down the name of variables for which you need to assign values of the data. For example, to assign value to variable x; use less than (<) and minus sign (–) i.e. "<-" . After using this, you need to write "c" and within brackets the values. As such the following command is to be written:

x<-c (10, 12, 14, 16, 17, 18, 20)

This command implies that you have assigned values of 10, 12, 14,16,17,18 and 20 to the x variable. Similarly, you may assign values to "y" variable. After assigning the values to x and y, you can simply write "x" or "y" and the software will fetch and display these values as shown in Fig. 2.

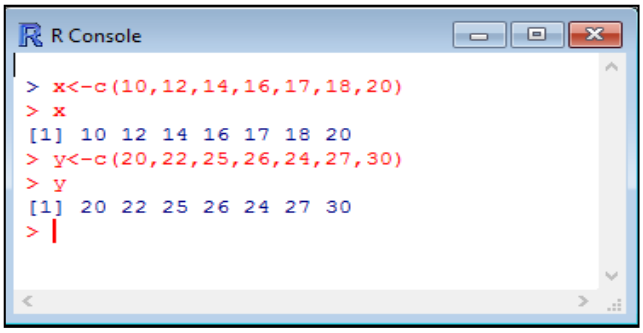


Fig. 2: Assigning Value to Variable

4. Opening Dataset in R

In this manual, the example of dataset for NSSO round “Schedule 25.2: Social Consumption: Education, 71st Round" is used. This survey was conducted during 2014.

To open a dataset in R, firstly the file which is to be open, needs to be *save as* ***CSV*** (comma separated values) file. After saving the file in CSV format, you may open the file by writing the “read.csv” command in R Console. Fig. 3shows the command to open a "csv" file in R. Here firstly, you need to assign the name to dataset. In this example, it is assigned as "Data". After that use "<-", then write the "read.csv" and within bracket, type the location of the file which need to be opened. Further, the term ***header=T***is to be used, if the first row in the "csv" file contains the name of variables.

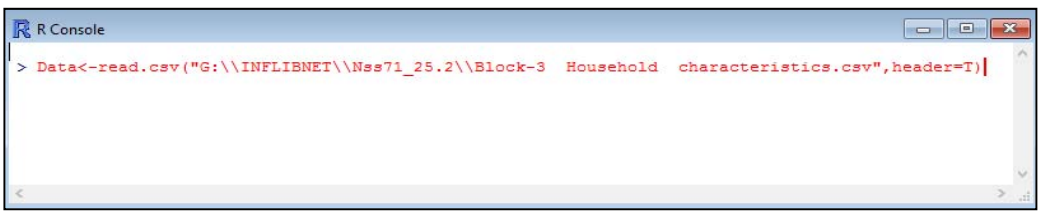


Fig. 3: Opening File in R by writing "read.csv" Command

To check whether data is uploaded or not onto the R system, type the name of dataset, in this case, write "Data" as shown in Fig. 4. This will produce the result as shown in Fig.5.

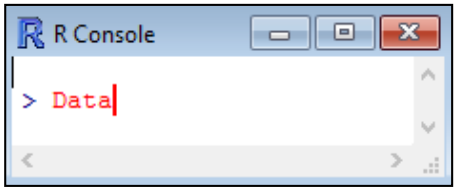


Fig. 4: Checking Status of Uploaded Data

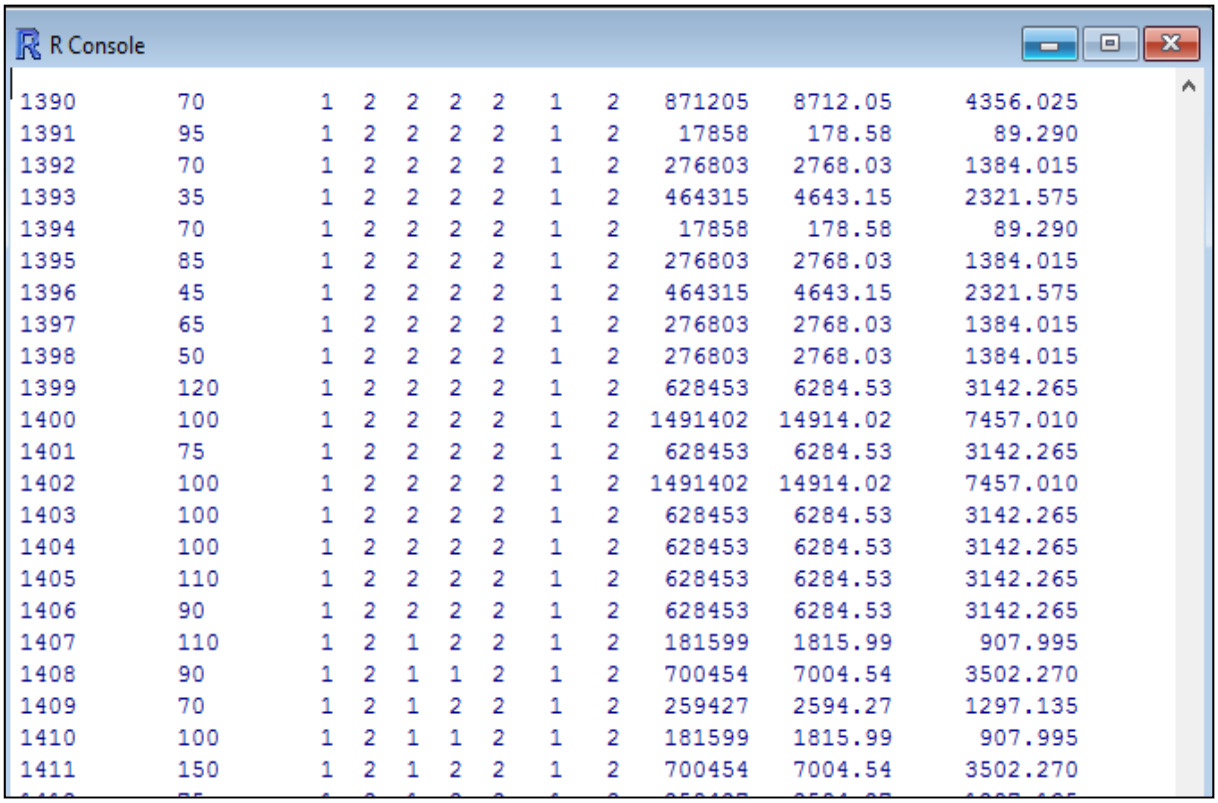


Fig. 5: Display of Uploaded Data

Similarly, you may also open other types of files in R, such as: sav, dta, txt, etc. These files can only be opened after uploading the concerned foreign software package, e.g. to open the sav and dta file, foreign software package is required. In this manual, the process on how to open sav file using foreign software package is described.

Before installing any new software into R, take the steps given below:

Set the CRAN Mirror from the **Packages dropdown menu**. As result, Fig. 6 will appear, where you need to select the nearest location for faster downloading process, i.e. "India" in this case. Please note that some of the foreign software package may not available in all CRAN mirror sites.

