Tutorial 2 - Importing data and more basics

Contents

How to do this Tutorial

Try re-running as much of the code as you have time for, taking time to understand what is happening in each line of code. You can enter the code directly into the Console. (Optional: you may enter your code in the code chunk below.)

Tip: (Tip:**) Instead of copying and pasting code, practice typing it yourself. This will help you to learn the syntax.

```
# Edit me (optional)
```

Importing data

- To import tabular data into R, we use the **`read.table()`** command

```
survey <- read.table("http://www.andrew.cmu.edu/user/achoulde/94842/data/survey_data.csv", header=TRUE, sep=",")
```

- Let's parse this command one component at a time
- The data is in a file called `survey_data.csv`, which is an online file
- The file contains a `header` as its first row
- The csv format means that the data is comma-separated, so `sep=","`
- Could've also used `read.csv()`, which is just `read.table()` with the preset `sep=","`

Exploring the data

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

```
class(survey)
```

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

- To view the first few rows of the data, use **`head()`**

```
head(survey, 3)
```

```
##   Program             PriorExp      Rexperience OperatingSystem TVhours
## 1    MISM Extensive experience Basic competence         Windows       0
## 2     PPM      Some experience       Never used         Windows       3
## 3     PPM      Some experience Basic competence        Mac OS X      30
##           Editor
## 1 Microsoft Word
## 2 Microsoft Word
## 3 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':    40 obs. of  6 variables:
##  $ Program        : Factor w/ 3 levels "MISM","Other",..: 1 3 3 2 1 3 2 2 3 1
...
##  $ PriorExp       : Factor w/ 3 levels "Extensive experience",..: 1 3 3 3 3 2 3
3 1 1 ...
##  $ Rexperience    : Factor w/ 3 levels "Basic competence",..: 1 3 1 1 1 3 2 1 2
3 ...
##  $ OperatingSystem: Factor w/ 2 levels "Mac OS X","Windows": 2 2 1 1 1 2 2 1 2 2
...
##  $ TVhours        : num  0 3 30 6 20 15 6 8 10 0 ...
##  $ Editor         : Factor w/ 3 levels "LaTeX","Microsoft Excel",..: 3 3 3 2 1 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
##  MISM :22   Extensive experience   : 7   Basic competence    :11
##  Other: 9   Never programmed before: 6   Installed on machine:11
##  PPM  : 9   Some experience        :27   Never used          :18
##
##
##
##  OperatingSystem    TVhours                    Editor
##  Mac OS X:18     Min.   : 0.000   LaTeX           : 6
##  Windows :22     1st Qu.: 1.500   Microsoft Excel: 1
##                  Median : 6.000   Microsoft Word :33
##                  Mean   : 9.062
##                  3rd Qu.:10.000
##                  Max.   :40.000
```

Data frame basics

- We will talk more about lists and data frames next week, so only give an intro here
- 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] 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
```

> 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] 40
```

```
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 "40 students responded to the survey", and have the number displayed automatically change when `nrow(survey)` changes.

Inline code chunks example

- Here's a more complex example of inline code use.

```
We collected data on `r ncol(survey)` survey questions from `r nrow(survey)` respon
dents.  Respondents represented `r length(unique(survey[["Program"]]))` CMU program
s.  `r sum(survey[["Program"]] == "PPM")` of the repondents were from PPM.
```

- Which results in

> We collected data on 6 survey questions from 40 respondents. Respondents represented 3 CMU programs. 9 of the repondents were from PPM.

- **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 40 entries. For display purposes, the settings have been adjusted so that only the first 22 are shown below

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

```
##  [1] MISM  PPM   PPM   Other MISM  PPM   Other Other PPM   MISM  MISM
## [12] MISM  PPM   MISM  MISM  MISM  MISM  PPM   MISM  MISM  Other Other
## Levels: MISM Other PPM
```

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

