Fast-R

D-Lab

11/8/2018

Table of Contents

[Introduction to R 2](#_Toc529479509)

[How to use R to “do research” 2](#_Toc529479510)

[Navigating RStudio 2](#_Toc529479511)

[Variable Assignment 3](#_Toc529479512)

[Data types 3](#_Toc529479513)

[Data structures 4](#_Toc529479514)

[Vectors 4](#_Toc529479515)

[Data frame 4](#_Toc529479516)

[Load data from files 5](#_Toc529479517)

[Getting help 6](#_Toc529479518)

[Subsetting your data 7](#_Toc529479519)

[In one dimension $ 7](#_Toc529479520)

[In two dimensions [ , ] 7](#_Toc529479521)

[Data summarization 7](#_Toc529479522)

[Visualization 8](#_Toc529479523)

[Statistical testing 11](#_Toc529479524)

[Key points: 12](#_Toc529479525)

[Bonus 13](#_Toc529479526)

Download these materials Visit this **Error! Hyperlink reference not valid.** to download these materials. 1. Click the green “CLone or Download” button  
2. Click “Download zip” 3. Extract these files to your Desktop

This is an [.HTML](https://html.com/) file made using [RMarkdown and knitr](https://rmarkdown.rstudio.com/) and will serve as our guide. Also open the “script.R” file - that is where you will show your work!

# Introduction to R

This two-part workshop series will teach you the basics of R programming for “doing research”.

##### Click these links to learn more about D-Lab:

* [workshops](http://dlab.berkeley.edu/training)
* [working groups](http://dlab.berkeley.edu/working-groups)
* [consulting services](http://dlab.berkeley.edu/consulting)
* [data resources](http://dlab.berkeley.edu/data-resources).
  + View the [calendar](http://dlab.berkeley.edu/calendar-node-field-date) for a compact view

##### The first two hours will cover:

* How to **navigate RStudio**
* **Variable Assignment** - how to save data in R
* Different **Data Types** - numeric, integer, character, logical, factor
* Different **Data Structures** - vectors, lists, matrices, and data frames
* Loading a dataset from a .csv file on your computer
* Subsetting your data - one column at a time, or by rows and columns
* Saving your data and figures

##### The second two hours:

* **Data summarization**
* **Visualization**
* **Testing**

### How to use R to “do research”

R can do [a lot!](https://en.wikipedia.org/wiki/R_(programming_language)) R makes it easy to import, clean, summarize, visualize, and test data. Start by thinking about your own interests.

Read literature in your field to help you ask a well-informed research question - where are there gaps in the research? In which directions is research going? What is worthwhile to you? What is possible? What is too much?

You can turn these questions into sets of [testable hypotheses](http://mathworld.wolfram.com/HypothesisTesting.html) and use data summaries, visualizations, and tests to interpret your conclusions.

# Navigating RStudio

1. Script (upper left window pane)

* Scripts are where you can write your **input**
* Click File –> New File –> R Script
* Code are the instructions we write to accomplish tasks
* R syntax is highly specific and case sensitive! # Command + Enter (Mac) and Ctrl + Enter (PC) will run a line of code.

1. Console (lower left)

* This is your **output**
* You can also type directly into the console, but it is harder to save than the script. For this reason it is fun to mess around in the console.
* > is the prompt - this means RStudio is waiting for you to give it instructions.

1. Global environment (upper right)

* When you save data into R as variables (see below), they actually live in a physical place.

1. Plots, packages, help (lower right)

* Plots - this is where your visualizations will appear
* Packages - install packages here
* Help - the help files are really helpful, you just don’t know it yet!

# Variable Assignment

Data are saved into R’s memory via variables. Think of a simple three-piece recipe to assign a variable: 1. Left side: unique\_name 2. Middle: equals sign 3. Right side: value, expression, other variable, function, or code to be evaluated.

Let’s try it. In your “Fast-R.R” script. In which window pane do these variables appear in RStudio?

# define the variable  
x = 5  
  
# "call" (retrieve) the variable  
x

## [1] 5

# Data types

The class() function will tell us what type of data a variable is. The most common types are: \* Numeric - this is the default type for all numbers in R \* Integer - negative and positive whole numbers, including zero \* Character - always wrapped in quotations! " " \* Logical - only two options (TRUE and FALSE) \* Factor - categorical data

NOTE: for integer data, we must coerce (change) numeric to integer type with the as.integer(). Also try using as.numeric(), as.character(), as.logical(), and as.factor(). What happens? Does it always work?

name = "Max"  
major = "IRLE"  
age = 22  
enrolled = TRUE  
  
class(name)

## [1] "character"

class(major)

## [1] "character"

class(age)

## [1] "numeric"

class(enrolled)

## [1] "logical"

# Data structures

One obvious reason to use R is because we can store complex data inside of a single variable!

