R PROGRAMMING WORKSHOP (BASICS) FOR RESEARCHERS

Lesson 1

Prepared by **PolyU IT Support for Research**, Aug 2020

COURSE INFO

DATE: 18 (TUE) , 21 (FRI) & 25 (TUE) AUG (3 SESSIONS)

TIME: 14:30-16:30

PLATFORM FOR HOSTING WORKSHOPS: MS TEAMS

GIT REPO URL:

https://polyu.hk/dAvpL

WHO AM I?

ALEX LAM

- SENIOR SPECIALIST
- INFORMATION AND TECHNOLOGY SERVICE
- HAS EXTENSIVE PROGRAMMING EXPERIENCE ACROSS C, PYTHON, PHP, JS AND R

CONTACT PERSON

Tommy Ng

- Department: School of Professional Education and Executive Development, The Hong Kong Polytechnic University
- Location: Room S1409, 14/F, South Tower, PolyU West Kowloon Campus
- tommy.ng@speed-polyu.edu.hk

REQUIREMENT

- RESEARCHER ORIENTED
- BASIC UNDERSTANDING OF ANY OF THE PROGRAMMING LANGUAGES

WHAT YOU WILL LEARN

COURSE OUTLINE

- INTRODUCTION TO R
- IDE TOOLS FOR R WITH ANACONDA
- MODULES & PACKAGES
- VARIABLES & TYPES
- BASIC OPERATORS
- BASIC STRING OPERATIONS
- DATA STRUCTURE
- CONDITIONS
- LOOPS
- FUNCTIONS
- DATA PROCESSING WITH DATAFRAME
- BASICDATA VISUALIZATION WITH GGPLOT
- BASIC STATISTICS WITH R

WHAT IS R & WHY R?

- R IS A PROGRAMMING LANGUAGE AND SOFTWARE ENVIRONMENT
- GOOD AT STATISTICAL ANALYSIS, GRAPHICS REPRESENTATION AND REPORTING.
- CREATED BY ROSS IHAKA AND ROBERT GENTLEMAN AT THE UNIVERSITY OF AUCKLAND
- FREELY AVAILABLE UNDER THE GNU GENERAL PUBLIC LICENSE
- NO COMPILATION NEEDED
- ENABLE FAST PROTOTYPE

R VS PYTHON

	R	Python
Usage	Statistical modeling, data analysis	Web development, Data analysis, scientific calculation
Users	Statisticians and data scientist. Common in R&D institutes	Software engineers and data scientist Widely used in varies industries.
Visualization	ggplot2	Matplotlib

REFERENCES

- HANDS ON PROGRAMMING WITH R (https://rstudio-education.github.io/hopr/index.html)
- R FOR DATA SCIENCE (HTTPS://R4DS.HAD.CO.NZ/)
- TUTORIALPOINT HAS GOOD QUICK REFERENCES (HTTPS://WWW.TUTORIALSPOINT.COM/R/INDEX.HTM)
- UDACITY "DATA AND VISUAL ANALYTICS (HTTPS://CLASSROOM.UDACITY.COM/COURSES/UD404)