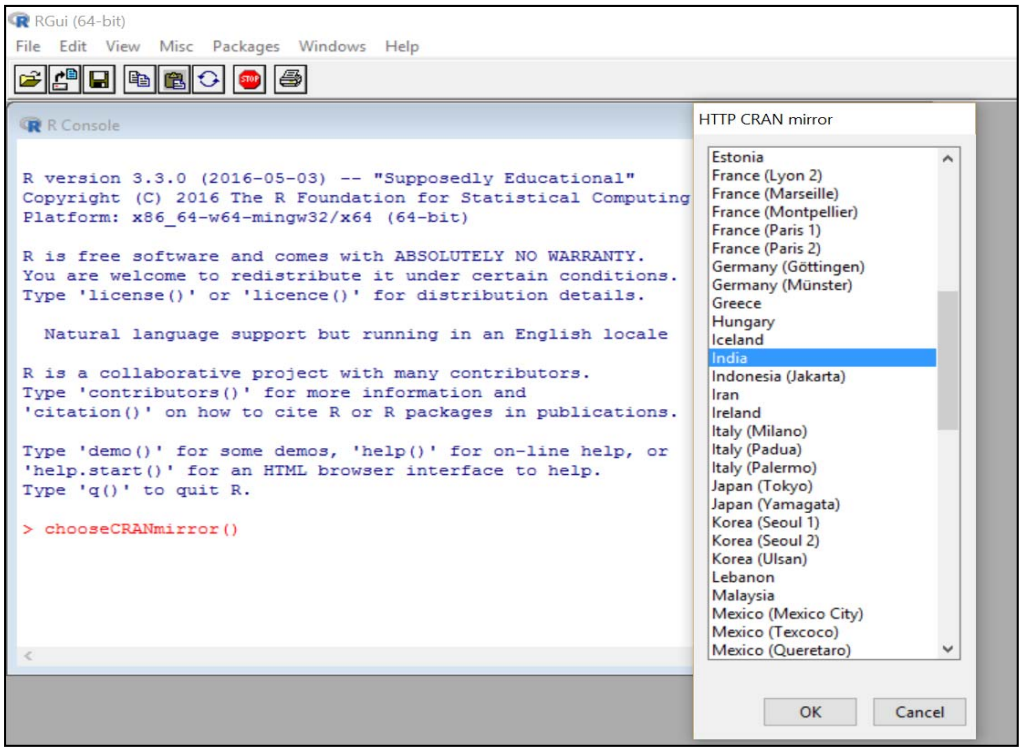


Fig. 6: Selection of CRAN Mirror to Download Foreign Software Package

Now, click on install packages from the dropdown menus, as a result, Fig. 7 will appear.

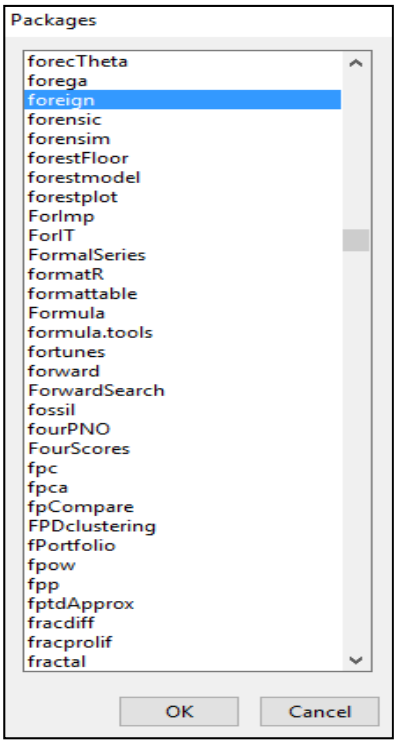


Fig. 7: Selecting Foreign Software Packages

Select the “foreign” from the list of packages, as is shown in Fig. 7 After installing the foreign software package, write down the following command: ***“require (foreign)“*** which will load the foreign package into R system. Then, use the "read.spss" command as given below to read the "sav" file. For example, "read.spss" command is used here to read the "Nss71_25.2\\Block-1&2-Level-01-Identification of sample household and particulars of field operations "file.

/Data<-read.spss("G:\\INFLIBNET \\Nss71_25.2\\Block-1&2- Level-01-Identification of sample household and particulars of field operations.sav",use.value.labels=TRUE, to.data.frame=TRUE).

5. Basic Commands

5.1 Str Comand

In many datasets, one of the basic things users want to check is about the structure of data. In R, "Str command" is used to check the structure of data. The command to be written as: Str(file name).

In this example, it has been written "str(HH)". As a result, Fig. 8 will appear, showing the numbers of observations, numbers of variables, characters of variables etc.

