Introductory Statistics Lectures

Introduction to R

Anthony Tanbakuchi
Department of Mathematics
Pima Community College

(Compile date: Tue May 19 14:47:55 2009)

Contents

1 Introduction to R

Why must we use computers in stats?

Who wants to find the mean of these numbers (100) by hand?

```
R: x
  [1] 8 2 4 4 3 3 1 4 6 5 1 2 5 4 8 5 1 6 4 0 8 3 8 9 9 1 8
 [28] 6 3 1 5 7 1 8 3 9 2 0 3 3 2 9 4 7 3 2 2 2 5 3 1 4 2 6
 [55] 2 6 0 5 8 1 5 3 7 8 2 9 6 6 5 1 8 8 0 3 6 6 1 9 1 3 5
 [82] 3 8 7 7 5 7 6 6 7 5 5 2 0 5 8 5 1 8 5
```

Computers are required when dealing with real data sets because they are large and they are becoming dynamic.

```
R: mean(x)
[1] 4.49
```

Figure 1: The Univac II, circa 1958.

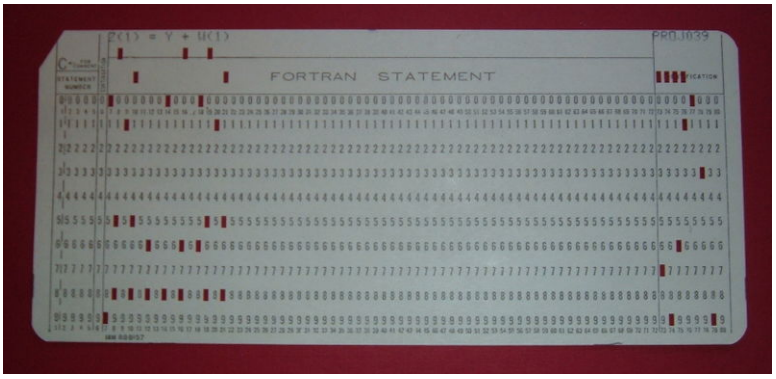


Figure 2: The program $Z(1) = Y + W(1)$ on a punch card.

Statistics and Computers

The U.S. Bureau of the Census used the UNIVAC computer to process the 1950 Census data. In 1951, a report stated "it would take at least 650 keypunch operators, working on 17 document punch machines during the week of peak Census processing, to produce a million punched cards completely edited and ready for tabulation." To realize productivity gains, and to pursue their interest in technical innovation, the Bureau acquired an electronic computing machine.

1.1 Introducing R

It's statistics program / language capable of most basic and advanced statistical procedures.

Some benefits:

- It's free (GNU General public license).
- Works on all major platforms.
- Compatible with S.

- It's a powerful calculator.
- It's simple (therefore easy).
- It encourages reproducible research.

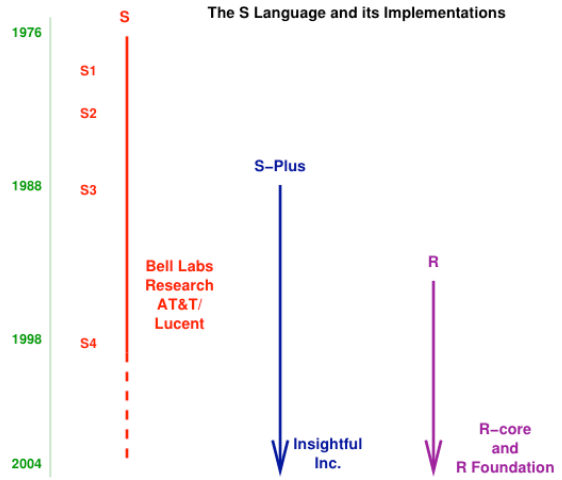


Figure 3: Timeline of R. (Credit: From John M. Chambers 2006 talk.)

Why not use another statistics package?