REFERENCES

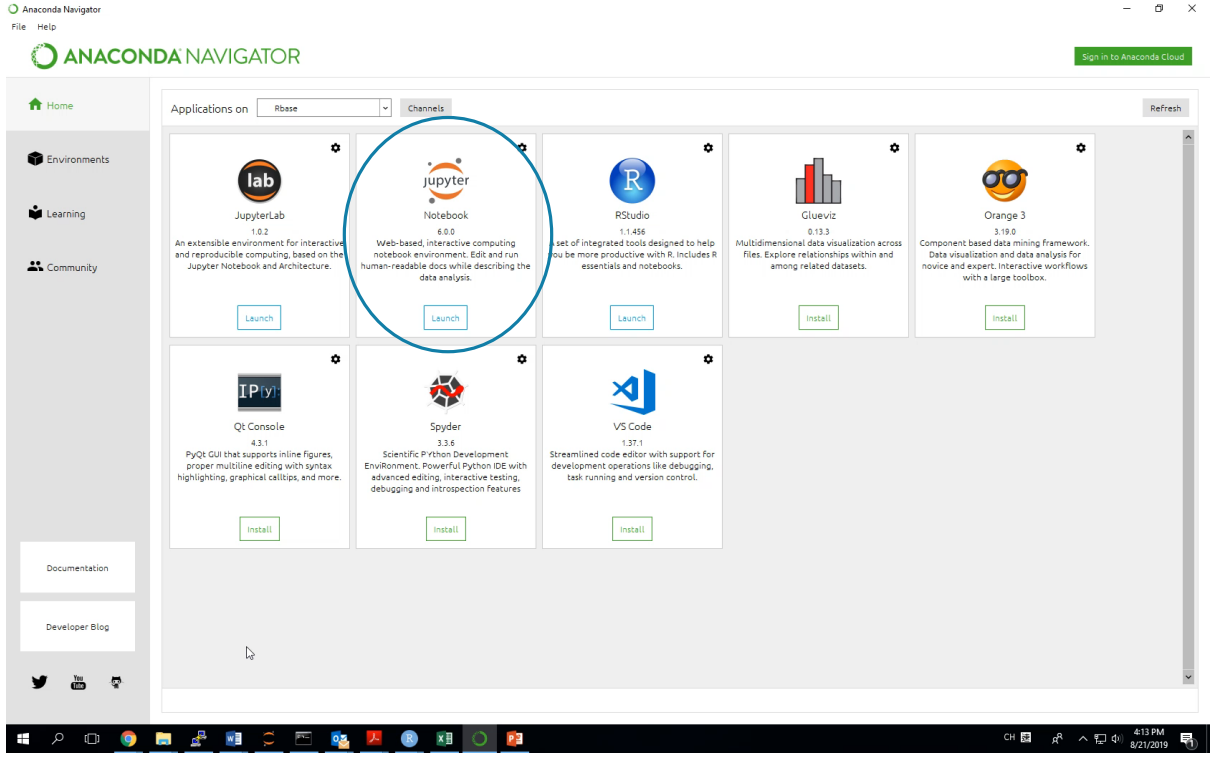
- INSTALLATION OF ANACONDA WITH RSTUDIO (HTTPS://DOCS.ANACONDA.COM/ANACONDA/NAVIGATOR/TUTORIALS/CREATE-R-ENVIRONMENT/)
- ONLY RSTUDIO (HTTPS://RSTUDIO.COM/PRODUCTS/RSTUDIO/DOWNLOAD/)

INTRODUCTION TO R IDE PLATFORMS

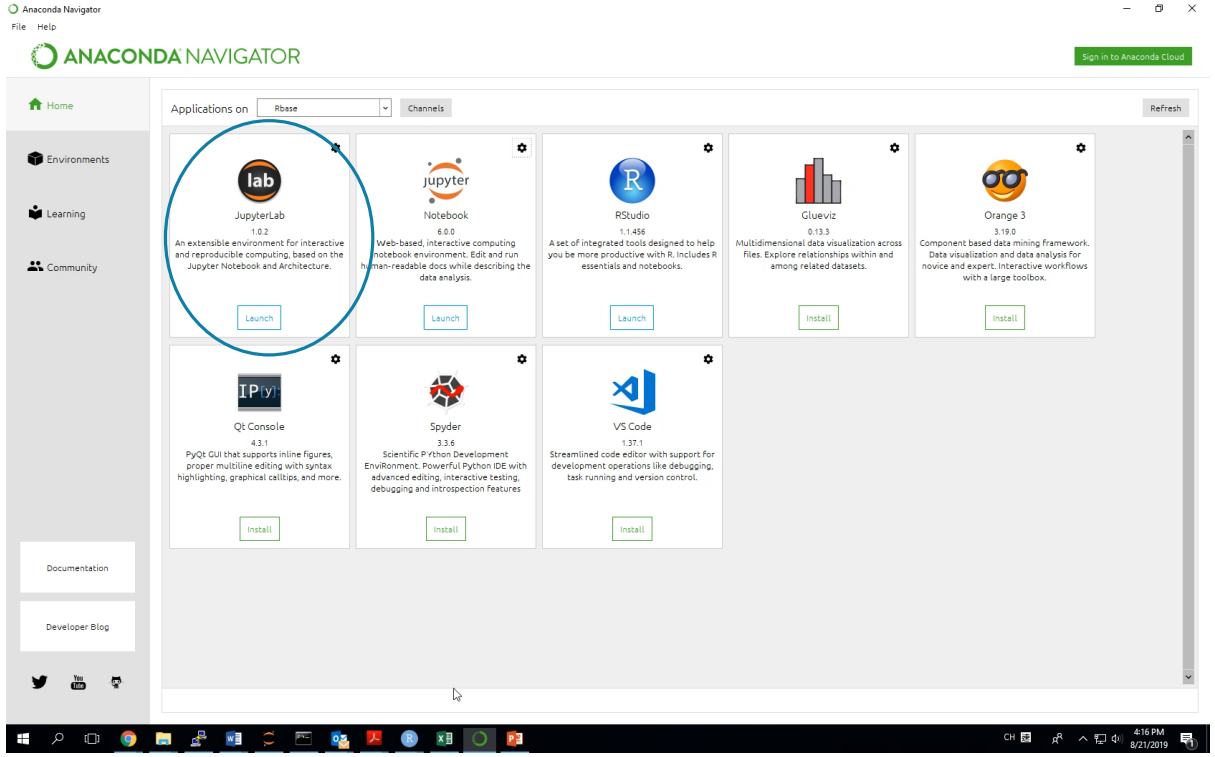
- ANACONDA IS A FREE AND OPEN-SOURCE DISTRIBUTION OF THE PYTHON AND R PROGRAMMING LANGUAGES FOR SCIENTIFIC COMPUTING
- AIMS TO SIMPLIFY PACKAGE MANAGEMENT AND DEPLOYMENT
- ANACONDA
  - *R STUDIO*
  - *JUPYTER NOTEBOOK*
  - *JUPYTER LAB*
- FOCUS ON RSTUDIO IN THIS COURSE