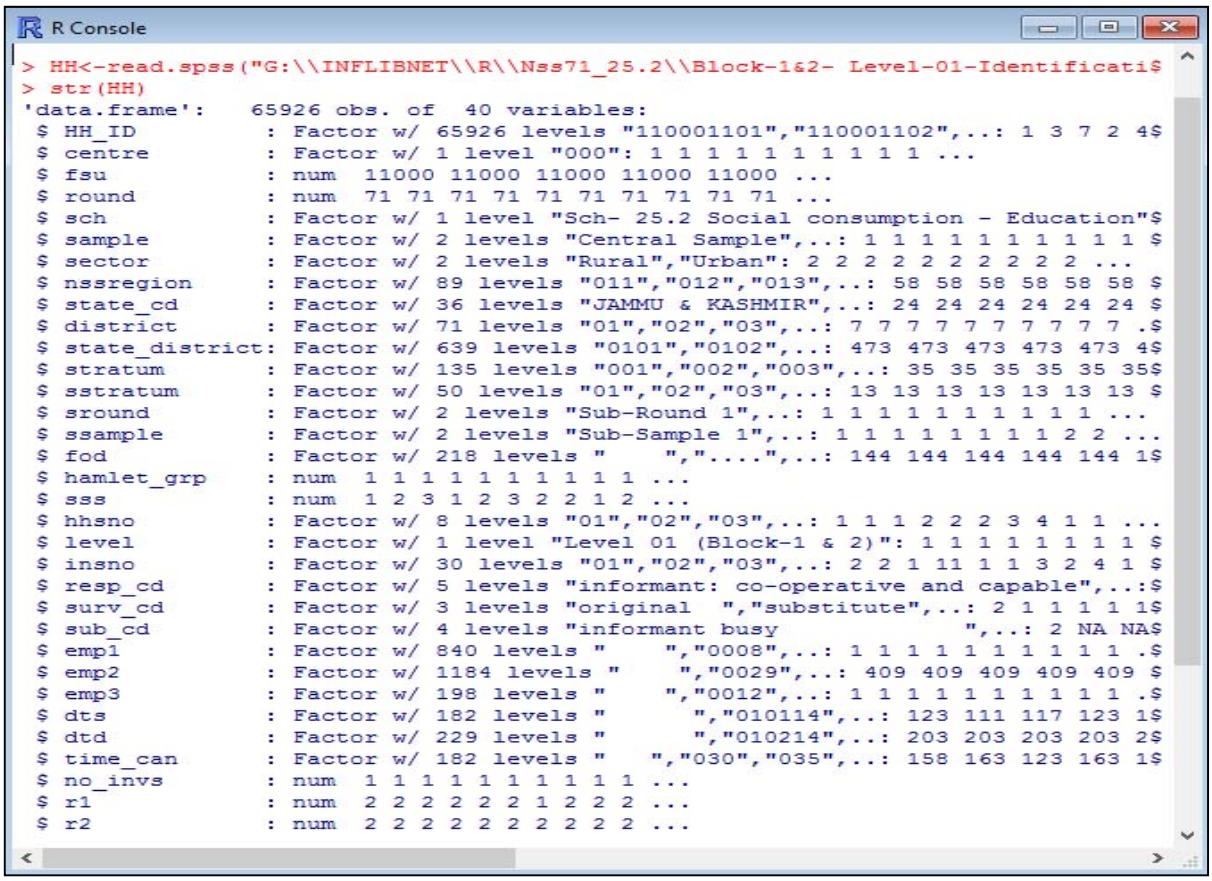
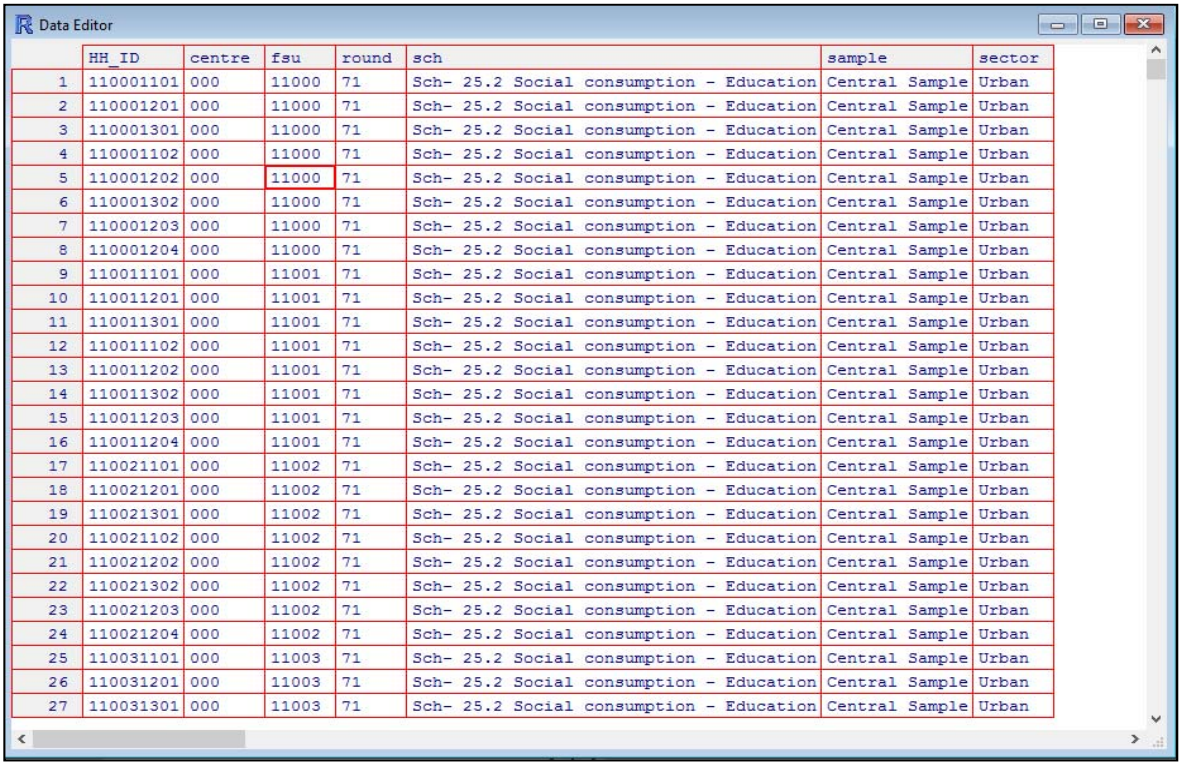


Fig. 8: Display of Data Structure

5.2 Fix Command

Fix command is used when data is to be edited. To edit data in a specific file, you need to write: "Fix(file name)". For example taken by us, it is written as "fix(HH)" which has produced Table 1 as a result. Here, the users may edit the data just by typing in the cells.



	HH_ID	centre	fsu	round	sch	sample	sector
1	110001101	000	11000	71	Sch- 25.2 Social consumption - Education	Central Sample	Urban
2	110001201	000	11000	71	Sch- 25.2 Social consumption - Education	Central Sample	Urban
3	110001301	000	11000	71	Sch- 25.2 Social consumption - Education	Central Sample	Urban
4	110001102	000	11000	71	Sch- 25.2 Social consumption - Education	Central Sample	Urban
5	110001202	000	11000	71	Sch- 25.2 Social consumption - Education	Central Sample	Urban
6	110001302	000	11000	71	Sch- 25.2 Social consumption - Education	Central Sample	Urban
7	110001203	000	11000	71	Sch- 25.2 Social consumption - Education	Central Sample	Urban
8	110001204	000	11000	71	Sch- 25.2 Social consumption - Education	Central Sample	Urban
9	110011101	000	11001	71	Sch- 25.2 Social consumption - Education	Central Sample	Urban
10	110011201	000	11001	71	Sch- 25.2 Social consumption - Education	Central Sample	Urban
11	110011301	000	11001	71	Sch- 25.2 Social consumption - Education	Central Sample	Urban
12	110011102	000	11001	71	Sch- 25.2 Social consumption - Education	Central Sample	Urban
13	110011202	000	11001	71	Sch- 25.2 Social consumption - Education	Central Sample	Urban
14	110011302	000	11001	71	Sch- 25.2 Social consumption - Education	Central Sample	Urban
15	110011203	000	11001	71	Sch- 25.2 Social consumption - Education	Central Sample	Urban
16	110011204	000	11001	71	Sch- 25.2 Social consumption - Education	Central Sample	Urban
17	110021101	000	11002	71	Sch- 25.2 Social consumption - Education	Central Sample	Urban
18	110021201	000	11002	71	Sch- 25.2 Social consumption - Education	Central Sample	Urban
19	110021301	000	11002	71	Sch- 25.2 Social consumption - Education	Central Sample	Urban
20	110021102	000	11002	71	Sch- 25.2 Social consumption - Education	Central Sample	Urban
21	110021202	000	11002	71	Sch- 25.2 Social consumption - Education	Central Sample	Urban
22	110021302	000	11002	71	Sch- 25.2 Social consumption - Education	Central Sample	Urban
23	110021203	000	11002	71	Sch- 25.2 Social consumption - Education	Central Sample	Urban
24	110021204	000	11002	71	Sch- 25.2 Social consumption - Education	Central Sample	Urban
25	110031101	000	11003	71	Sch- 25.2 Social consumption - Education	Central Sample	Urban
26	110031201	000	11003	71	Sch- 25.2 Social consumption - Education	Central Sample	Urban
27	110031301	000	11003	71	Sch- 25.2 Social consumption - Education	Central Sample	Urban

Table 1: Use of "fix command" to Edit Data

5.3 Summary Command

The "summary" command provides basic statistics of various variables, like minimum value, 1st quartile, median, mean, 3rd quartile and maximum value. To perform the above basic statistics with data, the following command to be used: **summary and within brackets the name of dataset.** In this case it is HH, so the command is written as "**summary (HH)**"which is shown below in Table 2.

```
> summary(HH)
       HH_ID          centre            fsu             round
 110001101:     1   000:65926   Min.   :11000   Min.   :71
 110001102:     1               1st Qu.:15650   1st Qu.:71
 110001201:     1               Median :31547   Median :71
 110001202:     1               Mean   :26072   Mean   :71
 110001203:     1               3rd Qu.:35922   3rd Qu.:71
 110001204:     1               Max.   :38880   Max.   :71
 (Other)   :65920
```