### Vectors

Vectors are organized groups of the same type of data. It doesn’t matter what type the data are, as long as they are all the same.

Use the c() function to concatenate some vectors.

# Numeric vector  
vec\_num = c(11.41, 7.23, 6.78, 8.12, 5.43, 9.34, 10.23, 5.21)  
  
# Integer vector  
vec\_int = c(1L, 4L, 6L, 8L, 12L, 14L, 7L, 3L)  
  
# Character vector  
vec\_char = c("Oakland", "SF", "Oakland", "SF", "Berkeley", "Berkeley", "Oakland", "Berkeley")  
  
# Logical vector  
vec\_logi = c(T, F, F, T)

### Data frame

Data frames are ordered groups of equal length vectors. The data.frame() function will combine equal-length vectors into a data frame.

data = data.frame(vec\_num, vec\_int, vec\_char, vec\_logi)  
data

## vec\_num vec\_int vec\_char vec\_logi  
## 1 11.41 1 Oakland TRUE  
## 2 7.23 4 SF FALSE  
## 3 6.78 6 Oakland FALSE  
## 4 8.12 8 SF TRUE  
## 5 5.43 12 Berkeley TRUE  
## 6 9.34 14 Berkeley FALSE  
## 7 10.23 7 Oakland FALSE  
## 8 5.21 3 Berkeley TRUE

We can rename columns of our data frame by passing in a vector of equal length to the colnames() function:

colnames(data) = c("Time", "Hour", "Location", "Passed")  
data

## Time Hour Location Passed  
## 1 11.41 1 Oakland TRUE  
## 2 7.23 4 SF FALSE  
## 3 6.78 6 Oakland FALSE  
## 4 8.12 8 SF TRUE  
## 5 5.43 12 Berkeley TRUE  
## 6 9.34 14 Berkeley FALSE  
## 7 10.23 7 Oakland FALSE  
## 8 5.21 3 Berkeley TRUE

# Load data from files

The read.csv() function makes it super easy to load data from files. However you first must set your “working directory”, or the folder location on your computer that RStudio is pointing to.

Click Session –> Set Working Directory –> Choose Directory, and set it to your “Fast-R” folder.

getwd() will show us the path location of this folder. dir() will show us the contents of this folder. ls() will list the variables in our global environment (same as the upper right window pane).

Let’s load the “gapminder-FiveYearData.csv” dataset as a variable named gap and explore it.

gap = read.csv("gapminder-FiveYearData.csv")  
head(gap)

## country year pop continent lifeExp gdpPercap  
## 1 Afghanistan 1952 8425333 Asia 28.801 779.4453  
## 2 Afghanistan 1957 9240934 Asia 30.332 820.8530  
## 3 Afghanistan 1962 10267083 Asia 31.997 853.1007  
## 4 Afghanistan 1967 11537966 Asia 34.020 836.1971  
## 5 Afghanistan 1972 13079460 Asia 36.088 739.9811  
## 6 Afghanistan 1977 14880372 Asia 38.438 786.1134

# View(gap) is also helpful!

What do these functions do? str(gap) nrow(gap) ncol(gap) dim(gap) names(gap)

str(gap)

## 'data.frame': 1704 obs. of 6 variables:  
## $ country : Factor w/ 142 levels "Afghanistan",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ year : int 1952 1957 1962 1967 1972 1977 1982 1987 1992 1997 ...  
## $ pop : num 8425333 9240934 10267083 11537966 13079460 ...  
## $ continent: Factor w/ 5 levels "Africa","Americas",..: 3 3 3 3 3 3 3 3 3 3 ...  
## $ lifeExp : num 28.8 30.3 32 34 36.1 ...  
## $ gdpPercap: num 779 821 853 836 740 ...

nrow(gap)

## [1] 1704

ncol(gap)

## [1] 6

dim(gap)

## [1] 1704 6

names(gap)

## [1] "country" "year" "pop" "continent" "lifeExp" "gdpPercap"

# Getting help

Remember to type a question mark before the name of a function to view its help pages

?mean  
?median  
?table  
?hist  
?t.test  
?cor.test  
?lm

# Subsetting your data

Once you get data loaded into R, you might want to clean it up (i.e., fix spelling variations, typos, formatting, etc.) but you might need to subset it - you might only be interested in certain ranges of values.

