Lecture 2 - R Basics and Importing Data

Dr. Sajid Bashir

03 February, 2020

Contents

Agenda

- Importing data
- Simple summaries of categorical and continuous data
- Coding style
- Homework 1
- Lab 1

Importing Data

We will discuss with a survey results from a group of students. To import tabular data into R, we use the command `read.table()`

```
survey <- read.table("survey_students.csv", header=TRUE, sep=",")
```

Let's parse this command one component at a time

- Data is in a file `survey_students.csv`, in current working directory.
- File contains a `header` as its first row.
- `csv` means data is in `comma-separated format`, so `sep=","`

An equivalent command is `read.csv()`

Exploring Data

R imports data into a **`data.frame`** object.

```
class(survey)
```

```
## [1] "data.frame"
```

In order to view first few rows of data, use **`head()`**

```
head(survey, 5)
```

```
##   Program                PriorExp      Rexperience OperatingSystem TVhours
## 1      CS         Some experience       Never used        Mac OS X       2
## 2      CS Never programmed before       Never used         Windows      15
## 3      CS         Some experience Basic competence        Mac OS X      16
## 4   Other         Some experience Basic competence        Mac OS X       0
## 5      CS Never programmed before       Never used         Windows       2
##           Editor
## 1 Microsoft Word
## 2 Microsoft Word
## 3 Microsoft Word
## 4          LaTeX
## 5 Microsoft Word
```

`head(data.frame, n)` returns the first `n` rows of the data frame. In the Console, you can also use `View(survey)` to get a spreadsheet view

Simple Summary

Use the `str()` function to get a simple summary of your data set

```
str(survey)
```

```
## 'data.frame':    31 obs. of  6 variables:
##  $ Program        : Factor w/ 4 levels "CS","DS","Other",..: 1 1 1 3 1 1 1 2 1 3 ...
##  $ PriorExp       : Factor w/ 3 levels "Extensive experience",..: 3 2 3 3 2 2 3 3 3 2 ...
##  $ Rexperience    : Factor w/ 4 levels "Basic competence",..: 3 3 1 1 3 3 1 1 1 3 ...
##  $ OperatingSystem: Factor w/ 2 levels "Mac OS X","Windows": 1 2 1 1 2 2 1 2 2 2 ...
##  $ TVhours        : int  2 15 16 0 2 5 0 4 0 0 ...
##  $ Editor         : Factor w/ 4 levels "Excel","LaTeX",..: 3 3 3 2 3 3 3 3 3 3 ...
```

This says that TVhours is a numeric variable, while all the rest are factors (categorical)

Another Simple Summary

```
summary(survey)
```

```
##   Program                     PriorExp                Rexperience
##  CS   :14   Extensive experience   : 3   Basic competence    :14
##  DS   : 9   Never programmed before: 5   Installed on machine: 7
##  Other: 5   Some experience        :23   Never used          : 9
```

```
##  SW   : 3                              R Wizard            : 1
##
##
##  OperatingSystem    TVhours                 Editor
##  Mac OS X:11     Min.   : 0.00   Excel          : 1
##  Windows :20     1st Qu.: 0.00   LaTeX          : 2
##                  Median : 4.00   Microsoft Word:24
##                  Mean   : 5.71   R Markdown     : 4
##                  3rd Qu.: 8.00
##                  Max.   :33.00
```

Data Frame Basics

We will give an introduction about `data frames`. Later we will talk more about `lists` and `data frames` as well. To see what an R object is made up of, you can use `attributes()`

```
attributes(survey)
```

```
## $names
## [1] "Program"         "PriorExp"        "Rexperience"     "OperatingSystem"
## [5] "TVhours"         "Editor"
##
## $class
## [1] "data.frame"
##
## $row.names
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
## [26] 26 27 28 29 30 31
```

An R **data frame** is a *list* whose columns you can refer to by *name* or *index*

Data Frame Dimensions

We can use `nrow()` and `ncol()` to determine the number of survey responses and the number of survey questions

```
nrow(survey) # Number of rows (responses)
```

```
## [1] 31
```

```
ncol(survey) # Number of columns (questions)
```

```
## [1] 6
```

When writing reports, you will often want to say how large your sample size was. To do this *inline*, use the syntax:

```
`r nrow(survey)`
```

This allows us to write

`"31 students responded to the survey",`

and have the number 31 displayed that automatically changes when `nrow(survey)` changes.