Table 2: Use of "summary command"

5.4 Head Command

Head command provides the first few entries of each variable. It is shown below in Table3, where the first 6 entries of variables are shown. The command should be "head" and "name of dataset within bracket", for e.g. head (HH). If, you are interested to view more than 6 entries, specifying the numbers of entries. For example, if you are interested to view the first 10 entries of variable, use the following command:
head (HH, n=10)

The result will be same as shown in Table 4, where the first 10 entries of variables are shown. Likewise, you may also see the last few entries of the variables by using tail command, i.e. **tail(HH, n=10).**

```
> head(HH)
      HH_ID centre   fsu round                                          sch
1 110001101    000 11000    71 Sch- 25.2 Social consumption - Education
2 110001201    000 11000    71 Sch- 25.2 Social consumption - Education
3 110001301    000 11000    71 Sch- 25.2 Social consumption - Education
4 110001102    000 11000    71 Sch- 25.2 Social consumption - Education
5 110001202    000 11000    71 Sch- 25.2 Social consumption - Education
6 110001302    000 11000    71 Sch- 25.2 Social consumption - Education
```

Table 3: Use of "head command"

```
> head(HH,n=10)
      HH_ID centre   fsu round                                          sch
1  110001101    000 11000    71 Sch- 25.2 Social consumption - Education
2  110001201    000 11000    71 Sch- 25.2 Social consumption - Education
3  110001301    000 11000    71 Sch- 25.2 Social consumption - Education
4  110001102    000 11000    71 Sch- 25.2 Social consumption - Education
5  110001202    000 11000    71 Sch- 25.2 Social consumption - Education
6  110001302    000 11000    71 Sch- 25.2 Social consumption - Education
7  110001203    000 11000    71 Sch- 25.2 Social consumption - Education
8  110001204    000 11000    71 Sch- 25.2 Social consumption - Education
9  110011101    000 11001    71 Sch- 25.2 Social consumption - Education
10 110011201    000 11001    71 Sch- 25.2 Social consumption - Education
```

Table 4: Use of "head command"

5.5 Is command

"ls" command provides name of variables in the dataset. Write "ls" and "name of dataset" as shown below:
ls (HH)

Resultant Table 5 is reproduced below.

```
> ls(HH)
 [1] "centre"         "district"       "dtd"            "dts"
 [5] "emp1"           "emp2"           "emp3"           "fod"
 [9] "fsu"            "hamlet_grp"     "HH_ID"          "hhsno"
[13] "insno"          "level"          "mlt"            "no_invs"
[17] "nsc"            "nss"            "nssregion"      "r1"
[21] "r2"             "r3"             "r4"             "resp_cd"
[25] "round"          "sample"         "sch"            "sector"
[29] "sround"         "ssample"        "sss"            "sstratum"
[33] "state_cd"       "state_district" "stratum"        "sub_cd"
[37] "surv_cd"        "time_can"       "wgt_combined"   "wgt_ss"
```

Table 5: Use of "ls command"

5.6 Class Command

"class" command is used to know the type of variables. Use of this command is as follows:
class (name of variable).

It will show the type of variable, i.e. numeric, factor, etc.

5.7 Levels command

The "levels" command is used only in factor type of variables. For example, sector contains rural and urban categories, similarly other variables may have more than two categories. As such, to find out those categories, use the command given below:
levels (variable name)

5.8 As.factor command

The "as.factor" command helps in changing one type of variable to other types of variable, such as factor or categorical variable. For example, here sector is categorical variable, but in "R" it will be considered as numeric variable because the values are assigned to different categories. As such, to change its type, the numeric variable needs to be converted into factor variable. Use the following command for this purpose:
sector<-as.factor(sector)

This command will convert the numeric variable to factor.

5.9 Dim Command

The "dim" command reveals number of rows and columns in a data sets. The command is used as follows:
dim(Dataset name), for e.g "**dim(HH)"**

This command will produce the numbers of rows and columns respectively.

5.10 Length Command

The "length" command is used to find out numbers of variables in the dataset. The command is to be written as follows:
length (dataset name)

6. Opening txt file in R

To open a fixed delimited file in R, "read.fwf" command is used. Most of the datasets are provided in text fixed delimited form which need to be broken according to the width provided along with the dataset. The command to be used for opening txt file is R is as follows:
read.fwf (file, widths, header =TRUE, sep = "\t", skip = 0, n = -1, buffersize = 2000)

In this command, "file" implies the name of file in the local computer hard disc, "header" is whether the data has variable names, if yes, write "TRUE". "Sep" means the character or the separator used internally, "skip" means how many initial lines user wishes to skip, and "buffersize" means maximum number of lines to be read from the data file.

Above-mentioned command will lead to uploading of data file into R. Further, the user are required to check carefully whether the data uploaded into R is correctly uploaded or not.

7. Contingency Table or Cross Tabulation

Cross tabulation is required to find out the relationship between two categorical variables. In order to do the cross tabulation in R, “table” command is used. For example, to see the relation between the sector and education, use the following command:
table name <- table (file name $variable name, file name $variable name)

In the example taken here, it is: **HH.tab<-table(HH $sector, HH $education).** Then, type the name of table i.e. **HH.tab,** to produce the result.

This command will produce the result as shown in Table 6, where rural and urban as well as type of education is given in codes.

```
> HH.tab<-table(HH$sector, HH$education)
> HH.tab

            1     2
  Rural  2958 33500
  Urban  2721 26706
>
```

Table 6: Calculation of Cross Tabulation- 1

Similarly, you may also calculate the row percentage and column percentage using **prop.table** command. For the example used here, the command to be used for calculating the row percentage is as follows:
round(prop.table (HH.tab,1), 2)

This command will produce the result as shown in Table 7. Here, "HH.tab" is the table name, assigned to earlier Table 6.

```
> round(prop.table(HH.tab,1), 2)

           1    2
  Rural 0.08 0.92
  Urban 0.09 0.91
>
```

Table 7: Calculation of Cross Tabulation- 2

To calculate column percentage, use the command:
round (prop.table (HH.tab,2),2)

This command will produce the result as shown in Table 8.

```
> round(prop.table(HH.tab,2),2)

           1    2
  Rural 0.52 0.56
  Urban 0.48 0.44
>
```

Table 8: Calculation of Cross Tabulation- 3

You may also calculate the "chi square" statistics for the two variables by using "**chisq.test(HH.tab)**"command. Resultant output produced is given in Fig. 9.

```
> chisq.test(HH.tab)

        Pearson's Chi-squared test with Yates' continuity correction

data:  HH.tab
X-squared = 26.402, df = 1, p-value = 2.772e-07

>
```

Fig. 9: Calculation of Chi Square

7.1 Calculating Three Way Table

Similar to two way tables, you may also calculate three way tables, using "table" command. For example, to prepare the table for level of education, marital status and sex, use the following command:

table<-table(HHgen_edu, HHmarital_status, HH$sex)