ANACONDA

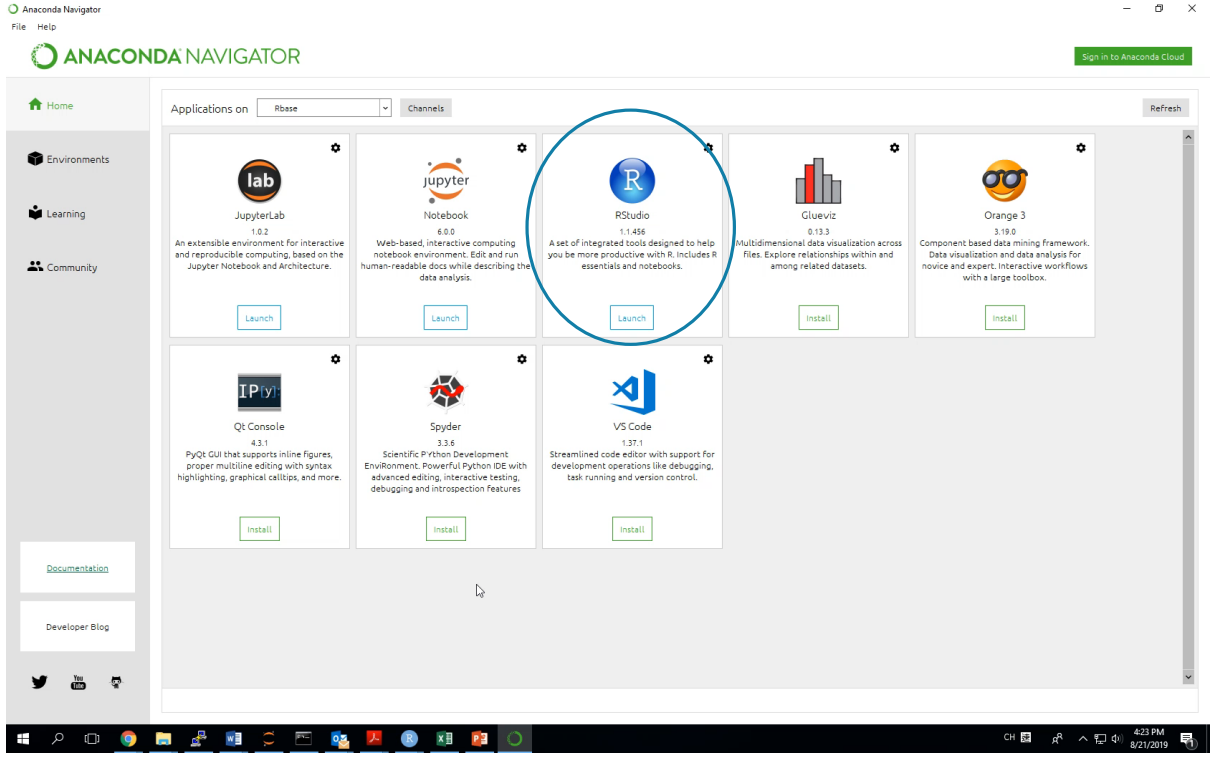
JUPYTER NOTEBOOK



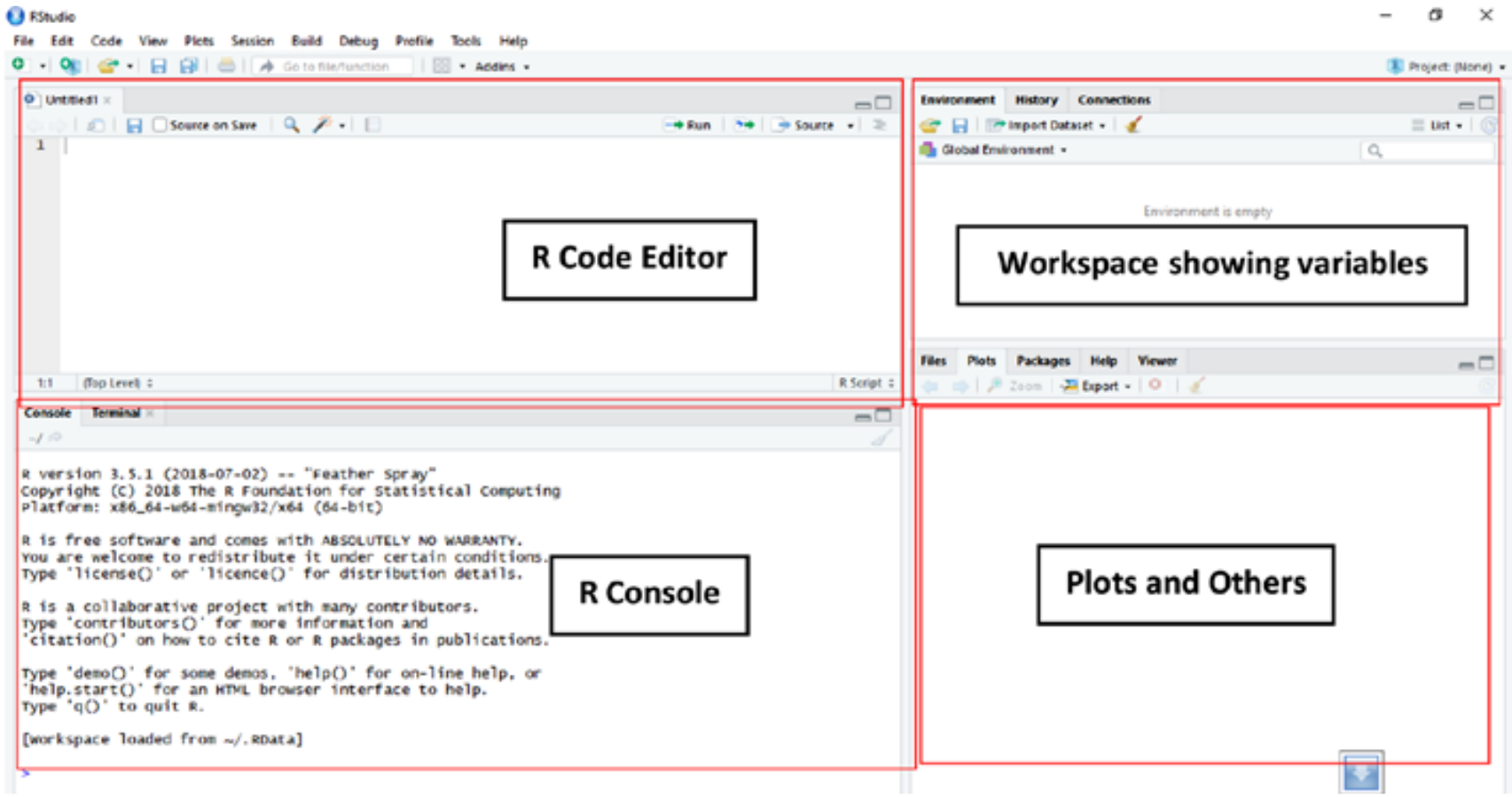
JUPYTER LAB


Anaconda Navigator
File Help
ANACONDA NAVIGATOR
Sign in to Anaconda Cloud
Home
Environments
Learning
Community
Documentation
Developer Blog
Applications on
Rbase
Channels
Refresh
JupyterLab
1.0.2
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.
Launch
Notebook
6.0.0
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.
Launch
RStudio
1.1.456
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.
Launch
Glueviz
0.13.3
Multidimensional data visualization across files. Explore relationships within and among related datasets.
Install
Orange 3
3.19.0
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.
Install
Qt Console
4.3.1
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.
Install
Spyder
3.3.6
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features
Install
VS Code
1.37.1
Streamlined code editor with support for development operations like debugging, task running and version control.
Install
4:16 PM
8/21/2019

RSTUDIO


Anaconda Navigator
File Help
ANACONDA NAVIGATOR
Sign in to Anaconda Cloud
Home
Environments
Learning
Community
Applications on
Rbase
Channels
Refresh
JupyterLab
1.0.2
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.
Launch
Notebook
6.0.0
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.
Launch
RStudio
1.1.456
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.
Launch
Glueviz
0.13.3
Multidimensional data visualization across files. Explore relationships within and among related datasets.
Install
Orange 3
3.19.0
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.
Install
Qt Console
4.3.1
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.
Install
Spyder
3.3.6
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features
Install
VS Code
1.37.1