Example: (Inline Code Chunks) Here's a more complex example of inline code use.

```
We collected data on `r ncol(survey)` survey questions from `r nrow(survey)` respondents.  Respondents r
```

Which results in

```
We collected data on 6 survey questions from 31 respondents.  Respondents represented 4 university progr
```

- **IMPORTANT**: You are expected to use inline code chunks instead of copying and pasting output whenever possible.

Indexing Data Frames

There are many different ways of indexing the same piece of a data frame. Each vector below contains 31 entries.

```
survey[["Program"]]  # "Program" element
```

```
##  [1] CS    CS    CS    Other CS    CS    CS    DS    CS    Other CS    CS
## [13] DS    Other CS    CS    CS    CS    CS    Other DS    DS    DS    DS
## [25] DS    DS    SW    SW    SW    DS    Other
## Levels: CS DS Other SW
```

```
head(survey[["Program"]],31)
```

```
##  [1] CS    CS    CS    Other CS    CS    CS    DS    CS    Other CS    CS
## [13] DS    Other CS    CS    CS    CS    CS    Other DS    DS    DS    DS
## [25] DS    DS    SW    SW    SW    DS    Other
## Levels: CS DS Other SW
```

```
survey$Program # "Program" element
```

```
##  [1] CS    CS    CS    Other CS    CS    CS    DS    CS    Other CS    CS
## [13] DS    Other CS    CS    CS    CS    CS    Other DS    DS    DS    DS
## [25] DS    DS    SW    SW    SW    DS    Other
## Levels: CS DS Other SW
```

```
head(survey$Program, 31) # "Program" element
```

```
##  [1] CS    CS    CS    Other CS    CS    CS    DS    CS    Other CS    CS
## [13] DS    Other CS    CS    CS    CS    CS    Other DS    DS    DS    DS
## [25] DS    DS    SW    SW    SW    DS    Other
## Levels: CS DS Other SW
```

```
survey[,1] # Data from 1st column
```

```
##  [1] CS    CS    CS    Other CS    CS    CS    DS    CS    Other CS    CS
## [13] DS    Other CS    CS    CS    CS    CS    Other DS    DS    DS    DS
## [25] DS    DS    SW    SW    SW    DS    Other
## Levels: CS DS Other SW
```

```
head(survey[,1], 11) # "Program" element
```

```
##  [1] CS    CS    CS    Other CS    CS    CS    DS    CS    Other CS
## Levels: CS DS Other SW
```

More Indexing

Note that single brackets and double brackets have different effects

```
survey[["Program"]]  # Returns the Program column as a vecto  r
```

```
##  [1] CS    CS    CS    Other CS    CS    CS    DS    CS    Other CS    CS
## [13] DS    Other CS    CS    CS    CS    CS    Other DS    DS    DS    DS
## [25] DS    DS    SW    SW    SW    DS    Other
## Levels: CS DS Other SW
```

```
head(survey[["Program"]], 22)
```

```
##  [1] CS    CS    CS    Other CS    CS    CS    DS    CS    Other CS    CS
## [13] DS    Other CS    CS    CS    CS    CS    Other DS    DS
## Levels: CS DS Other SW
```

