greatlearning

Introduction to R

CHAPTER I. Course Overview and Preliminary Steps

Learning objectives

- Know what is R and how it works
- Learn basics of working with data in R
- Get familiar with basic commands/functions
- Learn how to do basic analysis on any dataset and be able to create basic charts

What is R & Why we use it

- **It’s a tool** : Open-Source, cross platform, free programming language designed to build statistical solutions
- **Powerful** : Gives access to CRAN repository containing over 10,000 packages with pre-defined functions for almost every purpose
- **Stays Relevant** : Constantly being updated by users (Scientists, Statisticians, Researchers, Students!)
- **More:** Makes beautiful graphs, can create custom functions or modify existing ones, can be integrated into many environments and platforms such as Hadoop etc

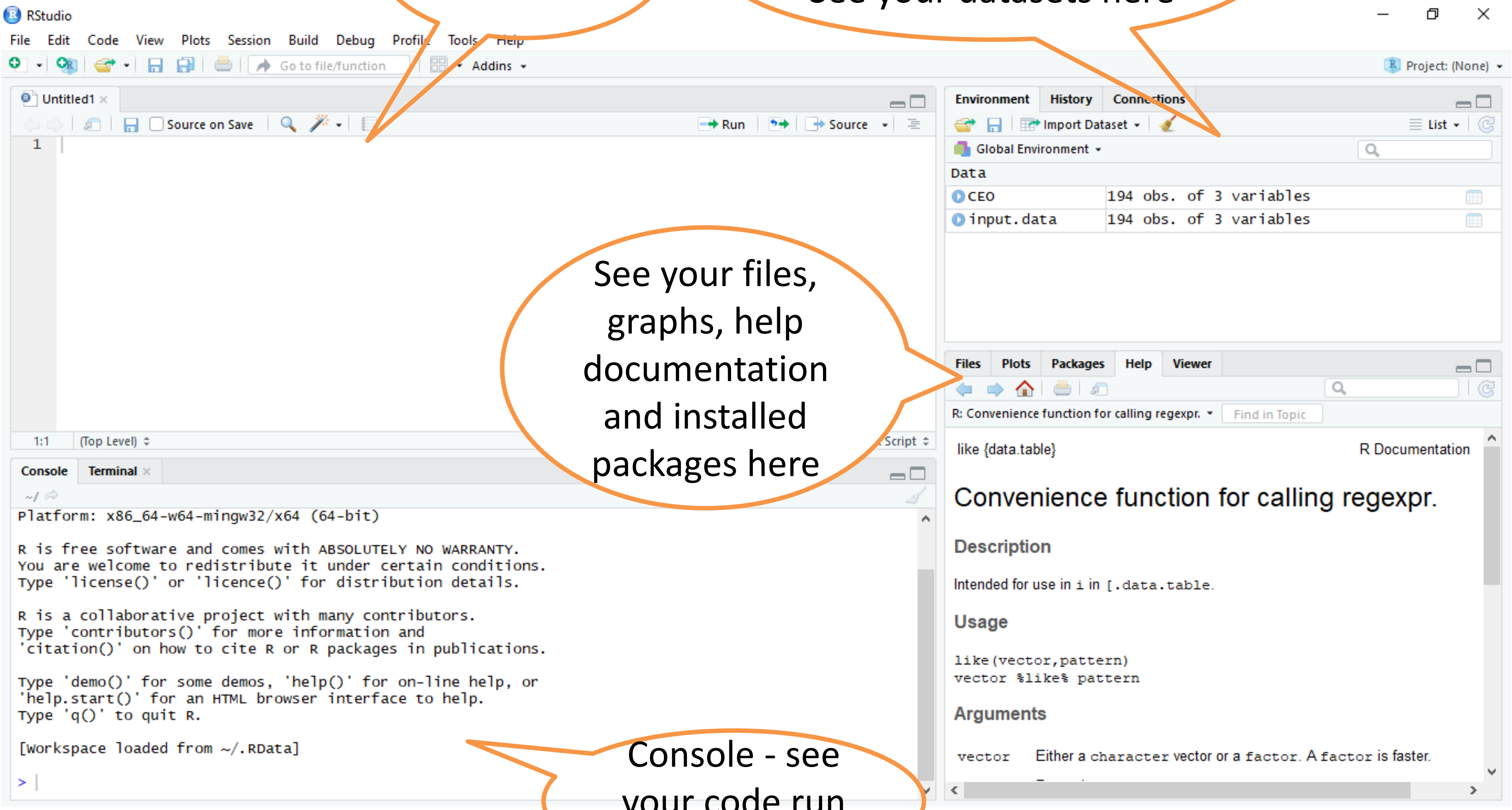
Installing R

- Can be downloaded for free from http://www.r-project.org/
- Download the version compatible with your OS
- Simple/Standard installation process

Installing R -Studio

- Can be downloaded for free from:
- https://www.rstudio.com/products/rstudio/download/
- Download the free version compatible with your OS
- R needs to be installed before installing R- Studio

R-Studio UI



Data

CEO 194 obs. of 3 variables

input.data 194 obs. of 3 variables

```
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/.RData]

> |
```

like {data.table} R Documentation

Convenience function for calling regexpr.

Description

Intended for use in `i` in `[.data.table`.

Usage

```
like(vector,pattern)
vector %like% pattern
```

Arguments

`vector` Either a `character` vector or a `factor`. A `factor` is faster.

Write your code here

Global Environment- See your datasets here

See your files, graphs, help documentation and installed packages here

Console - see your code run here

R Commands

• Assignments	E.g.: *x = 1, or x <- 1*
• Functions	E.g.: *print(“Hello World”)*
• Computations	E.g.: *17 + 3 ; x + 5*
• Mix	E.g.: *y = sqrt(16); y = 15 + 5*

- Assignment queries will update objects in your R environment
- Queries without assignment, as well as ‘call’ of R objects will either generate an output in the console, or in the plot tab

CHAPTER II. R Basics: Data Types

Variable Assignment in R

- A basic construct in programming is "variable"