Streamlined code editor with support for development operations like debugging, task running and version control.
Install
Documentation
Developer Blog
4:23 PM
8/21/2019

RSTUDIO


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Untitled1
Source on Save
Run
Source
R Code Editor
Environment History Connections
Import Dataset
Global Environment
Environment is empty
Workspace showing variables
Files Plots Packages Help Viewer
Console Terminal
R version 3.5.1 (2018-07-02) -- "Feather Spray"
Copyright (C) 2018 The R Foundation for Statistical Computing
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]
R Console
Plots and Others

HELP DOCUMENT OF R

- HELP(THE COMMAND TO BE CHECKED)
- FOR EXAMPLE:

  - *HELP(PLOT)*

  - *HELP(LIST)*

- EXERCISE:

  - *TRY TO LOAD THE DOCUMENT OF EACH OF THE FOLLOWING COMMAND:*

  - *AS*

  - *IS*

  - *MATRIX*

COMMAND FOR WORKSPACE

- LS()

  - LIST VARIABLE(S) IN WORKSPACE MEMORY

- SAVE.IMAGE(FILE="TEST")

  - SAVE THE WHOLE WORKSPACE TO A FILE

- SAVE(A,FILE="TESTA")

  - SAVE SPECIFIC VARIABLE(S) TO A FILE

- LOAD("TEST")

  - LOAD WORKSPACE VARIABLE(S) FROM A FILE

- GETWD()

  - GET CURRENT DIRECTORY

- SETWD(DIR)

  - SET WORKING DIRECTORY

MODULE AND PACKAGES

- INSTALL.PACKAGES("GGPLOT2")
- LIBRARY("GGPLOT2")

VARIABLE & R-OBJECT

- IN R, THE VARIABLES ARE NOT DECLARED AS SOME DATA TYPE AS C,JAVA ETC
- VARIABLES ARE ASSIGNED WITH R-OBJECTS.
- DATA TYPES OF THE R-OBJECT BECOMES THE DATA TYPE OF THE VARIABLE
- R-OBJECTS ARE BUILT UPON THE ATOMIC VECTORS.
- “ATOMIC VECTORS” HAS SINGLE VALUE

VARIABLE AND ASSIGNMENT

- DECLARE VARIABLE

  - A<-5

  - B<-"POLYU"

- FOR VARIABLE WITH MULTIPLE VALUES, USE C() TO COMBINE VALUES INTO VECTOR

  - V<-C(1,2,3,4)

  - G<-C('A','B','C','D')

- ALL VALUE SHOULD BE IN THE SAME TYPE

- FOR VARIABLE WITH MULTIPLE VALUES AND DIFFERENT TYPES, USE C() TO DECLARE VECTOR AS A INPUT TO LIST()

  - LIST1 <- LIST(C(18,4,6),'POLYU','HK',TRUE)

