PROGRAMMING IN R AND PYTHON

SETTING THE STAGE

CONTENTS

1. R and Python: Background and Comparison
2. Resources for Programming
3. Exercises for Hands-on Learning
4. Optional Exercises and Readings

A BIT OF HISTORY

R:

- a successor to “S”
- developed by statisticians as a ‘statistical programming language’
- built-in data structures and functionality intended to make working with data easier
- gained prominence as a free and open source alternative to expensive statistical software

Python:

- created in the early 90’s but popularized in the 00’s
- intended to be easy to read, easy to understand and easy to learn, relative to other OOLs
- has a massive base of open-source modules

[for a comprehensive comparison, consult https://www.datacamp.com/community/tutorials/r-or-python-for-data-analysis]

COMPARISON

R:

- technically object oriented, but this tends to be a bit hidden in practice
- lends itself to quick interactive scripting, data exploration
- has special built-in notation for statistical models
- has a special data type – the data frame – for handling datasets

Python:

- object oriented
- lends itself to writing structured, pre-designed computer code.
- intended to be a general programming language
- designed to create code that is easy to read

A NOTE : VECTORIZATION IN INTERPRETED LANGUAGES

High-level interpreted languages are slower than low-level/ compiled languages.

To get around this, these languages will sometimes hand off (behind the scenes) certain types of operations to functions written in lower-level languages (like C).

In order to take advantage of this, the R and Python, communities emphasize a certain programming strategy when using lists/vectors/arrays.

In particular, they avoid cycling through each item of a list, and instead often use special functions that **map** a chosen function or operation to every item in the list.

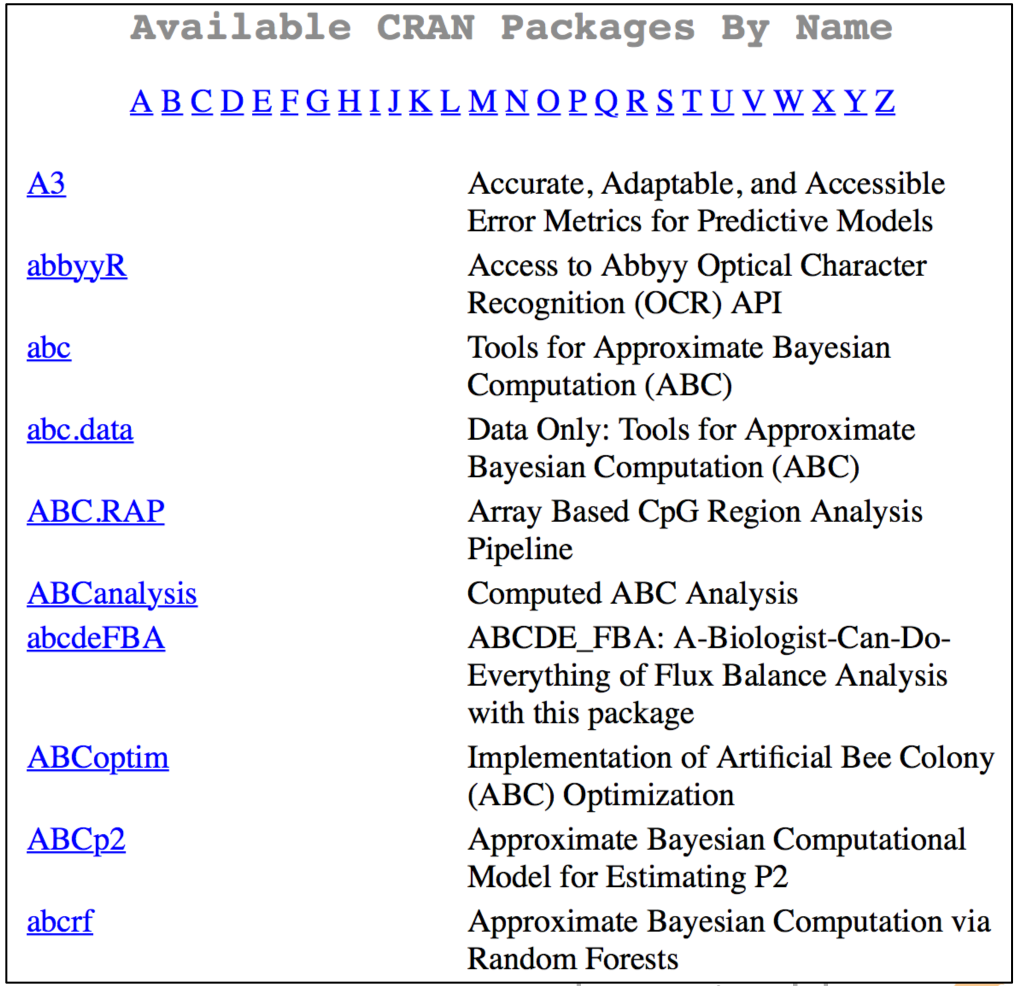
This can run counter to habits gained when learning other languages.