### In one dimension $

The dollar sign operator will extract one single column.

gap$country  
gap$year

### In two dimensions [ , ]

Bracket notation is an open and closed bracket with a comma in the middle. Everything before the comma refers to rows, everything after refers to columns.

For example, gap\_sub = gap[c(1:8, 10) , c(2:4)] will extract only  
\* rows 1 thru 8 and 10, and \* columns 2 thru 4 … and will save it in a new variable named gap\_sub

Let’s try it!

gap\_sub = gap[c(1:8, 10) , c(2:4)]  
gap\_sub

## year pop continent  
## 1 1952 8425333 Asia  
## 2 1957 9240934 Asia  
## 3 1962 10267083 Asia  
## 4 1967 11537966 Asia  
## 5 1972 13079460 Asia  
## 6 1977 14880372 Asia  
## 7 1982 12881816 Asia  
## 8 1987 13867957 Asia  
## 10 1997 22227415 Asia

# Data summarization

You can use the table(), prop.table(), and summary() functions to get useful information

table(gap$continent)

##   
## Africa Americas Asia Europe Oceania   
## 624 300 396 360 24

table(gap$continent, gap$year)

##   
## 1952 1957 1962 1967 1972 1977 1982 1987 1992 1997 2002 2007  
## Africa 52 52 52 52 52 52 52 52 52 52 52 52  
## Americas 25 25 25 25 25 25 25 25 25 25 25 25  
## Asia 33 33 33 33 33 33 33 33 33 33 33 33  
## Europe 30 30 30 30 30 30 30 30 30 30 30 30  
## Oceania 2 2 2 2 2 2 2 2 2 2 2 2

prop.table(table(gap$continent))

##   
## Africa Americas Asia Europe Oceania   
## 0.36619718 0.17605634 0.23239437 0.21126761 0.01408451

summary(gap)

## country year pop continent   
## Afghanistan: 12 Min. :1952 Min. :6.001e+04 Africa :624   
## Albania : 12 1st Qu.:1966 1st Qu.:2.794e+06 Americas:300   
## Algeria : 12 Median :1980 Median :7.024e+06 Asia :396   
## Angola : 12 Mean :1980 Mean :2.960e+07 Europe :360   
## Argentina : 12 3rd Qu.:1993 3rd Qu.:1.959e+07 Oceania : 24   
## Australia : 12 Max. :2007 Max. :1.319e+09   
## (Other) :1632   
## lifeExp gdpPercap   
## Min. :23.60 Min. : 241.2   
## 1st Qu.:48.20 1st Qu.: 1202.1   
## Median :60.71 Median : 3531.8   
## Mean :59.47 Mean : 7215.3   
## 3rd Qu.:70.85 3rd Qu.: 9325.5   
## Max. :82.60 Max. :113523.1   
##

summary(gap$lifeExp)

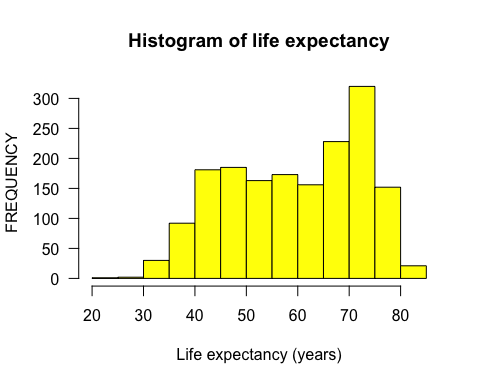
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 23.60 48.20 60.71 59.47 70.85 82.60

# Visualization

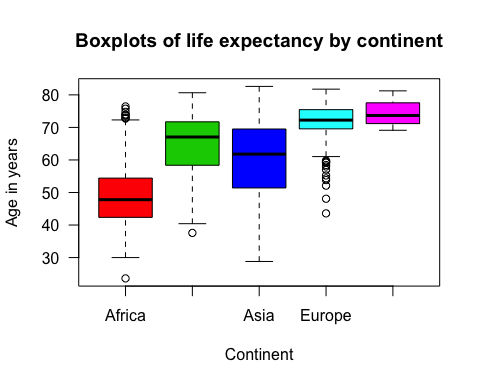
1. hist() will create a histogram of one continuous variable.
2. boxplot() will create boxplots of one factor and one continuous variable.
3. plot() will create a scatterplot of two continuous variables.

Note the other plotting arguments!