Resultant output is given in Table 9, where rows represent the level of education and columns represent the marital status. Level of education and marital status are shown separately for male in column1 and female in column 2. Similarly, you may also prepare four way, five way and so on tables.

```
> table<-table(HH$gen_edu, HH$marital_status, HH$sex)
> table
, ,  = 1

    1     2     3     4
1  14671 13622  1381    79
2     92   440    33     3
3    332    33     0     0
4     59    25     1     0
5    145   203    13     0
6  19165  5903   279    28
7  12208  9454   370    40
8  10968 12703   280    56
10  9671 10760   314    39
11 11139  6979   118    19
12   570   644    15     2
13   927   553     8     5
14   319   430     8     1
15  4070  7936   123    16
16   693  2566    48     9

, ,  = 2

    1     2     3     4
1  13309 24769  7303   191
2     69   363    90     2
3    301    31     8     0
4     55    34     7     3
5     70   198    33     1
6  15845  5853   962    80
7   9323 10071  1001   110
8   7927 11201   710    79
10  6176  9086   511    81
11  6613  5591   230    33
```

Table 9: Calculation of Three way Table

8. Select Cases

In R, you may perform calculation for selected cases of a variable. For example, in order to calculate mean age of males from a dataset, use the following command:

mean(HH $age[HH $sex==1])

Here "HH" is the name of dataset and "Sex==1" implies male only. Result of the output is shown in Fig. 10. Likewise, you may also calculate it for females.

```
> mean(HH $age[HH $sex==1])
[1] 28.48534
```

Fig. 10: Calculation of Selected Cases of a Variable

9. Data Separation

The "data separation" function in R, is used for separating the dataset for specific values. For example, to separate a given dataset into different age group, i.e. people above 50 years of age and people below 50 years, the following command is to be written:
HH$agecat <- ifelse(HH$age > 50,c("older"), c("younger"))

The above mentioned command will separate the age into older and younger people as shown in Table 10.

[78828]	"younger"	"older"	"younger"	"younger"	"younger"	"younger"	"younger"
[78835]	"younger"	"younger"	"younger"	"younger"	"younger"	"younger"	"older"
[78842]	"older"	"younger"	"younger"	"younger"	"younger"	"younger"	"younger"
[78849]	"younger"	"younger"	"younger"	"younger"	"younger"	"younger"	"younger"
[78856]	"younger"	"younger"	"younger"	"younger"	"younger"	"younger"	"younger"
[78863]	"younger"	"younger"	"younger"	"younger"	"younger"	"younger"	"older"
[78870]	"younger"	"younger"	"younger"	"younger"	"younger"	"younger"	"younger"
[78877]	"younger"	"younger"	"younger"	"younger"	"younger"	"younger"	"younger"
[78884]	"older"	"older"	"older"	"younger"	"younger"	"younger"	"younger"
[78891]	"younger"	"younger"	"younger"	"younger"	"younger"	"younger"	"younger"
[78898]	"younger"	"older"	"older"	"younger"	"younger"	"younger"	"younger"
[78905]	"younger"	"younger"	"younger"	"younger"	"younger"	"younger"	"younger"
[78912]	"younger"	"younger"	"younger"	"younger"	"younger"	"younger"	"younger"
[78919]	"older"	"younger"	"older"	"younger"	"older"	"older"	"younger"
[78926]	"younger"	"younger"	"younger"	"younger"	"younger"	"younger"	"younger"
[78933]	"younger"	"younger"	"older"	"younger"	"younger"	"younger"	"younger"
[78940]	"younger"	"younger"	"younger"	"younger"	"younger"	"younger"	"younger"
[78947]	"younger"	"younger"	"younger"	"younger"	"younger"	"younger"	"younger"
[78954]	"older"	"older"	"younger"	"younger"	"younger"	"younger"	"younger"
[78961]	"older"	"younger"	"younger"	"younger"	"younger"	"younger"	"younger"
[78968]	"younger"	"younger"	"younger"	"younger"	"younger"	"younger"	"younger"
[78975]	"younger"	"younger"	"younger"	"younger"	"younger"	"younger"	"younger"
[78982]	"younger"	"younger"	"younger"	"younger"	"younger"	"younger"	"younger"
[78989]	"younger"	"younger"	"younger"	"younger"	"younger"	"younger"	"younger"
[78996]	"younger"	"younger"	"older"	"older"	"younger"	"younger"	"younger"
[79003]	"younger"	"younger"	"younger"	"younger"	"younger"	"younger"	"younger"
[79010]	"older"	"older"	"younger"	"younger"	"younger"	"younger"	"younger"
[79017]	"younger"	"younger"	"younger"	"younger"	"younger"	"younger"	"younger"
[79024]	"younger"	"older"	"older"	"younger"	"younger"	"younger"	"younger"
[79031]	"younger"	"younger"	"younger"	"younger"	"younger"	"younger"	"younger"
[79038]	"younger"	"younger"	"younger"	"younger"	"older"	"older"	"older"
[79045]	"younger"	"younger"	"younger"	"younger"	"younger"	"younger"	"younger"
[79052]	"younger"	"older"	"older"	"younger"	"younger"	"younger"	"younger"
[79059]	"younger"	"younger"	"younger"	"younger"	"younger"	"younger"	"younger"
[79066]	"younger"	"younger"	"younger"	"younger"	"younger"	"older"	"older"

Table 10: Use of Data Separation Function

10. Data Transformation

Data transformation command changes the values of variables or observations through mathematical calculations, such as: subtraction, addition, multiplication and so on. For calculating the square root of a variable (e.g. age), use the following command:
HH$transformed_age <- sqrt(HH$age).

Here, "**HH**" is the name of dataset and "**sqrt"** is the function to be used for square root. This will produce a new variable with the name of "**transformed_age"** as shown in Table 11.

edu_attend	student_hostel	wgt_combined	transfor>
NA	NA	1071.655	5.656854
NA	NA	1071.655	5.477226
6	NA	1071.655	2.44949
NA	NA	1071.655	1.732051
NA	NA	1071.655	6.164414
7	2	1071.655	3.741657
NA	NA	1071.655	1.414214
7	2	1071.655	3.464102
6	NA	1071.655	2.645751

Table 11: Changing of Value with Use of Data Transformation Command

Likewise, you may also calculate the log of a variable by using the following command:
HH$log_age <- log(HH$age).

As a result, a new variable with the name of "**log_age**" will be created as shown in Table 12.

student_hostel	wgt_combined	transformed_age	log_age
NA	1071.655	5.656854	3.465736
NA	1071.655	5.477226	3.401197
NA	1071.655	2.44949	1.791759
NA	1071.655	1.732051	1.098612
NA	1071.655	6.164414	3.637586
2	1071.655	3.741657	2.639057
NA	1071.655	1.414214	0.6931472
2	1071.655	3.464102	2.484907
NA	1071.655	2.645751	1.94591

Table 12: Calculation of Log of a Variable

Transformation function in R also provides the mathematical computation between two variables. For example, to multiply the age values with the values assign in sex category, use the following command:
HH$S_age <- (HH$age*HH$sex)

As a result, a new variable with the name "**S_age**" will be created as shown in Table 13.

wgt_combined	transformed_age	log_age	l_age	S_age
1071.655	5.656854	3.465736	96	32
1071.655	5.477226	3.401197	90	60
1071.655	2.44949	1.791759	18	6
1071.655	1.732051	1.098612	9	6
1071.655	6.164414	3.637586	114	76
1071.655	3.741657	2.639057	42	14
1071.655	1.414214	0.6931472	6	2
1071.655	3.464102	2.484907	36	24
1071.655	2.645751	1.94591	21	14

Table 13: Computation between Two Variables with the Use of Data Transformation

Likewise, the values of above two variables may also be divided, using the following command:
HH$SS_age <- (HH$age/HH$sex).