```
##  [1] MISM  PPM   PPM   Other MISM  PPM   Other Other PPM   MISM  MISM
## [12] MISM  PPM   MISM  MISM  MISM  MISM  PPM   MISM  MISM  Other Other
## Levels: MISM Other PPM
```

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

```
##  [1] MISM  PPM   PPM   Other MISM  PPM   Other Other PPM   MISM  MISM
## [12] MISM  PPM   MISM  MISM  MISM  MISM  PPM   MISM  MISM  Other Other
## Levels: MISM Other PPM
```

More indexing

- Note that single brackets and double brackets have different effects

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

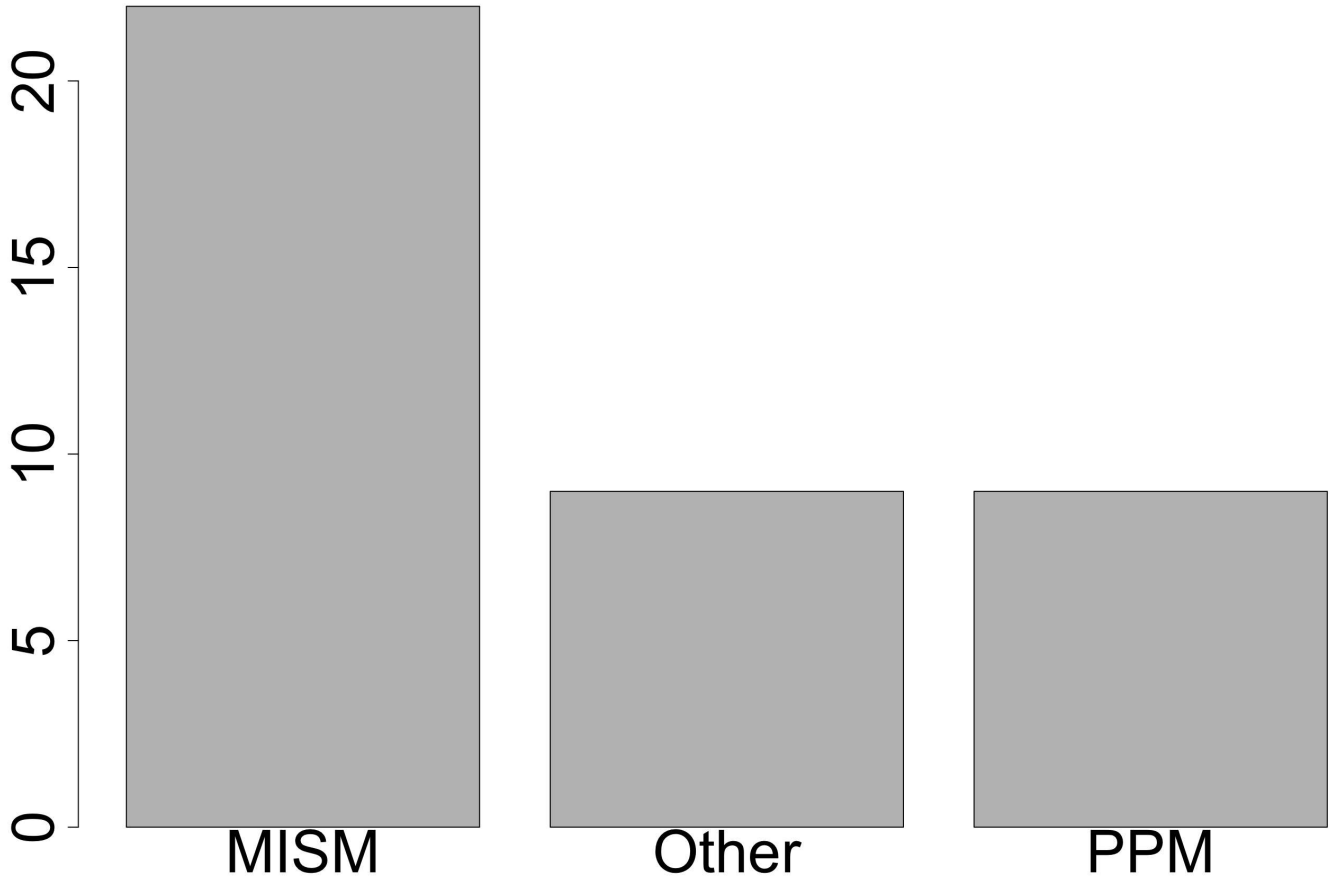
```
##  [1] MISM  PPM   PPM   Other MISM  PPM   Other Other PPM   MISM  MISM
## [12] MISM  PPM   MISM  MISM  MISM  MISM  PPM   MISM  MISM  Other Other
## Levels: MISM Other PPM
```

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