- We could, but most complete packages cost from hundreds to thousands of dollars! Since R can do basic through highly advanced statistics — and it's free — it is a good choice!
- Why not use Excel? Excel, while an spreadsheet excellent package, is not a statistical software package. Even with its statistical (Analysis Tool Pack) add-in it will not be able to adequately perform all the necessary functions.
- Why not use a TI statistics calculator? We could use it for trivial problems. But you'd not likely use it after the class for real data. It's just not reasonable to enter large data sets (with potentially thousands of numbers) into a calculator and you can't readily put your work or graphs into a report.
- By teaching you R you will learn a real world statistics program that you can actually use in your work if needed. Since it's free and accepted by the scientific community, you won't have to ask your company to buy an expensive piece of software. Perhaps you can convince your boss to give you a raise since you've saved them thousands of dollars by using R!

Four key things you must learn from this lecture

By the end of this lecture, you must be able to:

1. Store a set of data (vector) in a variable.
2. Know how to properly use functions and their arguments.
3. Load external data from .RData files (book data).

4. Know how to access columns of data in tables.

How to install R

Visit the course website for installation instructions!

1.2 R fundamentals

USING R AS A CALCULATOR

- Some simple calculations:

```
R: 6 + 5
[1] 11
R: 8^2
[1] 64
R: 3 * (8 - 3)
[1] 15
R: sqrt(4)
[1] 2
```

- R will alert you if something is unclear or wrong:

```
R: 3(8-3)
Error: attempt to apply non-function
```

R calculator use

1. Uses standard operators:
   `+, -, *, /, ^`
2. Observes normal order of operations.
3. Make sure to include all "*". Do 3*(4+2), NOT 3(4+2).
4. Use `sqrt(x)` to find $\sqrt{x}$.
5. Use parenthesis `(...)` as needed.
6. Use **up arrow key** $\uparrow$ to recall previous entries.

Try it out:

Use R to do the following:

1. 4.2/61 (Check: 0.06885246)
2. 8^{12} (Check: 68719476736)
3. $\frac{88-2.4}{12}$ (Check: 7.133333)
4. $\sqrt{52}$ (Check: 7.211103)

VARIABLES AND ENTERING DATA

R Command

STORING DATA IN A VARIABLE:
`variableName=data`

Variable name on the left of the equal sign, data on the right. (Hit enter to execute.) If no errors occur, no output will be given.

R Command
VIEWING DATA IN A VARIABLE: `variableName` Just type the variable name and hit enter. R will print out the value of the variable.

Variables store values

- Use variables to store values:

```
R: x = 0.95
```

- For remembering values:

```
R: x
[1] 0.95
```

- For calculations:

```
R: 100 * x
[1] 95
```

Types of variables

A variable stores data. It can be (to name a few):

scalar a single value

```
R: x
[1] 10.2
```

vector data set with multiple values

```
R: y
[1]  3 -8  5  7  2
```

table like (data frame) each column is a vector

```
R: z
     name age gender
1    John  27   MALE
2 Jessica  33 FEMALE
3   James  19   MALE
```

Try it out: storing scalars

- Store 2.5 in the variable `w`
- Store -8.4 in the variable `y`
- Multiply `w` and `y` (Check: -21)
- Store `w*y` in `f`

Variable names

- Use descriptive names that make sense.
- Must start with a letter.
- They are **cAsE** sensitive (X and x are different).
- No spaces, dashes, or symbols other than periods (.) or underscores (_).

Example 1 (Good variable names). x, mu, x.bar, studentHeight, student.height, student_height, n, N

Example 2 (Bad variable names). 2x, student height, student-height

R Command

STORE VECTOR (DATA SET):
`c(x1, x2, ...)`

The `c()` function to concatenates a **comma** separated list of values into a variable.

Example

```
R: x = c(-10, -8, -2, 0, 1, 2, 3, 6, 12)

R: x
 [1] -10  -8  -2   0   1   2   3   6  12

R: q = x^3

R: q
 [1] -1000  -512    -8     0     1     8    27   216  1728
```

Try it out: storing vectors

The ages (in years) for a group of people are $\{52, 43, 22, 68\}$

- Store the vector of ages in the variable `ages.years`
- make a new variable called `ages.days` that has the ages of the people in days.