SO MANY PACKAGES/MODULES!

The strength of both R and Python lies in their many technical packages and modules.

These allow a programmer to implement very sophisticated functionality simply by making a few function calls.

Let's open the `RPackagesDemo` and `PythonPackagesDemo` notebooks to see some of this in action.

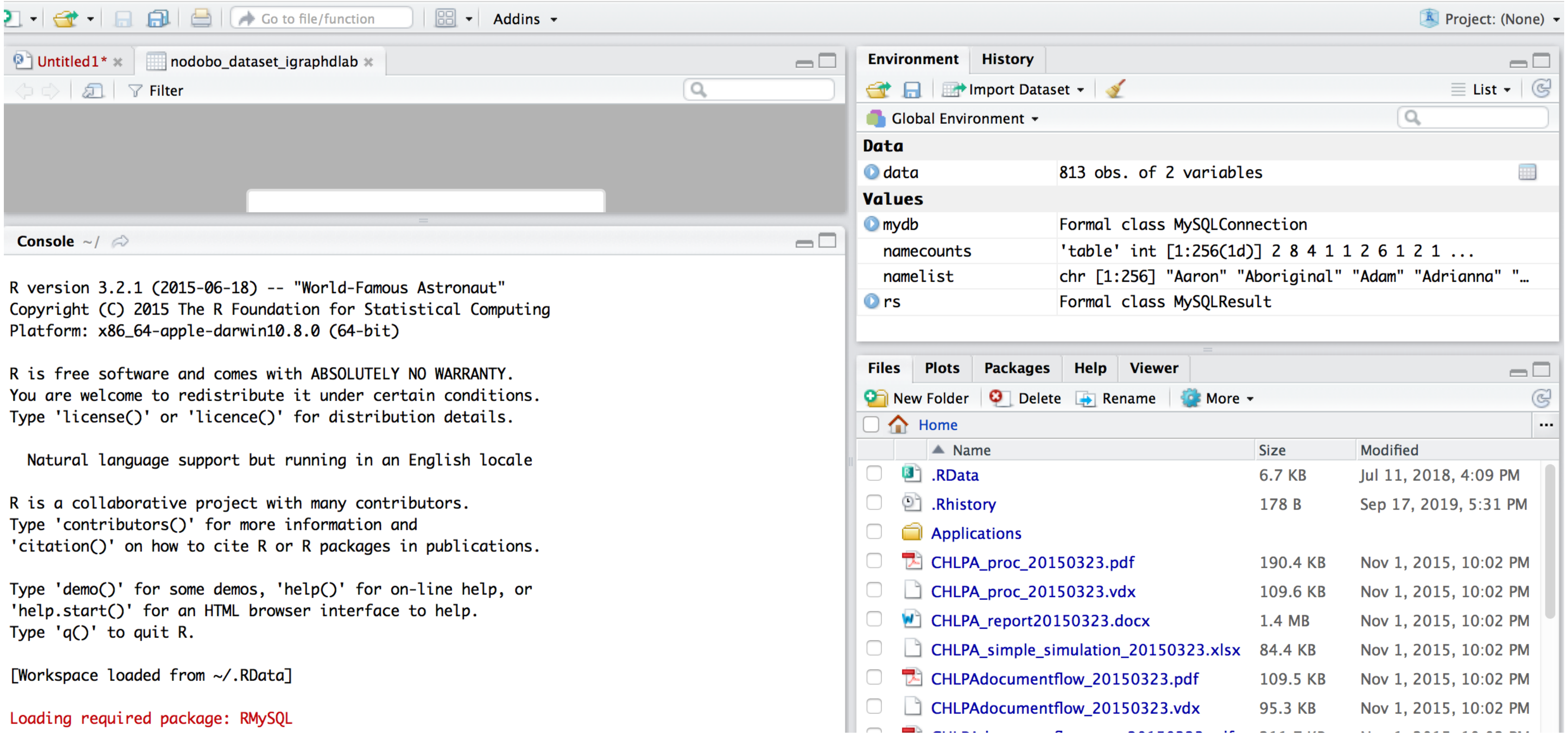


Available CRAN Packages By Name

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A3	Accurate, Adaptable, and Accessible Error Metrics for Predictive Models
abbyyR	Access to Abbyy Optical Character Recognition (OCR) API
abc	Tools for Approximate Bayesian Computation (ABC)
abc.data	Data Only: Tools for Approximate Bayesian Computation (ABC)
ABC.RAP	Array Based CpG Region Analysis Pipeline
ABCanalysis	Computed ABC Analysis
abcdeFBA	ABCDE_FBA: A-Biologist-Can-Do-Everything of Flux Balance Analysis with this package
ABCoptim	Implementation of Artificial Bee Colony (ABC) Optimization
ABCp2	Approximate Bayesian Computational Model for Estimating P2
abcrf	Approximate Bayesian Computation via Random Forests