- WILL EXPLAIN DATA TYPES AND DATA STRUCTURE FOR NEXT COUPLE OF SLIDES

- <- , = AND -> CAN BE USED, BUT <- IS RECOMMENDED

DATA TYPE

The variables are assigned with R-Objects and the data type of the R-object becomes the data type of the variable

DATA TYPE	EXAMPLE	Code	Result
Logical	TRUE, FALSE	v <- FALSE print(class(v))	?
Numeric	1.3,4.1,50,666.1	v <- 666.1 print(class(v))	?
Integer	5L, 88L,1091L	v <- 2L print(class(v))	?
Complex	4+5i	v <- 2+5i print(class(v))	?
Character	'a' , "good", "TRUE", '23.4'	v <- "TRUE" print(class(v))	?
Raw	"Hello" is stored as 48 65 6c 6c 6f	v <-charToRaw("Hello") print(class(v))	?

DATA TYPE

The variables are assigned with R-Objects and the data type of the R-object becomes the data type of the variable

DATA TYPE	EXAMPLE	Code	Result
Logical	TRUE, FALSE	v <- FALSE print(class(v))	
Numeric	1.3,4.1,50,666.1	v <- 666.1 print(class(v))	
Integer	5L, 88L,1091L	v <- 2L print(class(v))	
Complex	4+5i	v <- 2+5i print(class(v))	
Character	'a' , "good", "TRUE", '23.4'	v <- "TRUE" print(class(v))	
Raw	"Hello" is stored as 48 65 6c 6c 6f	v <- charToRaw("Hello") print(class(v))	

DON’T DO THIS

- D<-’1’
- E<-999
- D+E
- WHAT WOULD YOU GET?
- HOW TO SOLVE IT?

CONVERTING DATA TYPE

F <- AS.NUMERIC(D)

F + E

- AS.INTEGER
- AS.LOGICAL
- AS.CHARACTER
- AS.RAW
- SEE THE DOCUMENTATION WHEN YOU NEED THEM

BASIC STRING

- BEGINNING AND END OF A STRING SHOULD BE BOTH DOUBLE QUOTES OR BOTH SINGLE QUOTE
- DOUBLE QUOTES CAN BE INSERTED INTO A STRING STARTING AND ENDING WITH SINGLE QUOTE
- SINGLE QUOTE CAN BE INSERTED INTO A STRING STARTING AND ENDING WITH DOUBLE QUOTES
- PASTE(…)- FOR CONNECTING STRINGS
- FORMAT(..)- FOR FORMATTING STRINGS
- NCHAR(X)- USED FOR COUNTING NUMBER OF CHARACTERS
- TOUPPER(X) AND TOLOWER(X)- FOR ALTERING CASE

BASIC STRING

Exercise 1

Produce the string :"May,the force,be with,you“

Using paste() as seen in the previous example

DATA STRUCTURE AT A GLANCE

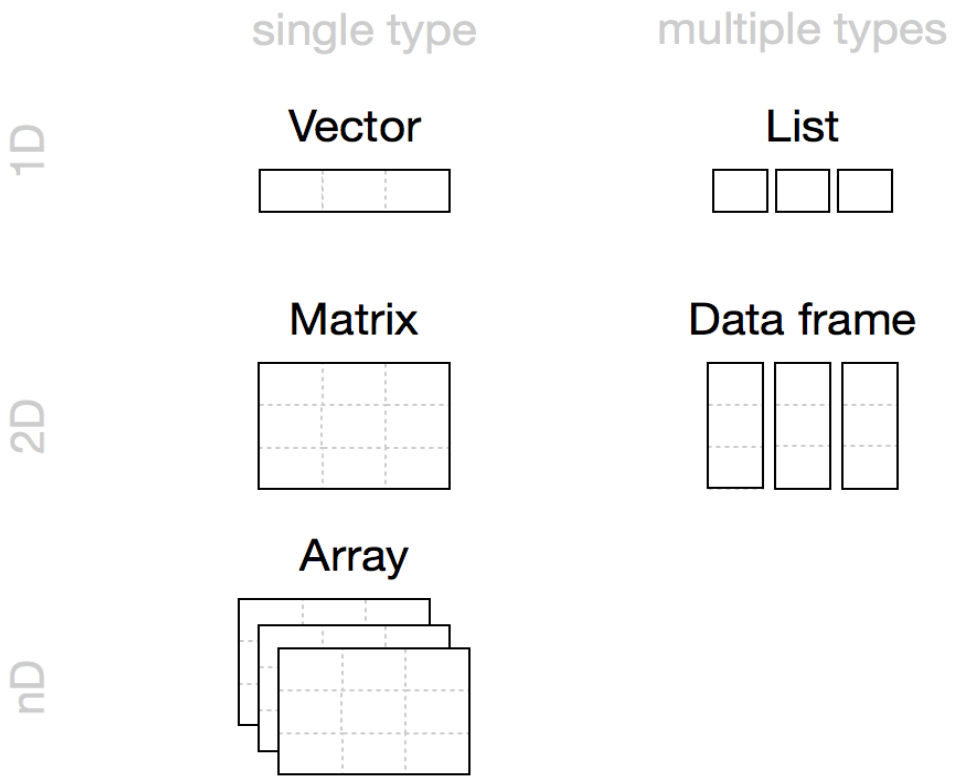
- Commonly used data structure
  - VECTORS
  - LISTS
  - MATRICES
  - ARRAYS
  - FACTORS
  - DATA FRAMES