R Command

ACCESSING VECTOR ELEMENTS:
`vectorName[filter]`

If `variableName` is a vector, then use the [] notation with a filter expression inside. Think of the expression between [] as your google search of the data in the variable.

Example

```
R: y
 [1]  3 -8  5  7  2
```

Getting the third element

```
R: y[3]
 [1] 5
```

Determining which elements are greater than 2

```
R: y > 2
 [1]  TRUE FALSE  TRUE  TRUE FALSE
```

Getting all elements in y where y is greater than 2

```
R: y[y > 2]
 [1] 3 5 7
```

Note above, our filter is y>2.

Try it out: storing vectors

you previously defined the ages in years as $\{52, 43, 22, 68\}$

- retrieve the second value in `ages.years`.
- retrieve all the ages in `ages.years` greater than 30.

Recall,

```
R: z
      name age gender
1     John  27   MALE
2  Jessica  33 FEMALE
3    James  19   MALE
```

SHOWING THE FIRST FEW ROWS OF A TABLE:
`head(tableName)`

Prints out the first couple of rows at the head of the table. Useful if the table has thousands of rows and you just want a glimpse of what is inside.

R COMMAND

LISTING COLUMN NAMES:
`names(tableName)`

Prints the names of all the columns for a table.

R COMMAND

```
R: names(z)
[1] "name"   "age"    "gender"
```

GETTING A COLUMN OF DATA:
`tableName$columnName`

Where `tableName` is the name of the table and `columnName` is a column name in the table.

R COMMAND

```
R: z$age
[1] 27 33 19
```

If you just want to get the male ages:

```
R: z$age[z$gender == "MALE"]
[1] 27 19
```

Note the **double equal sign** to indicate **equality** as opposed to a single equal sign for storing a value in a variable.

Try it out: tables

We will look at the built in table `women`.

- Type `women` and hit enter to see the build in data.
- What are the column names?
- Type `names(women)`
- Retrieve the vector of heights from the table and store it in `h`.

LISTING DEFINED VARIABLES:
`ls()`

Lists all variables you have defined or loaded.

R COMMAND

Example:

```
R: ls()
[1] "ages.days"   "ages.years"  "f"           "h"
[5] "q"           "w"           "x"           "y"
[9] "z"
```

FUNCTIONS

Functions
Use functions to manipulate data, make plots, and run calculations.

```
R: x = c(10, 24, 15)
R: sum(x)
[1] 49
```

Function analogy: pets
Think of functions like pets:

- They don't come unless we call them by name (spelled and capitalized properly).
- They have a mouth (parenthesis) where we feed them arguments (inputs).
- They complain if not fed properly.
- They reward you with droppings (outputs).

R Command

BASIC FORM OF FUNCTIONS:
`output = functionName(input)`

Example:

```
R: max(x)
[1] 24
```

Example of the log function: `log(x, base)`

```
R: log(256, 2)
[1] 8
```

R Command

GENERAL FORM OF FUNCTIONS:
`output = functionName(arg1, arg2, ..., opt1=default1, opt2=default2,...)`

All arguments are comma seperated (spaces don't matter).
required arguments (arg1, arg2, ...) must all be given **in order**.
optional arguments have the form `name=default` . You specify them after the required arguments (order unimportant) only if you want to change their default value. Must use the format `name=value` .

Example 3 (mean function: trim option). The mean function has the optional argument `trim=0` which by default trims 0 of the data. If you want to trim some of the data then set trim to a different value.

```
mean(x, trim = 0)
```

Let's first find the mean with the default trim value:

```
R: w = c(10, 11, 9, 8, 785, 12, 10, 12, 10, 10, 9,
+     13, 8)
R: mean(w)
[1] 69.769
```

Now let's change the default value for trim:

```
R: mean(w, trim = 0.1)
[1] 10.364
```

Try it out: functions

Using the vector of women heights `h` you defined:

- Find the mean of `h` using `mean(h)`.
- Now find the 10% trimmed mean using `mean(h, trim=0.1)`

GRAPHICS

R has many plotting functions. They generally all work in the same way.