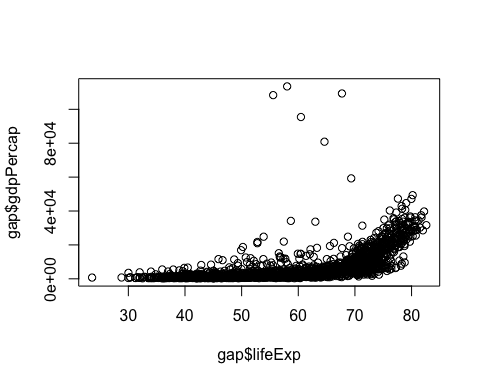
hist(gap$lifeExp,  
 # Change bar colors  
 col = "yellow",  
 # x-axis label  
 xlab = "Life expectancy (years)",  
 # y-axis label  
 ylab = "FREQUENCY",  
 # Title  
 main = "Histogram of life expectancy",  
 # Tick mark orientation  
 las = 1)



boxplot(gap$lifeExp ~ gap$continent,   
 col = 2:6,  
 xlab = "Continent",   
 ylab = "Age in years",  
 main = "Boxplots of life expectancy by continent",  
 las = 1)



plot(x = gap$lifeExp, y = gap$gdpPercap)



Click the “Export” button to save it as a .PDF and export it to .TIFF!

What does this code do? Click the first [Useful link](https://ggplot2.tidyverse.org/) to read more about part 2!

install.packages("ggplot2")  
library(ggplot2)  
?ggplot2

# Statistical testing

What does this function do? Check out [this excellent blog post](http://blog.yhat.com/posts/r-lm-summary.html) to learn more.

?lm

We can fit a linear regression model with only a small amount of code. However, remember to always check a test’s [statistical assumptions](http://www.sthda.com/english/wiki/statistical-tests-and-assumptions) before applying it!

gap\_lm = lm(gap$gdpPercap ~ gap$lifeExp)  
summary(gap\_lm)

##   
## Call:  
## lm(formula = gap$gdpPercap ~ gap$lifeExp)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -11483 -4539 -1223 2482 106950   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -19277.25 914.09 -21.09 <2e-16 \*\*\*  
## gap$lifeExp 445.44 15.02 29.66 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 8006 on 1702 degrees of freedom  
## Multiple R-squared: 0.3407, Adjusted R-squared: 0.3403   
## F-statistic: 879.6 on 1 and 1702 DF, p-value: < 2.2e-16

Learn more about linear models by [clicking here.](http://blog.yhat.com/posts/r-lm-summary.html)

# Key points:

1. Data are saved in **variables**
2. Syntax for **Variable Assignment** Looks like this:

* x = 5 (numeric)
* y = cos(pi) (numeric)
* city = "Berkeley" (character)
* value = TRUE (logical)
* factor1 = factor(variable\_name) (factor)

1. R is comprised of functions and arguments:

* **Functions** perform an action on a thing
* These “things” are called **arguments** - values, expressions, other variables, and even entire datasets. They can also be options inside of functions that we can define or turn on or off or otherwise define somehow.

1. All variables in are have a class, or type. A variable’s **class** determines how we can manipulate that data.
2. **Data structures** allow us to store more than a single number or word inside of a single variable:

* Vectors
* Lists
* Matrices
* Data frames
  + NOTE: lists and matrices will not be covered in this workshop.
* Concatenate a vector: x = c(1, 4, 18, 981) or y = c(T, F, F, T)
* Create a data frame: df = data.frame(x, y)

1. We can **load data from files** and easily load data from Excel ([.CSV files](https://www.computerhope.com/issues/ch001356.htm) are preferred), and [many other softwares and types.](https://www.datacamp.com/community/tutorials/r-data-import-tutorial)
2. These data often need to be subsetted in one ($) or two dimensions ([ , ]) to make working with the data easier.
3. Summarize

* table()
* prop.table()
* summary()

1. Visualize

* hist()
* boxplot()
* plot()

1. Test

* lm()

# Bonus

The ggplot() function works in layers. Remember to add a plus sign + to the end of a line when you want to add another layer. You need three things to make a ggplot: 1. Data 2. “aes”thetics - define your x and y coordinates, colors, shapes 3. geoms - how to represent the data - points, bars, boxes, etc.

For example:

# Make sure to first un-hashtag the below line to install the package  
# install.packages("ggplot2")  
  
library(ggplot2)  
ggplot(data = gap, aes(x = lifeExp, y = gdpPercap, color = continent, shape = continent)) +   
 geom\_point(size = 2, alpha = 0.5) +   
 ggtitle("Scatterplot of lifeExpt and gdpPercap") +   
 xlab("Life expectancy (years)") +   
 ylab("gdpPercap ($)") +   
 theme\_bw() +   
 theme(legend.position = "top") +   
 theme(plot.title = element\_text(hjust = 0.5))