As a result, new variable with the name of "**SS_age**" will be created as shown in Table 14.

wgt_combined	transformed_age	log_age	l_age	S_age	SS_age
1071.655	5.656854	3.465736	96	32	32
1071.655	5.477226	3.401197	90	60	15
1071.655	2.44949	1.791759	18	6	6
1071.655	1.732051	1.098612	9	6	1.5
1071.655	6.164414	3.637586	114	76	19
1071.655	3.741657	2.639057	42	14	14
1071.655	1.414214	0.6931472	6	2	2
1071.655	3.464102	2.484907	36	24	6
1071.655	2.645751	1.94591	21	14	3.5

Table 14: Division of Values of Two Variables Using Data Transformation Command

11. Sort Case

The "sort case" function in R sort the data in ascending or descending order. For example, to arrange the data in ascending order, use the following command:
newdata <- HH[order(age),]

The above mentioned command will create a new dataset with the name "**newdata**" where the data is arranged in ascending order of age as shown in Table 15.

relation_to_head	sex	age	marital_status	gen_edu
5	2	0	1	1
6	2	0	1	1
6	2	0	1	1
5	1	0	1	1
6	1	0	1	1

Table 15: Use of "sort case" for Ascending Order of Data- 1

In order to sort the data in ascending order of age and descending order of sex, use the following command:
newdata1<- HH[order(age, -sex),]

As a result, the new dataset will be created with the name of "**newdata1**" as shown in Table 16.

relation_to_head	sex	age	marital_status	gen_edu
5	2	0	1	1
6	2	0	1	1
6	2	0	1	1
6	2	0	1	1
6	2	0	1	1
5	2	0	1	1
5	2	0	1	1
6	2	0	1	1
6	2	0	1	1
5	2	0	1	1
5	2	0	1	1

Table 16: Use of "sort case" for Ascending Order of Data- 2

12. Calculation of Decile, Quartile and Percentile

In NSS published reports, many results are shown in decile form. R provides the command to calculate the decile of a particular variable. To calculate the decile of a given variable, first the software package of "**dplyr**" is to be uploaded in the R. After uploading it, use the following command:
library (dplyr).

The above mentioned command will install the software package in R. After that, use the following command:
HHH<-HH %>% mutate(decile = ntile(age, 10)).

As a result, a new dataset with the name of "**HHH**" will be created, containing decile class of age with the name of decile as shown in Table 17.

log_age	l_age	S_age	SS_age	qunatil	decile
3.465736	96	32	32	(26,32]	6
3.401197	90	60	15	(26,32]	6
1.791759	18	6	6	(0,6]	1
1.098612	9	6	1.5	(0,6]	1
3.637586	114	76	19	(32,39]	7
2.639057	42	14	14	(12,16]	3
0.6931472	6	2	2	(0,6]	1
2.484907	36	24	6	(6,12]	2
1.94591	21	14	3.5	(6,12]	2

Table 17: Calculation of Decile

Similarly, by using **Q_HH<-HH %>% mutate(quartile = ntile(age, 4))**command, you can change a variable into four equal parts as shown in Table 18.

log_age	l_age	S_age	SS_age	qunatil	quartile
3.496508	99	33	33	(32,39]	3
3.401197	90	60	15	(26,32]	3
2.079442	24	8	8	(6,12]	1
1.791759	18	6	6	(0,6]	1
2.302585	30	20	5	(6,12]	1
2.197225	27	18	4.5	(6,12]	1
4.382027	240	160	40	(55,112]	4
3.401197	90	60	15	(26,32]	3
1.386294	12	4	4	(0,6]	1
1.098612	9	3	3	(0,6]	1
0.6931472	6	2	2	(0,6]	1
3.806662	135	45	45	(39,45]	4
3.688879	120	80	20	(39,45]	3
2.890372	54	36	9	(16,20]	2

Table 18: Calculation of Quartile

Likewise, to calculate percentile, use the following command:
P_HH<-HH %>% mutate(percentile = ntile(age, 100))

This will create the new dataset with "**P_HH**" name, based on percentile of age as shown in Table 19.

log_age	l_age	S_age	SS_age	qunatil	percentile
3.465736	96	32	32	(26,32]	59
3.401197	90	60	15	(26,32]	55
1.791759	18	6	6	(0,6]	9
1.098612	9	6	1.5	(0,6]	4
3.637586	114	76	19	(32,39]	68
2.639057	42	14	14	(12,16]	25
0.6931472	6	2	2	(0,6]	3
2.484907	36	24	6	(6,12]	20
1.94591	21	14	3.5	(6,12]	11

Table 19: Calculation of Percentile

13. Data Aggregation

Data aggregation function in R deals with gathering the data for numerical variable based on some categorical variables. For example, to calculate the total expenditure of people belonging to different age and different sector, use the following command:

aggregatdata <-aggregate(Data$tot_exp, by=list(sector, age), FUN=sum, na.rm=TRUE)

In this example, total expenditure is being aggregated (summed-up) based on sector and age. As a result, new dataset named "aggregatedata" will be created as shown in Table 20.

In Table 20, Group.1 shows the sector (rural and urban), Group. 2 represent age of individual, and "x" represent the total expenditure of individuals.

Group.1	Group.2	x	var4	var5
1	5	2869615		
2	5	5542863		
1	6	7911490		
2	6	16509975		
1	7	9888436		
2	7	21852465		
1	8	11142148		
2	8	24572101		
1	9	9127970		
2	9	18124514		
1	10	13167549		

Table 20: Data Aggregation of Variable

Similarly, data can also be aggregated by calculating the mean of total expenditure of individuals using the following command:

aggregatedata <-aggregate(Data$tot_exp, by=list(sector, age), FUN=mean,na.rm=TRUE)

As a result, a new dataset will be created named "aggregatedata" as shown in Table 21 wherein Group1 and Group2 represent the sector and age of individual respectively and "x" represents the mean expenditure of individuals.

Group.1	Group.2	x	var4	var5
1	5	3160.369		
2	5	8633.743		
1	6	3028.901		
2	6	9582.11		
1	7	3140.183		
2	7	10230.55		
1	8	2860.628		
2	8	10120.31		
1	9	3256.5		
2	9	9980.459		
1	10	3080.128		
2	10	10434.33		
1	11	3484.827		

Table 21: Data Aggregation of Variable

14. Recoding

The recoding of data command is used when the two or more cases of a variable are to be combined. In order to recode variable, you need to install “car” software package, using **install(car)** command and **library(car)** command. After uploading the car package use the following command:

Data$age.rec <- recode(Data$age,"18:19='18to19';20:29='20to29';30:39='30to39'; else=1").