Graphing functions:	R Command
`hist(x)` **x** vector of values to make a histogram	

Example:

```
R: hist(h)
```

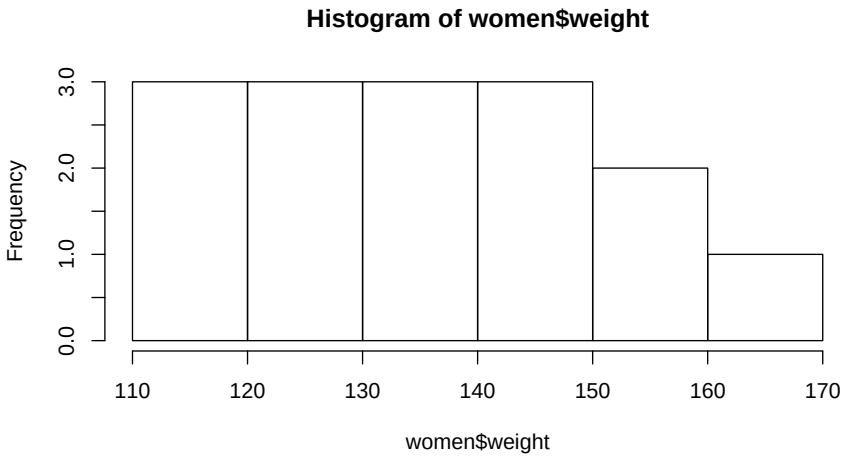
Optional arguments for most plotting functions

main=" " set the plot title
xlab=" " label the x axis
ylab=" " label the y axis

Making a simple plot

Histogram of women weights

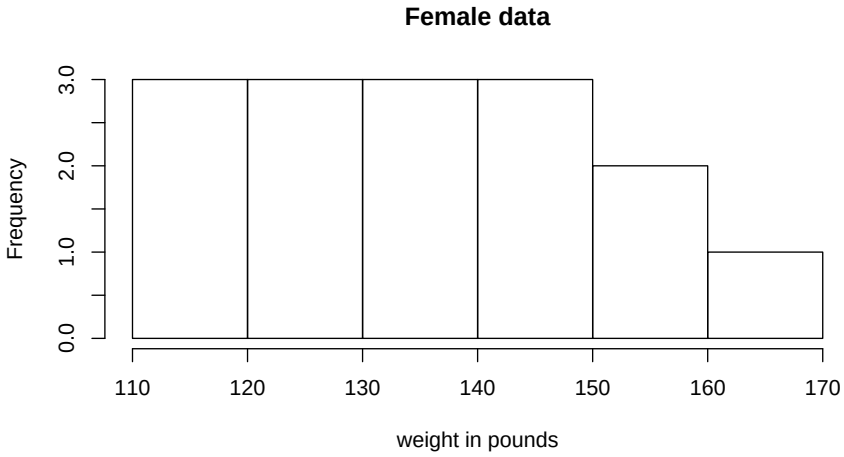
```
R: hist(women$weight)
```



Making a better plot with labels

Set the main title and x axis label

```
R: hist(women$weight, main = "Female data", xlab = "weight in ↩
    pounds")
```



Try it out: functions

Using the vector of women heights `h` you defined:

- Make a histogram of the women heights. Set the title to "Heights of women"

GETTING HELP

Help in R

1. Use the quick reference sheet.
2. To get help with a function put a ? in-front of its name: `?curve`
3. **Email me.** Copy and paste all the related R input and output and send it to me with a summary of what you are having a problem with.

Common errors

- Capitalization
- Spelling of function names
- Matching parenthesis and quotes

LOADING EXTERNAL DATA

Loading Book Data

How to load book data.