- A variable allows you to store a piece of data (‘datum’, e.g. 6, ‘Hello’, etc..) or several pieces of data of a common type, and assign them a unique name
- You can then later ‘call’ this variable's name to easily access the value(s) that is/are stored within this variable.

Careful, R is case sensitive: The variables ‘x’ and ‘X’ can coexist in R environment and have different values.

Basic data types in R

- R works with numerous data types. The most common types are:
- Decimals values like 3.5, called 'numeric'
- Natural numbers like 3 are called 'integers'. Integers are also numeric
- Boolean variables (TRUE or FALSE) are classified as ‘logical’
- Text (or string) values are classified as 'character’

Basic data types in R

- Categorical variables are called ‘factors’. They have a finite and defined set of values they can take (e.g. eye_color can take have a value contained in {‘blue’, ‘green’, ’brown’, ‘black’})
- Other variables can contain time data such as dates, day of the week, hours, minutes, etc..

CHAPTER III. R Basics: Data Structures

R Objects: Vectors

- To assign multiple values to a variable, we can use an R object called a ‘vector’
- A vector is a sequence/collection of data elements of the same basic type. Members in a vector are officially called components. For Example: my_*vector = c(14,26,38,30)*
- To access a specific element in the vector, we simply need to call variable_name[i], ‘i’ being the element’s position in the vector. For example: *vect[3]* would return 38

R Objects: Matrices

- A matrix is a sequence/collection of data elements of the same basic type arranged in a two-dimensional rectangular layout.

- Being a 2-dimensional object, in order to obtain a specific value within the matrix, 2 coordinates needs to be entered. For example: *my_matrix[i,j]* would return the element on the ith row, in the jth column

- my_matrix[i,] would return the entire ith row

- my_matrix[,j] would return the entire jth column

R Objects: Data Frames

- A data frame is used for storing data tables. It is a list of vectors of equal length. Unlike matrices, it can gather vectors containing **different** basic types

- Selection of specific elements in data frames works the same way as for matrices. For example: *my_dataframe[i,j]* would return the element on the ith row, in the jth column