R STUDIO



```
R version 3.2.1 (2015-06-18) -- "World-Famous Astronaut"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin10.8.0 (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.

  Natural language support but running in an English locale

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]

Loading required package: RMySQL
```

Data	
data	813 obs. of 2 variables
Values	
mydb	Formal class MySQLConnection
namecounts	'table' int [1:256(1d)] 2 8 4 1 1 2 6 1 2 1 ...
namelist	chr [1:256] "Aaron" "Aboriginal" "Adam" "Adrianna" "…
rs	Formal class MySQLResult

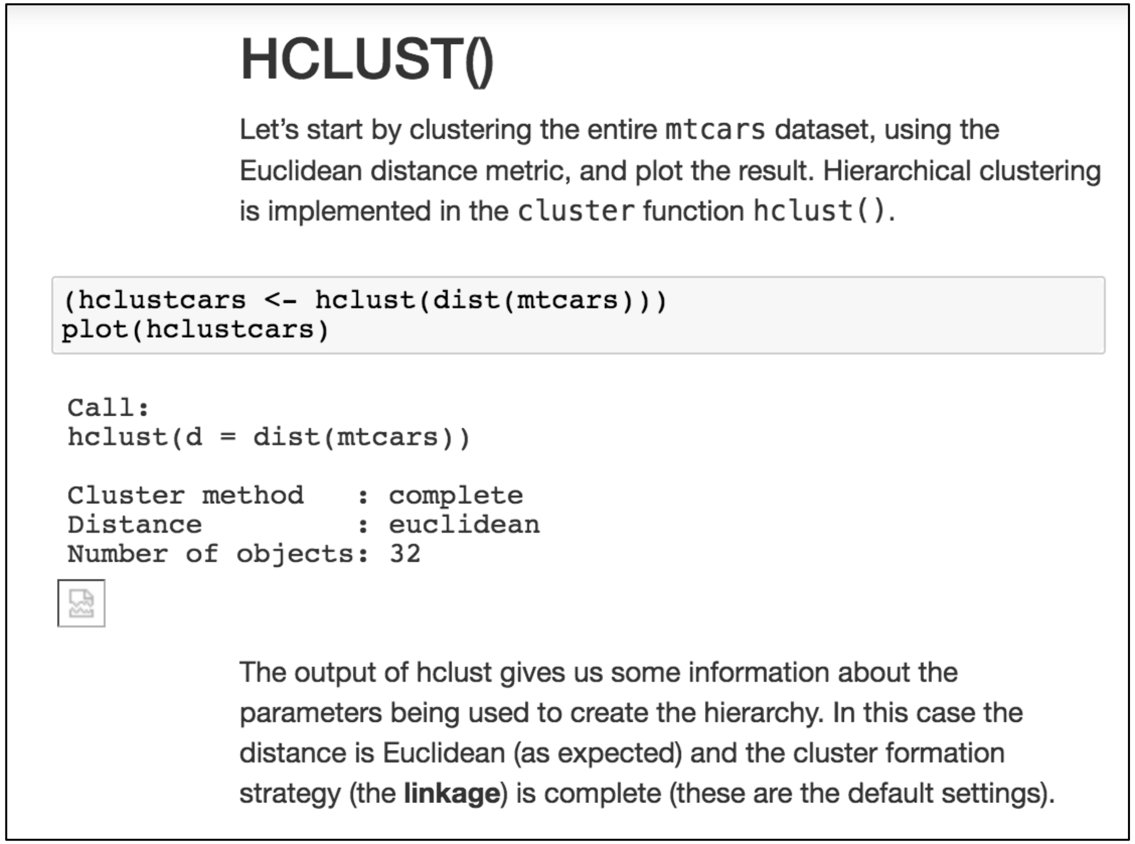
Name	Size	Modified
.RData	6.7 KB	Jul 11, 2018, 4:09 PM
.Rhistory	178 B	Sep 17, 2019, 5:31 PM
Applications		
CHLPA_proc_20150323.pdf	190.4 KB	Nov 1, 2015, 10:02 PM
CHLPA_proc_20150323.vdx	109.6 KB	Nov 1, 2015, 10:02 PM
CHLPA_report20150323.docx	1.4 MB	Nov 1, 2015, 10:02 PM
CHLPA_simple_simulation_20150323.xlsx	84.4 KB	Nov 1, 2015, 10:02 PM
CHLPAdocumentflow_20150323.pdf	109.5 KB	Nov 1, 2015, 10:02 PM
CHLPAdocumentflow_20150323.vdx	95.3 KB	Nov 1, 2015, 10:02 PM