In the above command, age variable for 18:19 as 18to19, 20:29 as 20 to 29 and so on is being recoding. As result, a new variable with the name of "**age.rec**" will be created in the last column as shown in Table 22.

tot_exp	hostel	nss	nsc	mlt	wgt_ss	wgt_combined	age.rec
16800	NA	1	2	214331	2143.31	1071.655	1
1780	NA	1	2	214331	2143.31	1071.655	1
1520	NA	1	2	214331	2143.31	1071.655	1
1520	NA	1	2	214331	2143.31	1071.655	1
1430	NA	1	2	214331	2143.31	1071.655	1
1400	NA	1	2	214331	2143.31	1071.655	1
9300	NA	1	2	843642	8436.42	4218.21	18to19
9200	NA	1	2	843642	8436.42	4218.21	18to19
1350	NA	1	2	843642	8436.42	4218.21	1
1830	NA	1	2	843642	8436.42	4218.21	20to29
2055	NA	1	2	205210	2052.1	1026.05	1
56500	NA	1	2	5913	59.13	29.565	20to29
56500	NA	1	2	5913	59.13	29.565	20to29
6000	NA	1	2	5913	59.13	29.565	20to29
5700	NA	1	2	130077	1300.77	650.385	18to19

Table 22: Recoding Variables

15. Merge Data

“Merge" command is used to merge two variables from two different files. In other words, merge command is used to add a variable in the dataset. In this example, variables from two different datasets named "Data" and "Data2" are being merged using the command given below:

merged_data <- merge(Data,Data2,by="HH_ID")

Using above mentioned command, two datasets are merged using a common ID i.e. "HH_ID" as shown in Table 23. Similarly, to add rows of the two datasets, i.e. "Data" and "Data2", use the following command:

merged_data <- rbind(Data,Data2).

Then, the function "**rbind**" is to be used to merge the two datasets.

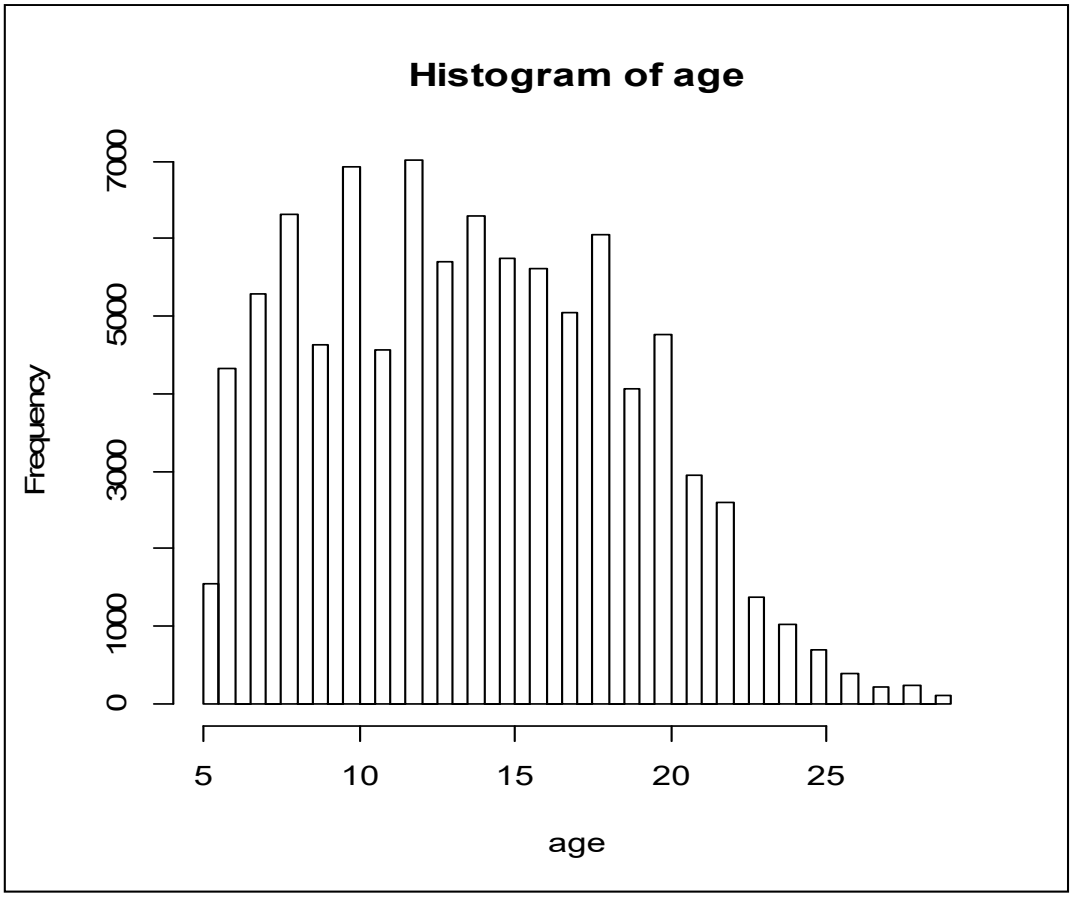
HH_ID	psrl_no.x	relation_to_head	sex	age.x	marital_status
110001101	01	1	1	53	2
110001101	01	1	1	53	2
110001101	03	5	2	20	1
110001101	03	5	2	20	1
110001101	04	5	2	19	1
110001101	04	5	2	19	1
110001101	02	2	2	52	2
110001101	02	2	2	52	2
110001102	04	4	2	32	2
110001102	09	6	2	18	1
110001102	02	2	2	57	2
110001102	03	3	1	37	2

Table 23: Merging Data from Two Variables

16. Graphs

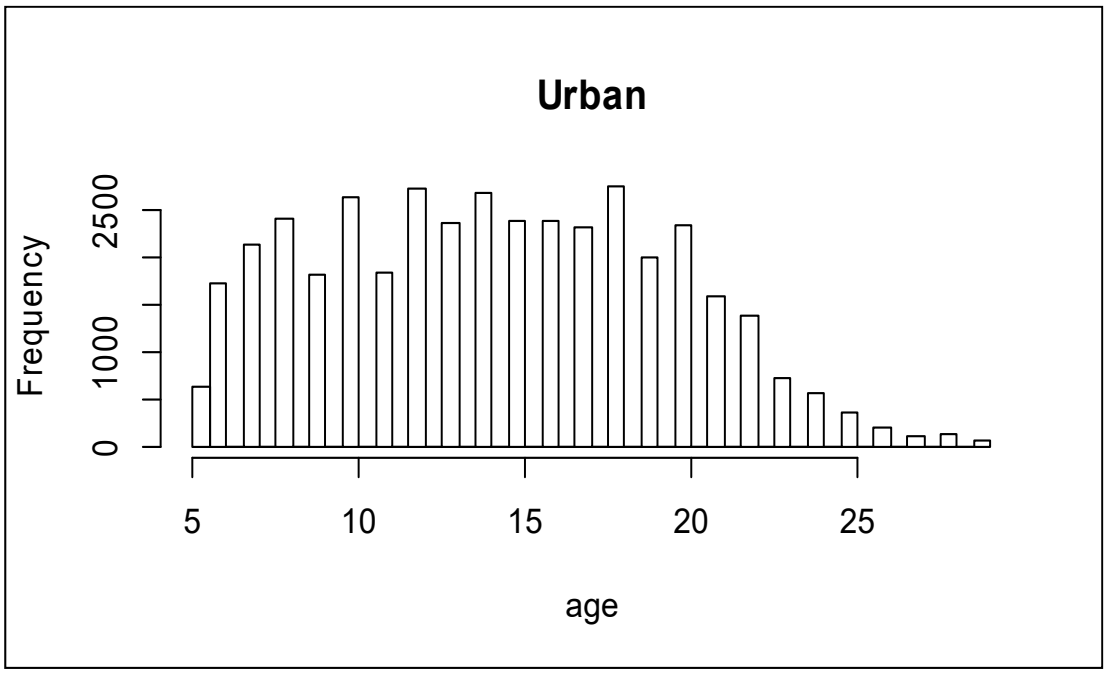
In R, you may draw a wide ranges of graph, e.g. histogram, bar, line, area, scatter, plot, etc. Histograms are shown in Graph 1, 2, 3, and 4. Graph1 is obtained through **hist(age)**command and Graph2 is obtained using **hist(age,breaks="FD")** command, where FD is used to automatically adjust the width of graph.