R Objects: Lists

- A list in R allows you to gather a variety of objects under one name (that is, the name of the list) in an ordered way. These objects can be matrices, vectors, data frames, even other lists, etc. It is not even required that these objects are related to each other.
- To access the i$_{th}$ object in the list, *write my_list*[[i]]
- If you want to access a variable in the i$_{th}$ object in the list, *write my_list*[[i]] [variable coordinates]. See examples in R

CHAPTER IV. Importing Packages and Datasets, Viewing Data

R: Packages

- R Packages are collections of R functions and data sets
- Some standard ones come with R installation
- Others can be installed in a few clicks in Rstudio, or using **install.packages("package name")** function. You can choose the CRAN Mirror closest to your location, but the default Rstudio is consistently good all over the world.
- Some have to be downloaded (from [http://cran.r-project.org/,](http://cran.r-project.org/) or through Google and manually installed
- Once installed we need to call the package in when needed using **"library("package name")"**

R: Importing Data

- More often than not, data is already available in different formats ready to be imported to R.

- R accepts files of many formats, we will learn importing files of the following formats:
  - Text (.txt)
  - CSV (.csv)
  - Excel (.xls)

R: Importing Data

- Text files: use *read.table()* for space separated files, comma separated files etc..
- CSV files: use *read_csv()* from **readr** package (used by Rstudio interface)
- Excel files: use *read_excel()* from **readxl** package (used by Rstudio interface)

See Rstudio examples to set Working Directory and import different datasets

R: Importing Data

- For more formats (such as SPSS, SAS, STATA files etc…) you can visit http://cran.rproject.org/doc/manuals/R-data.pdf , here you get information on how to import image files as well !

Data Views

There are several ways to look at a data set:

- First, you can simply look at it entirely by double clicking on it in the Global Environment, or by using *View(data_name)* function
- You can look a specific column by calling it. E.g. *data-name$column_name*
- Else, you can look at the first *k* rows, or the last k rows by using *head(data_name, k)* or *tail(data_name, k)* respectively

Data Overviews

You can also use functions to have a quick overview of the data set you are working with:

- Try to *use summary(data_name)*

- *You can also use str(data_name)*

CHAPTER V. Data Manipulations

Filtering/Subsetting

- Use a Logical Operator
- ==, >, <, <=, >=, != are all logical operators.
- Note that the “equals” logical operator is two "==" signs, as one "= " only is reserved for assignment.
- Result is a Logical variable
- To filter out rows in a dataset, place logic condition(s) in the dataset’s squared brackets, before the coma
- You can filter using several conditions and separate them with logical operators “|” (OR) and/or “&” (AND)

- See examples in Rstudio

Binding

- **Binding columns:** If 2 datasets, a dataset and a vector, or 2 vectors have the same number of values (rows in the case of datasets), they can be placed together into one same dataset using cbind()
- This is different from « merging » (see later chapter), hence there is no row matching system: rows need to be in the exact same order for the data to make sense.
- See example in Rstudio
- **Binding rows:** If 2 datasets have the same columns (order, data types, names), one can be appended under the other using rbind()
- See example in Rstudio

Transforming

- You can create new columns or modify existing ones by applying transformations to them

- Transformations can be adding, subtracting, multiplying, dividing, powering etc..
- But it can also be using functions such as log(), exp() etc..

- See examples in R studio

Sorting

- In R, you can sort your dataset’s rows based on a column’s alphabetical order (character variables), or numerical order (numeric variables)
- You can apply an ascending or descending direction to this order

- See examples in R studio

CHAPTER VI. Joins, Summary Tables and Data Export

Joins

- Joining consists in combining 2 or more datasets' rows based on a common column/field between them
- For a join to happen, 2 datasets need at least one same column. It matches rows that have identical values in this column.
- Eg.