```
##    Program
## 1     MISM
## 2      PPM
## 3      PPM
## 4    Other
## 5     MISM
## 6      PPM
## 7    Other
## 8    Other
## 9      PPM
## 10    MISM
## 11    MISM
## 12    MISM
## 13     PPM
## 14    MISM
## 15    MISM
## 16    MISM
## 17    MISM
## 18     PPM
## 19    MISM
## 20    MISM
## 21   Other
## 22   Other
## 23   Other
## 24   Other
## 25     PPM
## 26    MISM
## 27    MISM
## 28    MISM
## 29    MISM
## 30    MISM
## 31    MISM
## 32    MISM
## 33    MISM
## 34     PPM
## 35    MISM
## 36     PPM
## 37    MISM
## 38   Other
## 39   Other
## 40    MISM
```

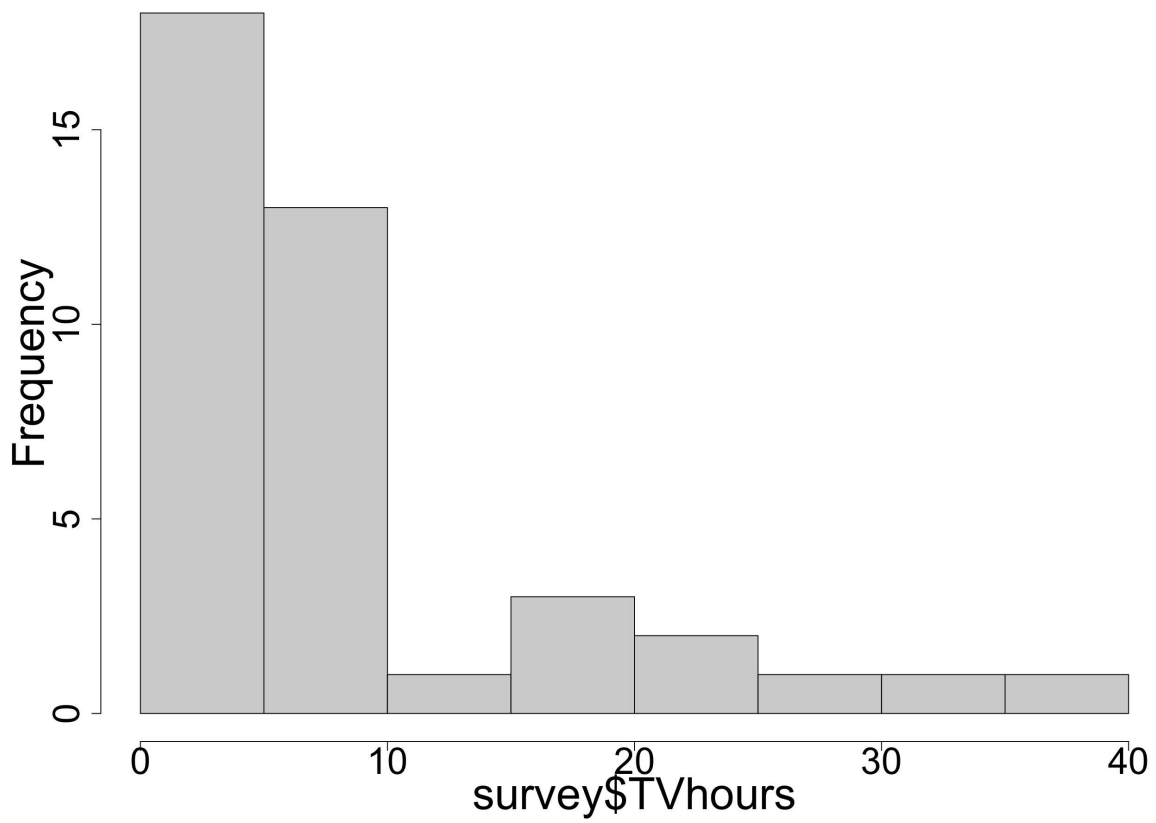
Bar plot (categorical data)