R AND PYTHON NOTEBOOKS

Over the course we will be providing you with many notebooks of sample code.

You can use these notebooks to:

- get a sense of what can be done
- gain exposure to many examples of the language syntax
- help you write your own code
- learn why the code works the way it does, and some theory behind the code



HCLUST()

Let's start by clustering the entire `mtcars` dataset, using the Euclidean distance metric, and plot the result. Hierarchical clustering is implemented in the `cluster` function `hclust()`.

```
(hclustcars <- hclust(dist(mtcars)))
plot(hclustcars)
```

```
Call:
hclust(d = dist(mtcars))

Cluster method   : complete
Distance         : euclidean
Number of objects: 32
```

The output of hclust gives us some information about the parameters being used to create the hierarchy. In this case the distance is Euclidean (as expected) and the cluster formation strategy (the **linkage**) is complete (these are the default settings).

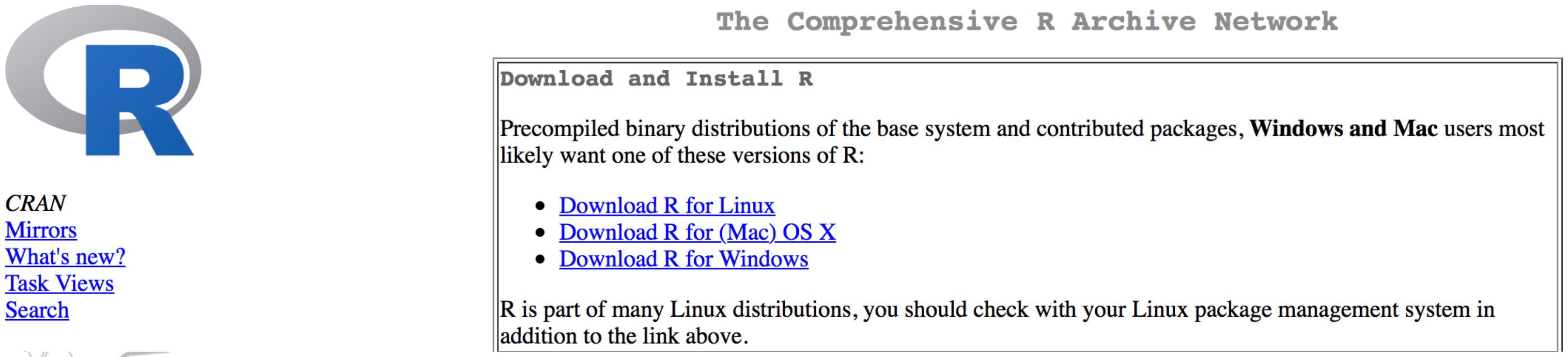
ON-LINE RESOURCES

Stack Exchange/Stack Overflow/Cross Validated

Blogs (e.g. R Bloggers)

Official Sites:

- Python Software Foundation: https://www.python.org
- Comprehensive R Archive Network (CRAN): https://cran.r-project.org



CRAN
Mirrors
What's new?
Task Views
Search

The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux
- Download R for (Mac) OS X
- Download R for Windows

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

GETTING INTO PROGRAMMING

Develop or assess R and Python skills by carrying out the following exercises in both R and Python. There is no need to do the exercises in any particular order.