Dimension	Type	
	Homogeneous	Heterogeneous
1d	Vector	List
2d	Matrix	Data Frame
nd	Array	N/A

From:* *Hadley Wickham's book Advanced R

- Build up some intuition now and we will go through each of them in detail later

DATA STRUCTURE AT A GLANCE



VECTORS

- As seen previously , use c() to declare vector

```
# Create a vector.
car<- c('BMW','TOYOTA',"AUDI")
print(car)
# Get the class of the vector.
print(class(car))
```

- Forms for creating numerical sequence

```
#Short from of sequence of numerical values with interval ==1
number<- 1:10
#Sequence of number with specific interval
number2<- seq(1,10, by=0.5)
```

- More on later slides

VECTORS

- Empty vector

```
y=vector(mode="logical",length=9)
z=vector(mode="numeric",length=4)
```

- Short form for creating repetitive numerical sequence

```
rep(1:4, 2)
rep(1:4, each = 2) # not the same
rep(1:4, c(2,2,2,2)) # same as second
rep(1:4, c(2,1,2,1))
rep(1:4, each = 2, len = 4) # first 4 only
rep(1:4, each = 2, len = 10) # 8 integers plus two recycled 1's
rep(1:4, each = 2, times = 3) # length 24, 3 complete replications
```

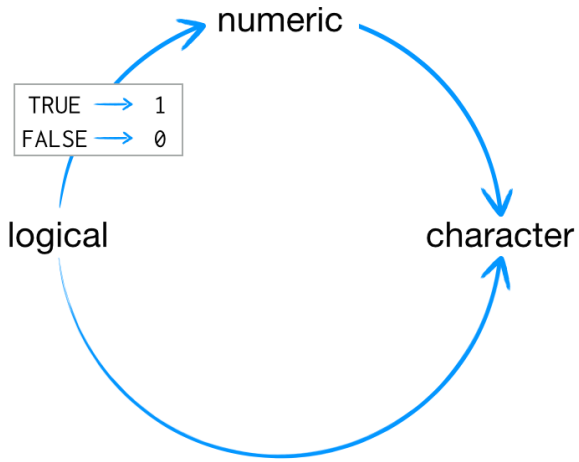
VECTORS

- What if:

```
# Create a vector.
data<- c('BMW','TOYOTA',1)
print(data)
# Get the class of the vector.
print(class(data))
# Create a vector.
data1<- c(TRUE,FALSE,1)
print(data)
# Get the class of the vector.
print(class(data))
```

VECTORS COERCION

- If a character string is present in as vector, R will convert everything else in the vector to character strings
- If a vector only contains logical values and numbers, R will convert the logical values to numbers; every TRUE becomes a 1, and every FALSE becomes a 0



LISTS

- Similar to vectors
- Contains many different types of elements
- Vectors contains one type of elements

```
# Create a list.
list1 <- list(c(2,5,3),21.3,sin)

# Print the list.
print(list1)

print(class(list))
```

MATRICES

- A matrix is a two-dimensional rectangular data set.
- As vector, element is selected in this form m[i][j]

```
# Create a matrix by row.
M <- matrix( c('a','a','b','c','b','a'), nrow = 2, ncol = 3, byrow = TRUE)
print(M)

# Create a matrix by column.
N <-matrix( c('a','a','b','c','b','a'), nrow = 2, ncol = 3, byrow = FALSE)
print(N)
# Create a numerical matrix.
N <-matrix( c(1:6), nrow = 2, ncol = 3, byrow = TRUE)
print(N)
```