1. Go to class website and download the .RData file. (You may need to right click on the link and select save link as...)
2. To load the data in to R, either double click on the .RData file or:
   - Windows: File→Load Workspace...
   - Mac: Workspace→Load Workspace File...
3. Type `ls()` in R to see the loaded variables.

Try it out

- Load the book data from the website (download it first).
- Look at the data in the object browser.
- Get the bear's ages from the `Bears` table.
- Find the mean of the bear ages.

1.3 Summary

Summary

- Store data in variables (cAsE sensitive).
- Enter data sets with `c()` function. ie `x=c(1,2,3,4)`
- List all variables with `ls()`
- Access vector elements with `x[...]`, ie `x[1]`
- Access table columns with dollar sign notation: `tableName$columnName`
- Use R's help: `?functionName` & email me if stuck!
- Functions: `output = functionName(arg1, arg2, ..., opt1=default1, opt2=default2,...)`
  - Arguments inside parenthesis ( ), **order matters**.
  - Optional arguments: name=value. Only set if you want to change the default.
- Use graphics optional arguments: main=" ", xlab=" " , ylab=" ", xlim=c(xmin, xmax)

- Use **up arrow key** ↑ to recall/correct previous entries.

1.4 What you must know to survive

Key skills

Question 1. How would you store 3 in the variable `n`?

Question 2. How would you store $\{2, 5, 7\}$ in the variable `j`?

Question 3. How would you find the mean of `j`?

Question 4. How do you load the book data?

Question 5. How do you list all the variables/tables?

Question 6. How do you see what is inside a variable?

Question 7. How do you list the column names for a table called `survey.data` ?

Question 8. How would you get the vector of data from the column named "Weight" in the table `survey.data` ?

1.5 Example student session and submitting HW

EXAMPLE SESSION

A student needs to load the book data, then find the table containing the survey data on bears, get the column of weight data, and find the mean weight.

1. Load the book data (use menu).
2. View the variables to find table name:

```
R: ls()
 [1] "BODYTEMP"  "Bears"     "Bostrain"  "Cans"      "Chmovie"
 [6] "Cigaret"   "Coins"     "Cola"      "Cotinine"  "ELECTRIC"
[11] "Fhealth"   "Garbage"   "Homeruns"  "Homes"     "MM"
[16] "Mhealth"   "OldFAITH"  "POPLAR"    "WEATHER"
```

3. Table is named `Bears` , view the column names (can type `Bears` and hit enter to look at table):

```
R: names(Bears)
[1] "AGE"      "MONTH"    "SEX"      "HEADLEN"  "HEADWTH"
[6] "NECK"     "LENGTH"   "CHEST"    "WEIGHT"
```

4. Data is in the column named `WEIGHT` , thus to view column of data type:

```
R: Bears$WEIGHT
  [1]  80 344 416 348 166 220 262 360 204 144 332  34 140 180
 [15] 105 166 204  26 120 436 125 132  90  40 220  46 154 116
 [29] 182 150  65 356 316  94  86 150 270 202 202 365  79 148
 [43] 446  62 236 212  60  64 114  76  48  29 514 140
```

5. Now find the mean of the data:

```
R: mean(Bears$WEIGHT)
[1] 182.89
```

Observe that the **table** of data is in the variable `Bears` and the **column** of data is accessed as `Bears$WEIGHT` . You must give the mean function the **column** (vector) of data, not the table.

SUBMITTING HW

How to submit HW when using R

If you use R for a HW problem I want to see your work, output, and any plots. Only include your correct output, I don't want to see initial attempts containing output with errors.

Create a word document that has each problem clearly labeled. For each problem:

- Highlight and copy your R commands and output and paste them into word.
- If you make any plots, copy and paste them into word:
  - Windows: Right click on the plot and select Copy as bitmap.
  - Mac: Click on the plot, then select in the menu Edit→Copy.

You can save paper by formatting your word document as two column.

You don't have to type up problems you do by hand if it's easier to do them on paper. Just put "see attached" for the problems you do using R and staple your word document to the end.