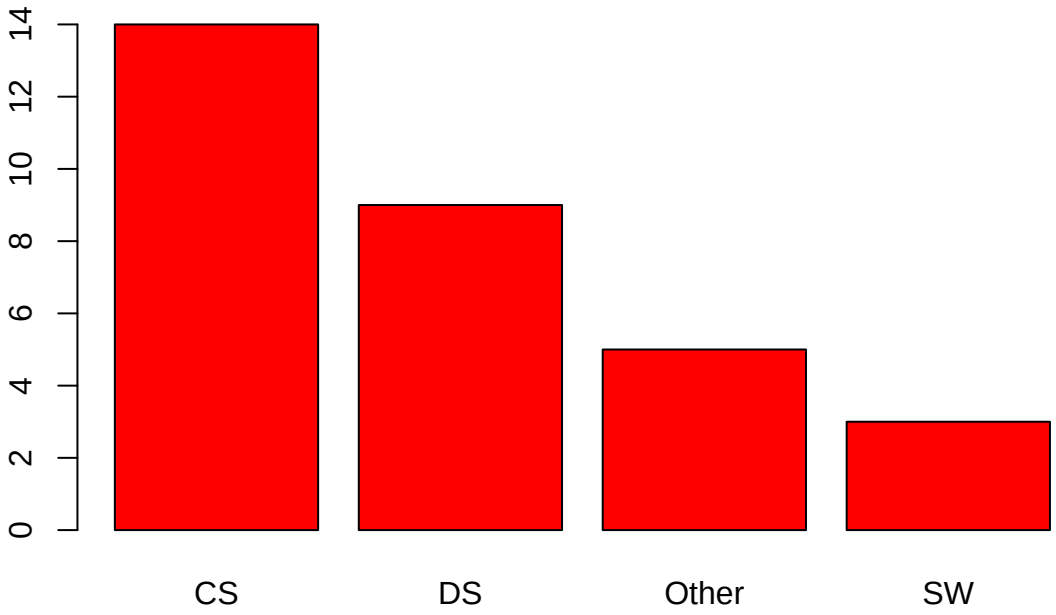
```
survey["Program"] # sub-data-frame containing only "Program"
```

```
##    Program
## 1       CS
## 2       CS
## 3       CS
## 4    Other
## 5       CS
## 6       CS
## 7       CS
## 8       DS
## 9       CS
## 10   Other
## 11      CS
## 12      CS
## 13      DS
## 14   Other
## 15      CS
## 16      CS
## 17      CS
## 18      CS
## 19      CS
## 20   Other
## 21      DS
## 22      DS
## 23      DS
## 24      DS
## 25      DS
## 26      DS
## 27      SW
## 28      SW
## 29      SW
## 30      DS
## 31   Other
```

```
options(repr.plot.width=4, repr.plot.height=5)
```

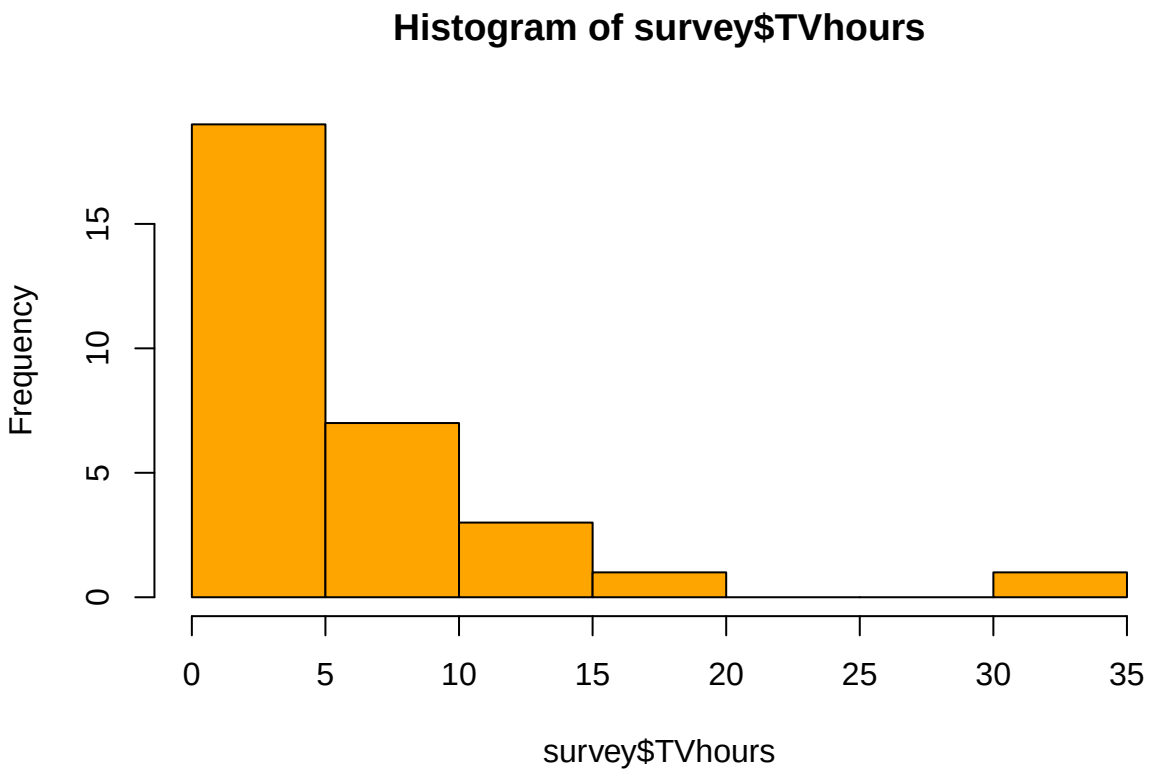
Bar Plot (Categorical Data)

```
plot(survey[["Program"]],col = "red", cex.lab = 1, cex.axis = 1, cex.main = 1, cex.names = 1)
```