```
plot(survey[["Program"]])
```



Histogram (continuous data)

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



Indexing multiple columns

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

```
##   Program TVhours
## 1    MISM       0
## 2     PPM       3
## 3     PPM      30
## 4   Other       6
## 5    MISM      20
## 6     PPM      15
```

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

```
##   Program          Editor
## 1    MISM  Microsoft Word
## 2     PPM  Microsoft Word
## 3     PPM  Microsoft Word
## 4   Other Microsoft Excel
## 5    MISM           LaTeX
## 6     PPM  Microsoft Word
```

Indexing rows and columns

- Data frames have two dimensions to index across

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

```
##   Program                PriorExp Rexperience OperatingSystem TVhours
## 6     PPM Never programmed before  Never used         Windows      15
##           Editor
## 6 Microsoft Word
```

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

```
## [1] 15
```

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

```
## [1] PPM
## Levels: MISM Other PPM
```

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

```
## [1] PPM
## Levels: MISM Other PPM
```

More indexing

- In Lab 1, you were introduced to the colon operator `:`
- We can use this operator for indexing

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

```
##   Program             PriorExp      Rexperience OperatingSystem TVhours
## 1    MISM Extensive experience Basic competence         Windows       0
## 2     PPM      Some experience       Never used         Windows       3
## 3     PPM      Some experience Basic competence       Mac OS X      30
##           Editor
## 1 Microsoft Word
## 2 Microsoft Word
## 3 Microsoft Word
```

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

```
##   Program TVhours
## 3     PPM      30
## 4   Other       6
## 5    MISM      20
```

Subsets of data

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

```
survey[survey$Program=="MISM", ] # Data from the MISM students
survey[which(survey$Program=="MISM"), ] # Does the same thing
```

```
##    Program             PriorExp          Rexperience OperatingSystem
## 1     MISM Extensive experience     Basic competence         Windows
## 5     MISM      Some experience     Basic competence       Mac OS X
## 10    MISM Extensive experience           Never used         Windows
## 11    MISM      Some experience           Never used         Windows
## 12    MISM Extensive experience Installed on machine       Mac OS X
## 14    MISM      Some experience Installed on machine         Windows
## 15    MISM      Some experience           Never used         Windows
## 16    MISM      Some experience           Never used       Mac OS X
## 17    MISM      Some experience Installed on machine         Windows
## 19    MISM      Some experience           Never used         Windows
## 20    MISM      Some experience Installed on machine         Windows
## 26    MISM      Some experience           Never used         Windows
## 27    MISM      Some experience     Basic competence       Mac OS X
## 28    MISM      Some experience     Basic competence       Mac OS X
## 29    MISM      Some experience Installed on machine         Windows
## 30    MISM      Some experience           Never used         Windows
## 31    MISM Extensive experience     Basic competence       Mac OS X
## 32    MISM      Some experience           Never used       Mac OS X
## 33    MISM      Some experience Installed on machine       Mac OS X
## 35    MISM      Some experience     Basic competence       Mac OS X
## 37    MISM Extensive experience     Basic competence         Windows
## 40    MISM Extensive experience Installed on machine       Mac OS X
##    TVhours         Editor
## 1        0 Microsoft Word
## 5       20          LaTeX
## 10       0 Microsoft Word
## 11       4 Microsoft Word
## 12       0 Microsoft Word
## 14       2 Microsoft Word
## 15      24 Microsoft Word
## 16      10 Microsoft Word
## 17      25 Microsoft Word
## 19       0 Microsoft Word
## 20      10 Microsoft Word
## 26      20 Microsoft Word
## 27       0          LaTeX
## 28       0          LaTeX
## 29      10 Microsoft Word
## 30       5 Microsoft Word
## 31      40 Microsoft Word
## 32      10 Microsoft Word
## 33       0 Microsoft Word
## 35       0          LaTeX
## 37       2          LaTeX
## 40       7 Microsoft Word
```

More subset examples

- Let's pull all of the PPM students who have never used R before

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