Table 1

Column A	Column B
A1	**B1**
A2	**B1**
A3	**B2**

Table 2

Column B	Column C
B2	C1
B1	C2
B2	C3

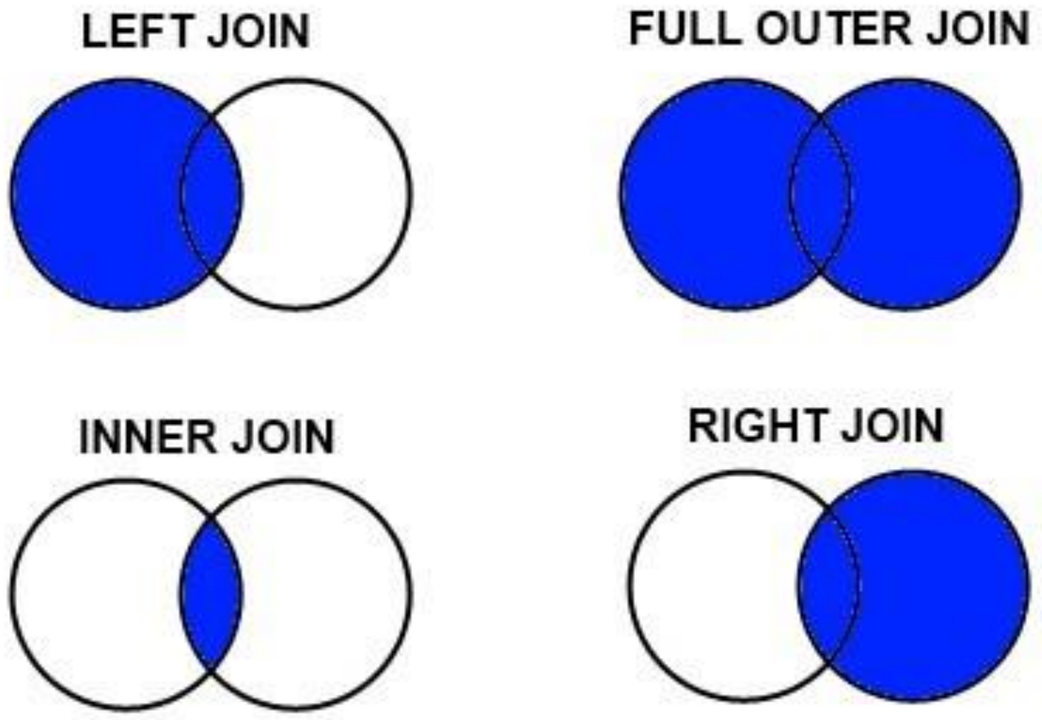
Joined Tables

Column A	Column B	Column C
A1	**B1**	C2
A2	**B1**	C2
A3	**B2**	C1
A3	**B2**	C3

- **Note:** It is not like what the *cbind()* function does: *cbind()* fuses datasets by pasting them one next to the other, regardless of what is in the data

Joins

- There are different types of joins :



Summary Tables

- **Contingency tables:** Use *table(cat_var1,cat_var2)* (where *cat_var1* and *cat_var2* are categorical variables) to obtain the observations count for each combination of these variables' levels.

- **Diverse summary tables:** Use *data %>% group_by(cat_var1) %>% summarise()* from the "Dplyr" package to aggregate datasets and obtain the summary numbers you want.

- See examples in Rstudio

Export Data

- **Export data to use outside of R:** You can export your datasets as .csv files using the *write.csv()* function.

- **Export data for later use in R:** You can export your datasets as R objects called .RDS files using *saveRDS()*. You can import them into R using *readRDS()*. These execute a lot faster.

- See examples in Rstudio

CHAPTER VII. Plots

Plots

- Plots (Graphs, Visualisations,..) are very powerful tools. They allow you to quickly grasp trends and patterns in data sets, some of which could not be spotted by analysing summary tables only

- In R, ‘ggplot2’ package gives you endless possibilities to create visualisations.

- In this video, we focus on *qplot()* function (from ‘ggplot2’), which can provide high quality graphs with very little effort.

Plots

With *qplot()*, we can create:

- **Histograms** and **Density plots** to visualise Numerical variables
- **Bar plots** to visualise categorical variables
- **Box plots** to visualise correlations between numerical and categorical variables
- **Dot Plots** to visualise correlations between numerical variables

We can also use color coding to add information to graphs while keeping them easily interpretable. See examples in R studio.

Plots

Finally, you can save your graphs as images.

Simply use the *ggsave()* function from the ggplot2 package

See examples in Rstudio

Thank you