Histogram (Continuous Data)

```
hist(survey$TVhours, col="orange")
```



Multiple Columns

```
head(survey[, c(1,5)]) # Data from 1st and 5th columns
```

```
##   Program TVhours
## 1      CS       2
## 2      CS      15
## 3      CS      16
## 4   Other       0
## 5      CS       2
## 6      CS       5
```

```
head(survey[c("Program", "Editor")]) # Data from "Program" and "Editor"
```

```
##   Program         Editor
## 1      CS Microsoft Word
## 2      CS Microsoft Word
## 3      CS Microsoft Word
## 4   Other          LaTeX
## 5      CS Microsoft Word
## 6      CS Microsoft Word
```

Indexing Rows and Columns

Data frames have two dimensions to index across

```
survey[6,] # 6th row
```

```
##   Program                PriorExp Rexperience OperatingSystem TVhours
## 6      CS Never programmed before  Never used         Windows       5
##           Editor
## 6 Microsoft Word
```

```
survey[6,5] # row 6, column 5
```

```
## [1] 5
```

```
survey[6, "Program"] # Program of 6th survey respondent
```

```
## [1] CS
## Levels: CS DS Other SW
```

```
survey[["Program"]][6]  # Program of 6th survey respondent
```

```
## [1] CS
## Levels: CS DS Other SW
```

More indexing

In Lab 1, we introduced the colon operator :. We can use this operator for indexing

```
survey[1:3,]  # equivalent to head(survey, 3)
```

```
##   Program                PriorExp      Rexperience OperatingSystem TVhours
## 1      CS         Some experience       Never used        Mac OS X       2
## 2      CS Never programmed before       Never used         Windows      15
## 3      CS         Some experience Basic competence        Mac OS X      16
##           Editor
## 1 Microsoft Word
## 2 Microsoft Word
## 3 Microsoft Word
```

```
survey[3:5, c(1,5)]
```

```
##   Program TVhours
## 3      CS      16
## 4   Other       0
## 5      CS       2
```

Subsets of Data

We are often interested in learning something about a specific subset of data

```
survey[survey$Program=="DS", ] # Data from the DS students
```

```
##    Program             PriorExp          Rexperience OperatingSystem TVhours
## 8       DS      Some experience     Basic competence         Windows       4
## 13      DS Extensive experience     Basic competence         Windows      10
## 21      DS      Some experience     Basic competence         Windows       8
## 22      DS Extensive experience Installed on machine         Windows       3
## 23      DS      Some experience     Basic competence         Windows       0
## 24      DS      Some experience     Basic competence         Windows       0
## 25      DS      Some experience     Basic competence         Windows       0
## 26      DS Extensive experience             R Wizard        Mac OS X       0
```

```
## 30      DS      Some experience           Never used         Windows       0
##            Editor
## 8  Microsoft Word
## 13 Microsoft Word
## 21 Microsoft Word
## 22 Microsoft Word
## 23 Microsoft Word
## 24          Excel
## 25     R Markdown
## 26          LaTeX
## 30 Microsoft Word
```

```
survey[which(survey$Program=="DS"), ] # Does the same thing
```

```
##    Program             PriorExp          Rexperience OperatingSystem TVhours
## 8       DS      Some experience     Basic competence         Windows       4
## 13      DS Extensive experience     Basic competence         Windows      10
## 21      DS      Some experience     Basic competence         Windows       8
## 22      DS Extensive experience Installed on machine         Windows       3
## 23      DS      Some experience     Basic competence         Windows       0
## 24      DS      Some experience     Basic competence         Windows       0
## 25      DS      Some experience     Basic competence         Windows       0
## 26      DS Extensive experience              R Wizard        Mac OS X       0
## 30      DS      Some experience           Never used         Windows       0
##            Editor
## 8  Microsoft Word
## 13 Microsoft Word
## 21 Microsoft Word
## 22 Microsoft Word
## 23 Microsoft Word
## 24          Excel
## 25     R Markdown
## 26          LaTeX
## 30 Microsoft Word
```