Graph 1: Drawing of Histogram

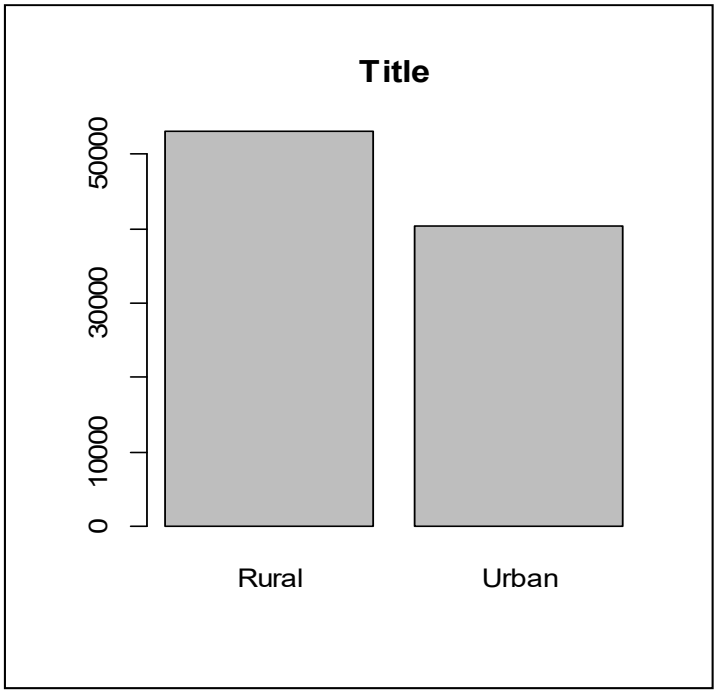


Graph 2: Drawing of Histogram

Graph 3 is drawn only for "urban" sector using **hist(age[sector=="2"],breaks="FD", main="Urban",xlab="age")** command, and Graph 4 is drawn using **hist(age[sector],breaks="FD",xlab="age")** command to show the frequency of both urban and rural sectors.



Graph 3: Drawing of Histogram

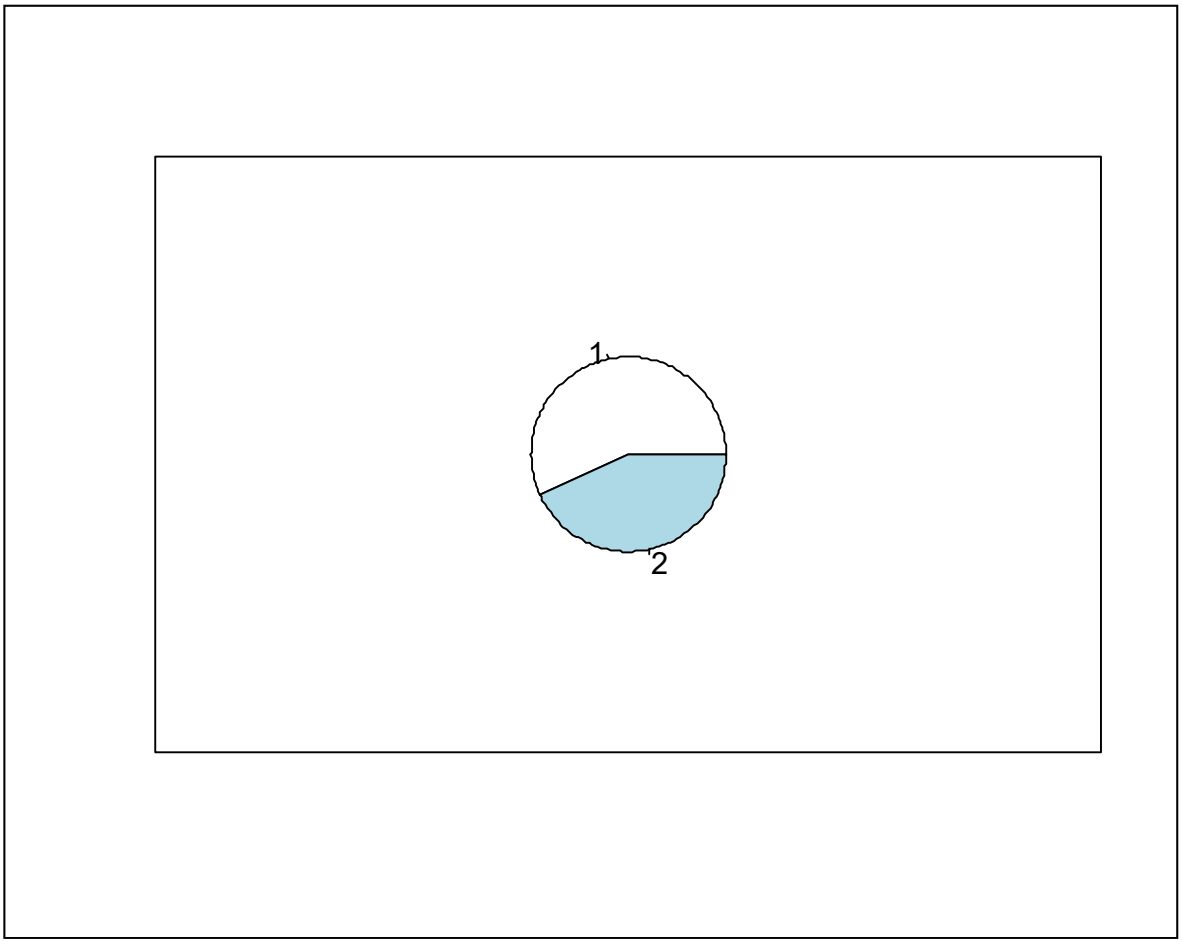


Graph 4: Drawing of Histogram

Similarly, you may also prepare the pie chart in R to exhibit the percentage of variables. In this example,pie chart for sector (rural and urban) is calculated using the command given below:

- **table(sector)**
- **count<-table(sector)**
- **pie(count)**
- **box()**

With the use of "table" command, first table of sector is prepared followed by "count" command to assign name to that table, i.e. count. "pie(count)" command is used to actually draw the Pie chart as shown in Fig. 11. Here, in the pie chart, code 1 represents rural sector and code 2 represents urban sector. Finally, the command box(), is used to put the pie chart in a box.



Pie chart 1: Showing the Percentage of Variables

For further information please contact

Dr. Upendra Choudhury
Member-Secretary
Indian Council of Social Science Research
Aruna Asaf Ali Marg
New Delhi - 110067
E-mail: ms@icssr.org

Dr. Jagdish Arora
Director
Information and Library Network Centre
An IUC of UGC
Infocity, Gandhinagar -382007, Gujarat
E-mail: director@inflibnet.ac.in

Indian Council of Social Science Research
JNU Institutional Area, Aruna Asaf Ali Marg
New Delhi - 110067 (INDIA)

For more information : http://www.icssrdataservice.in/.