```
##    Program                PriorExp Rexperience OperatingSystem TVhours
## 2      PPM         Some experience  Never used         Windows     3.0
## 6      PPM Never programmed before  Never used         Windows    15.0
## 13     PPM         Some experience  Never used         Windows     0.0
## 18     PPM         Some experience  Never used         Windows     0.0
## 25     PPM         Some experience  Never used       Mac OS X      5.5
##            Editor
## 2  Microsoft Word
## 6  Microsoft Word
## 13 Microsoft Word
## 18 Microsoft Word
## 25 Microsoft Word
```

Cleaner subsetting

- When the subset conditions get long or messy, it is preferable to use the **`subset()`** function
- Here's an example of selecting the OperatingSystem and TVhours responses from all of the students who are either in PPM or Other and who listed their R experience as "Basic competence".

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

```
##    OperatingSystem TVhours
## 3          Mac OS X      30
## 4          Mac OS X       6
## 8          Mac OS X       8
## 39          Windows      10
```

Splitting a long function call

- 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")
       )
```

```
##    OperatingSystem TVhours
## 3          Mac OS X      30
## 4          Mac OS X       6
## 8          Mac OS X       8
## 39          Windows      10
```

Some simple calculations

```
mean(survey$TVhours[survey$Program == "PPM"]) # Average time PPM's spent watching T
V
```

```
## [1] 9.944444
```

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

```
## [1] 8.590909
```

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

```
## [1] 9.333333
```

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

Defining variables

- If we wanted to focus on a particular column of the data frame, we could always define it as a new variable

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

```
## [1] 9.0625
```

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

```
## [1] 10.26332
```

```
tv.hours >= 5         # (Settings adjusted to print first 40 elements)
```

```
##  [1] FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE
## [12] FALSE FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE
## [23] FALSE FALSE  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE FALSE
## [34]  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE
```

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

```
## [1] 24
```

R coding style

- Coding style (and code commenting) will become increasingly more important as we get into more advanced and involved programming tasks
- A few R "style guides" exist:
  - Google's (https://google.github.io/styleguide/Rguide.xml)
  - Hadley Wickham's (http://r-pkgs.had.co.nz/style.html)
- 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 (http://r-pkgs.had.co.nz/style.html) guide is short and easy to follow
- We'll revisit the question of coding style several times over the course of the class

Enforced style: Assignment operator

Assignment operator. USE `<-`

```
student.names <- c("Eric", "Hao", "Jennifer")  # Good
student.names = c("Eric", "Hao", "Jennifer") # Bad
```

- Note: When specifying function arguments, only `=` is valid

```
sort(tv.hours, decreasing=TRUE) # Good
sort(tv.hours, decreasing<-TRUE) # Bad!!
```

Enforced style: Spacing

- Binary operators should have spaces around them
- Commas should have a space after, but not before (just like in writing)

```
3 * 4 # Good
3*4 # Bad
which(student.names == "Eric") # Good
which(student.names=="Eric") # Bad
```

- For specifying arguments, spacing around `=` is optional

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

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 `.`

```
# Accepted
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
```

- Go ahead and complete **Lab 2**