Examples

Let's pull all of `Data Science (DS)` students who have never used R before

```
survey[survey$Program=="DS" & survey$Rexperience=="Never used", ]
```

```
##    Program        PriorExp Rexperience OperatingSystem TVhours         Editor
## 30      DS Some experience  Never used         Windows       0 Microsoft Word
```

Cleaner Subsetting

When subset conditions get long or messy, it is preferable to use **`subset()`** function

Example:

Selecting `OperatingSystem` and `TVhours` responses from all students who are either in `DS` or `Other` and who listed their R experience as `Basic competence`.

```
subset(survey, select=c("OperatingSystem", "TVhours"),  subset=(Program == "PPM" | Program == "Other") &
```

```
##   OperatingSystem TVhours
## 4        Mac OS X       0
```

Splitting Long Function Calls

As your function calls get longer and more complicated, you may find it useful to split them over multiple lines. Here's one way to rewrite the previous line

```
subset(survey,
       select = c("OperatingSystem", "TVhours"),
       subset = (Program == "PPM" | Program == "Other") &
                (Rexperience == "Basic competence")
       )
```

Some Simple Calculations

```
mean(survey$TVhours[survey$Program == "DS"]) # Average time DS's spent watching TV
```

```
## [1] 2.777778
```

```
mean(survey$TVhours[survey$Program == "SW"]) # Average time SW's spent watching TV
```

```
## [1] 2.333333
```

```
mean(survey$TVhours[survey$Program == "Other"]) # Average time "Others" spent watching TV
```

```
## [1] 5.4
```

Later we'll learn a better way of doing these calculations by using **aggregate()** function.

Columns as Variables

If we want to focus on a particular column of data frame, we can define it as a new variable.

```
tv.hours <- survey$TVhours  # Vector of TVhours watched
mean(tv.hours)              # Average time spent watching TV
```

```
## [1] 5.709677
```

```
sd(tv.hours)                # Standard deviation of TV watching time
```

```
## [1] 7.156319
```

Respondents spending more than 5 hours on TV:

```
tv.hours >= 5
```

```
##  [1] FALSE  TRUE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE  TRUE
## [13]  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [25] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
```

```
head(tv.hours >= 5, 20) # (Settings adjusted to print first 20 elements)
```

```
##  [1] FALSE  TRUE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE  TRUE
## [13]  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE
```

Respondents spending more than 5 hours on TV:

```
sum(tv.hours >= 5)   # How many people watched 5 or more hours of TV?
```

```
## [1] 13
```

R Coding Style

Coding style (and code commenting) becomes increasingly important with more advanced and involved programming tasks. A few R "style guides" exist:

- Google's
- Hadley Wickham's

Borrowing Hadley Wickham's words:

> You don't have to use my style, but you really should use a **consistent** style.

R Style Recommendations

- Hadley Wickham's guide is short and easy to follow.
- We'll revisit the question of *coding style* several times over and again.

Enforced Style: Assignment Operator

Assignment operator. Use `<-`

> **Good** `student.names <- c("Wajid", "Ahsan", "Fatima")` **Bad** `student.names = c("Wajid", "Ahsan", "Fatima")`

Note:

When specifying function arguments, only `=` is valid

> **Good** `sort(tv.hours, decreasing = TRUE)` **Bad** `sort(tv.hours, decreasing <- TRUE)`

Enforced Style: Spacing

- Binary operators should have spaces around them > **Good** `5 * 45` **Bad** `5*45`
- Commas should have a space after, but not before (just like in writing)

  Good `which(student.names == "Fatima")` **Bad** `which(student.names=="Fatima")`
- For specifying arguments, spacing around `=` is optional

  Accepted sort(tv.hours, decreasing=TRUE) sort(tv.hours, decreasing = FALSE)

Enforced Style: Variable Names

To make code easy to read, debug, and maintain, you should use **concise** but **descriptive** variable names

- Terms in variable names should be separated by `_` or `.`

  Good `day_one` `day.one` `day_1` `day.1` `day1` **bad** `d1` `DayOne` `dayone`
  Can be made more concise: `first.day.of.the.month`
- Avoid using variable names that are already pre-defined in R

  EXTREMELY bad `c` `T` `pi` `sum` `mean`

Homework and Lab

- **Homework 1** will be posted next week
  - You will have to submit .Rmd solution file through email
- **Lab 1** is now available
  - Submit Lab 1 through email by **11:59pm 8 Feb**