MATRICES

- You can also create a matrix by using column binding and row binding functions:

```
B <- cbind(c(1, 2, 3), c(4, 5, 6));
print(B);
C <- rbind(c(1, 2, 3), c(4, 5, 6));
 print(C);
```

- Define the column and row names.

```
rownames = c("row1", "row2", "row3", "row4")
colnames = c("col1", "col2", "col3")
P <- matrix(c(3:14), nrow = 4, byrow = TRUE, dimnames =
list(rownames, colnames))
print(P)
#Change name of matrix
row.names(P)<-c("a","b","c","d")
row.names(P)<-c("a","b","c","d")
```

ARRAYS

- Arrays can be of any number of dimensions compared to 2D Matrix

 - *array(c(11:14, 21:24, 31:34), dim = c(2, 2, 3))*

- 3D, 4D , 5D example
- Naming of array must meet dimension of array

EXERCISE 2

- Create a 5 by 5 matrix

1	**3**	**5**	**7**	**9**
11	13	15	17	19
21	23	25	27	29
31	33	35	37	39
41	43	45	47	49

- Create the same matrix using cbind or rbind
- Create a 3D array has 3 elements with each has a 3*3 matrix

FACTORS

- Stores the distinct values of the elements in the vector as labels
- For all data types

```
# Create a vector.
apple_colors <-
c('green','green','yellow','red','red','red','green')

# Create a factor object.
factor_apple <- factor(apple_colors)

# Print the factor.
print(factor_apple)
print(nlevels(factor_apple))
```

FACTORS

- Apply the factor function with required order of the level.

- *new_factor_apple <- factor(factor_apple,levels = c("yellow","green","red"))*

- *print(new_factor_apple)*

DATA FRAMES

- Data frames are tabular data objects
- A list of vectors of equal length.

```
A <- data.frame(emp_id=c(1, 2, 3), names=c("John", "James",
                                           "Mary"), salary=c(111.1, 222.2, 333.3));
```

- Create Dataframe by csv

```
animal<-read.csv("AnimalData.csv")
```

OPERATORS

- Arithmetic Operators
- Relational Operators
- Logical Operators
- Assignment Operators
- Miscellaneous Operators(Introduced when needed)

ATHEMATIC OPERATORS

Operators	Meanings	Examples
+	Add	a<-1; c<-5 ;a+c v <- c(2,5.5,6) ; t <- c(8, 3, 4); v+t
-	Minus	c-a ; t-v
*	Multiply	c*a ; t*v
/	Divide	c/a ;t/v
%%	Reminder	c%%a ; t%%v
%/%	Integer quotient	c%/%a ; t%/%v
^	Power	c^a ; t^v

RELATIONAL OPERATORS

Each element of the first vector is compared with the corresponding element of the second vector. The result of comparison is a Boolean value.

Operators	Meanings	Examples
>	Greater	a<-1; c<-5 ;a>c ;c>a v <- c(2,5.5,6,9) t <- c(8,2.5,14,9); v>t
<	Less than	a<c ; t<v
==	Equal to (Not assignment!)	c==a ; t==v
<=	less than or equal to	c<=a ;t<=v
>=	Greater than or equal to	c>=a ; t>=v
!=	Not equal	c!=a ; t!=v

LOGICAL OPERATORS

It is applicable only to vectors of type logical, numeric or complex. All numbers greater than 1 are considered as logical value TRUE.

Operators	Meanings	Examples
&	AND	v <- c(3,1,TRUE,2+3i) t <- c(4,1,FALSE,2+3i) print(v&t)
|	OR	v <- c(3,0,TRUE,2+2i) t <- c(4,0,FALSE,2+3i) print(v|t)
!	NOT	v <- c(3,0,TRUE,2+2i) print(!v)
&&	AND for first element of vector	v <- c(3,0,TRUE,2+2i) t <- c(1,3,TRUE,2+3i) print(v&&t)
!!	NOT for first element of vector	v <- c(0,0,TRUE,2+2i) t <- c(0,3,TRUE,2+3i) print(v||t)

Q&A

THANK YOU!