

# SSER Workshop on Using Large-scale Survey Data

Notes on R Programming Language  
December 16-23, 2016

Vikas Rawal





# Day 1: How to Read Data into R

## Why should you use programming for statistical analysis? Why use R?

- Programming is crucial for reproducibility and verifiability of your research
- R is open-source and free of cost
  - Will always remain available
  - Development is fast; tools for using new, cutting-edge statistical techniques become available as add-on packages.

## How to read data in R

### Reading fixed-width text files

- NSS/ASI data are given as fixed-width, text files.
- First look at the survey schedule and instructions, and understand which variables are of interest to you. The survey schedule is divided into blocks.
- Look at the layout file to see which data files (levels) have the data you are interested in.
- The layout file gives you byte position of each variable in each file.
- In R, `read.fwf` can be used to read fixed-width files.

### Reading other types of files.

- You often need to read spreadsheet files into R. Excel is notoriously messy. Use your spreadsheet application to save such files as csv (comma separated value files), using a proper delimiter (often better to use `|` instead of a comma), and import into R using `read.table` command.
- You can also use `readxl` package to directly read excel files into R
- You can use `read.sta` to read stata files, `read.spss` to use spss files directly into R.

### Box 1 Reading other types of files

The following command reads a file called R6810L01.TXT into R and saves it as a data frame (called newobj here). Note that everything in R is case sensitive. So use upper and lower cases correctly.

```
1 read.fwf("R6810L01.TXT",
2          c(3,5,2,3,1,1,3,2,2,2,1,1,1,4,1,1,2,2,5,2,1,1,1,
3            6,6,3,3,1,1,1,1,2,55,3,3,10,3,3,10),
4          col.names = c("roundcode","fsunumber","round","schedule_number",
5                        "sample", "sector","state_region","district",
6                        "stratum_number", "sub_stratum","filler1",
7                        "sub_round","sub_sample","fod_sub_region",
8                        "hamlet_group","second_stage_stratum","hhs_no",
9                        "level","filler2","informant_sno","response_code",
10                       "survey_code","subsitution_code","survey_date",
11                       "despatch_date","canvas_time","canvas_time_block8",
12                       "remarks1_block9","remarks2_block9",
13                       "remarks1_elsewhere","remarks2_elsewhere",
14                       "special_characters","blank","NSS","NSC","MLT",
15                       "nss_sr","nsc_sr","mlt_sr"))->level1
```

- The command has three arguments: name of the file to be read, the width of each column in the file, and variable name to be given to each column.
- The width of each column is given in the layout file for NSS/ASI data
- There are no strict naming conventions for variable names. But give variable names sensibly. Do not use spaces and other special characters in names. Variable names are also case sensitive.

### Box 2 Reading other types of files

The following command reads a file called filename.csv into R and saves it as a data frame (called newobj here)

```
1 read.table("filename.csv",sep="|",header=T)->newobj
```

- argument sep tells the command which character should be used as a delimiter.
- argument "header=T" tells the command that the first line of the file has the variable names.

## Day 2: Working with Dataframes

### Dataframes and vectors

- Dataframes are the main objects in which you store data.
- Dataframes are data tables with variables in columns and observations in rows.

- Each column/variable is of a particular type. There are three main types: numeric, character and factor.
  - character variables are text strings,
  - numeric variables are numbers
  - factors are categorical variables, in which each observation belongs to a category. Category labels may be numeric or characters. Even when they are numeric, they are just labels for categories and should not be confused as numbers.

### Box 3 Changing variable type

Unless you specify while importing data into R, R guesses the type of vector for each variable. Sometimes the guessed data type may not be what you want, and you may have to change it. Typical use cases are as follows.

```
1 class(level1$sector)
2 as.character(level1$sector)->level1$sector
3 as.numeric(level1$sector)->level1$sector
4 as.numeric(as.character(level1$sector))->level1$sector
```

- Line 1 tells you vector type of variable sector in level1
- Line 2 converts variable sector in level1 to character and overwrites the original vector
- Line 3 converts character variable sector in level1 to numeric and overwrites the original vector
- Line 4 converts a factor with numeric labels into character, and then converts it into a numeric vector.

### Box 4 Logical operators in R

```
1 ifelse(level3$age<18,"Child","Adult")->level3$agegroup1
2 ifelse(level3$age<18,"Child",
3       ifelse(level3$age<60,"Working-age adult","Old person"))->level3$agegroup2
```

Operator	Meaning	Example
==	equal to	level3\$age==10
!=	equal to	level3\$age!=10
>, >=	greater than/equal	level3\$age>=60
<, <=	less than/equal	level3\$age<18
is.na	is blank	is.na(level3\$age)
!is.na	is not blank	!is.na(level3\$age)
%in%	is part of the sequence	level3\$state %in% c("Haryana","Punjab")
	OR	level3\$age<18   level3\$age>=60
&	AND	level3\$age<18 & level3\$sex==1

## Mergng data

Different items of data collected from a sample unit (household or firm, for example) are disaggregated at different levels.

- household-level information,
- information on individuals,
- information on each occupation of an individual,
- information on each plot of land,
- information on each item of consumption of a household,
- information on each input used by a firm

Such data are provided in separate levels/files.

When you need to put together information provided in different data tables, you would need to merge data frames that have those tables.

There are three types of merges/joins

- inner: matching rows from two data frames are merged; non-matching rows are dropped.
- one-sided: matching rows from two data frames are merged; non-matching rows from one data frame are included; non-matching rows from the other data frame are dropped.
- outer: all rows are included; where possible, rows are matched; where matching information is not found, blank cells are created.

### Box 5 Merging dataframes

```
1 ; inner join
2 merge(level2,level3,
3       by=c("fsunumber","hamlet_group","second_stage_stratum","hhs_no"))->emp68
4
5 ; left join
6 merge(level2,level3,
7       by=c("fsunumber","hamlet_group","second_stage_stratum","hhs_no"),
8       all.x=T)->emp68
9
10 ; right join
11 merge(level2,level3,
12       by=c("fsunumber","hamlet_group","second_stage_stratum","hhs_no"),
13       all.y=T)->emp68
14
15 ; outer join
16 merge(level2,level3,
17       by=c("fsunumber","hamlet_group","second_stage_stratum","hhs_no"),
18       all.x=T)->emp68
```





**Published by**

Society for Social and Economic Research, New Delhi.

E-mail: [office@sser.in](mailto:office@sser.in)

©SSER, 2016

---

The Society for Social and Economic Research (SSER) is a charitable trust based in New Delhi, India. SSER undertakes research on a wide range of issues related to social and economic development. These include issues related to food security and nutrition, sustainable agricultural development and rural livelihoods, employment and labour relations, industrial development, and living conditions in rural and urban areas.