You may choose to carry out each of the exercises separately, or to write a single program that carries out all of the individual exercises.

You will find much of the base code you need in this week's course notebooks, but you will need to tweak and add to this code to carry out the exercises. You will also find a lot of helpful information and code on the internet!

EXPRESSIONS, VARIABLES, DATA STRUCTURES AND OPERATORS (1)

Create three variables and assign numerical values to each of these variables.

Then write one or more statements that carry out the following types of operations using these variables: addition, subtraction, multiplication, division, raising to a power.

EXPRESSIONS, VARIABLES, DATA STRUCTURES AND OPERATORS (2)

Create three variables and assign string values to each of these variables.

Write a statement that joins the three strings into a single string. Write some code that prints the string.

Write some code that tests to see if a substring of your choosing is contained within the larger string.

EXPRESSIONS, VARIABLES, DATA STRUCTURES AND OPERATORS (3)

Create three variables and assign lists to each of these variables. Join the three lists into a new list containing three distinct sub-lists (a list of three lists).

Create a list without sub-lists (all original list elements are part of a single larger list).

Create a fourth list by splitting this resulting list in half and assigning the second half of the list to a new variable.

Extract the last item of this list (it can either stay in the original list or be removed from it) and assign this element to a variable.

STATEMENTS, BLOCKS, CONTROL FLOW, LOGICAL OPERATORS

Write a statement that contains at least three nested blocks.

Use at least three of the following control flow options: if, if else, while, for, break, continue (Python only), next, switch.

FUNCTIONS

Write a function that takes three arguments as input and returns one value.

Call the function with arguments of your choosing.

LIBRARIES/PACKAGES/MODULES

Execute the relevant command that shows a list of the packages (for R) or modules (for Python) that are currently installed in your Jupyter notebook environment.

- hint: use the internet, example notebooks and handouts to help you find the relevant command.

Use available documentation to determine what some of these do.

- hint: take a look at the Python and R notebooks that are available – you may notice some relevant information there.
- choose a module/package from this list and load the relevant package/module if necessary.

Write some code that uses functions and objects supplied by this package.

INPUTS/OUTPUTS (1)

Print to the standard output of the Jupyter notebook (in this case, the standard output is the space below a code cell in the notebook that is generated when you run a cell) three sentences of your choosing, on three separate lines, using a single statement of code.

INPUTS/OUTPUTS (2)

Locate a comma separated values (CSV) file stored on your computer

- (Hint - there should be a folder called Data in the main notebook directory).

Read this file into the notebook and store the results in one or more variables.

INPUTS/OUTPUTS (3)

Create a new file and write four lines in CSV format to this file.

In a separate statement, write four more lines to this existing file, without overwriting the original file.

INTERPRETERS/COMPILERS

Write enough code to generate at least five different error messages from the Jupyter Notebook interpreter.

Copy these error messages into a markdown cell, and write a short note under each explaining the meaning of the error message, and how the code was fixed.

OPTIONAL EXERCISES

1. Using a language of your choice, write a function that, when passed a dataset, reports 5 interesting pieces of information about the dataset. Load a dataset and run the function on this dataset.
2. Using a language of your choice, write two functions. The output of the first function should work as the input to the second function. The first function should read in a dataset and generate a subset of the dataset based on some chosen criteria. The second function should read in a dataset and provide summary data of some type for each column in the dataset. Load a dataset and run both functions on the dataset.

OPTIONAL READINGS

Best Python Resources: https://www.fullstackpython.com/best-python-resources.html

Top R language resources to improve your data skills: https://www.computerworld.com/article/2497464/business-intelligence/top-r-language-resources-to-improve-your-data-skills.html