Lecture #1

A: **RStudio**

**Objectives**

1: Know how to use and navigate RStudio Integrated Development Environment (IDE)

1. Navigate the IDE
2. Learn some commonly used keyboard shortcuts
3. A look at the complete cheat sheep for future reference

2: Why we should know how to use the IDE

B: **R Software**

**Objectives**

1: Use R as a calculator

2: Use Washington post article to describe the following concepts:

1. an observation: the process of watching someone or something, informal or formal, and involves some form of data collection (e.g., a chart showing data collected over a period of time or a chart showing who is raising US-born children)
2. variables: quantity capable of assuming any of a set of values, such as x in the expression x +1 or the assignment y = x

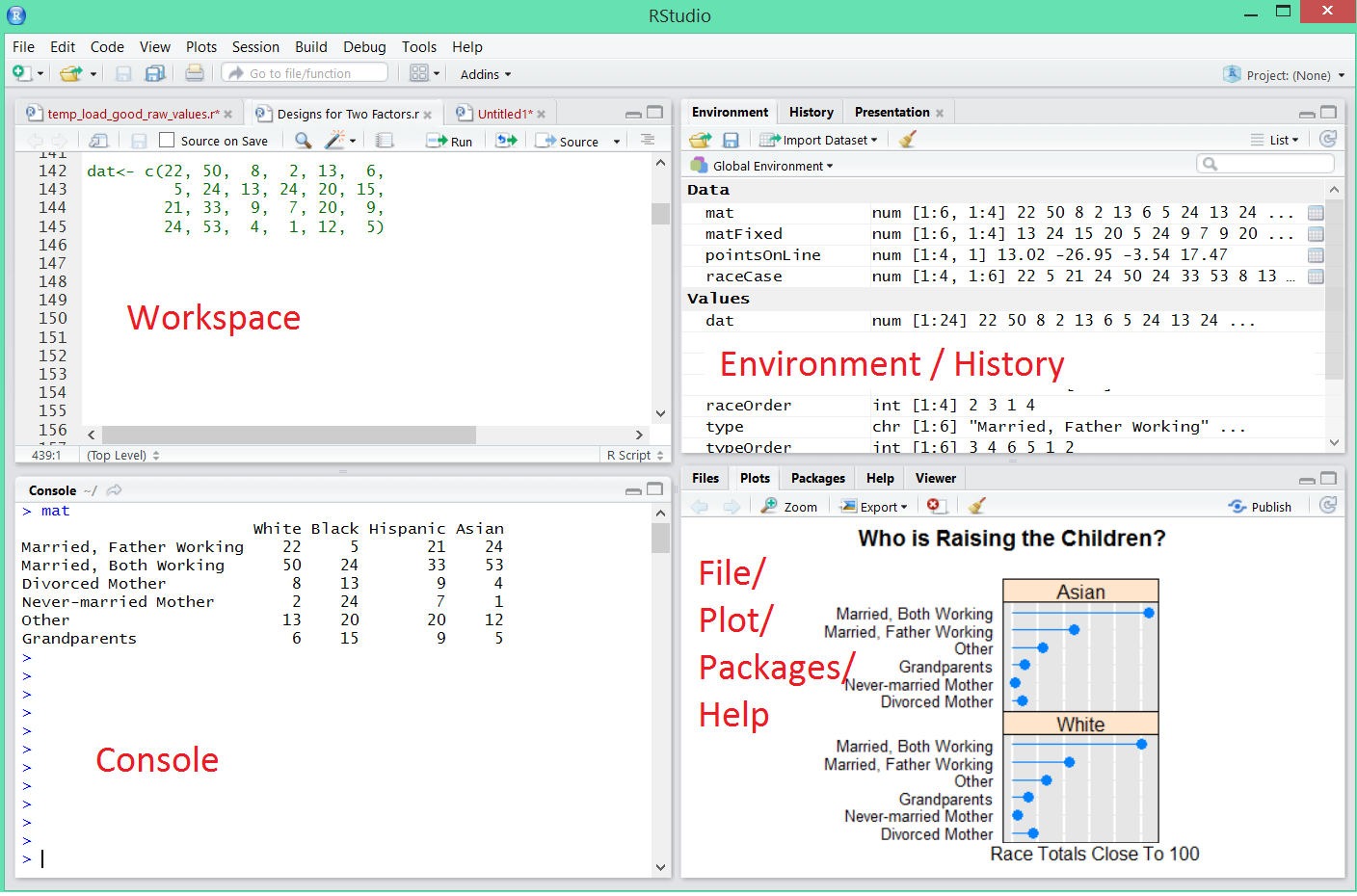
3: R data types: vectors, arrays, lists, matrices, data frame and factors

4: Write simple and save R programs (scripts) to be modified for future use.

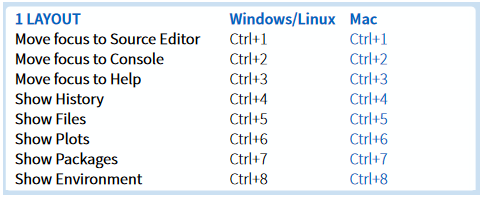
5: In class lab exercise

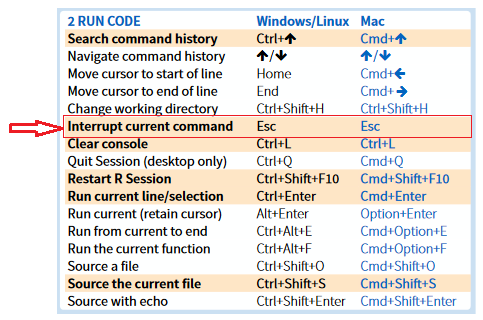
6: Homework 1

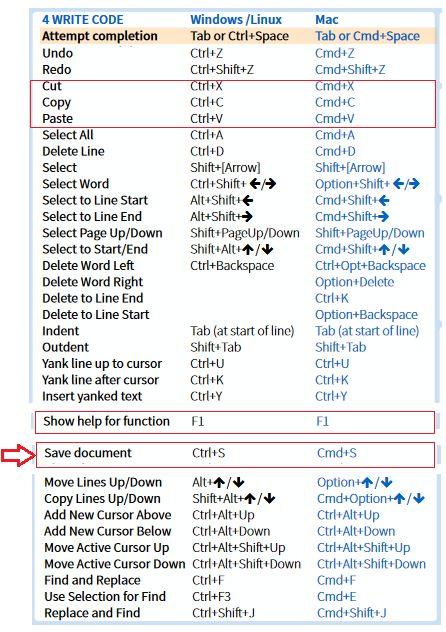
A: **RStudio**



**Keyboard short cuts**





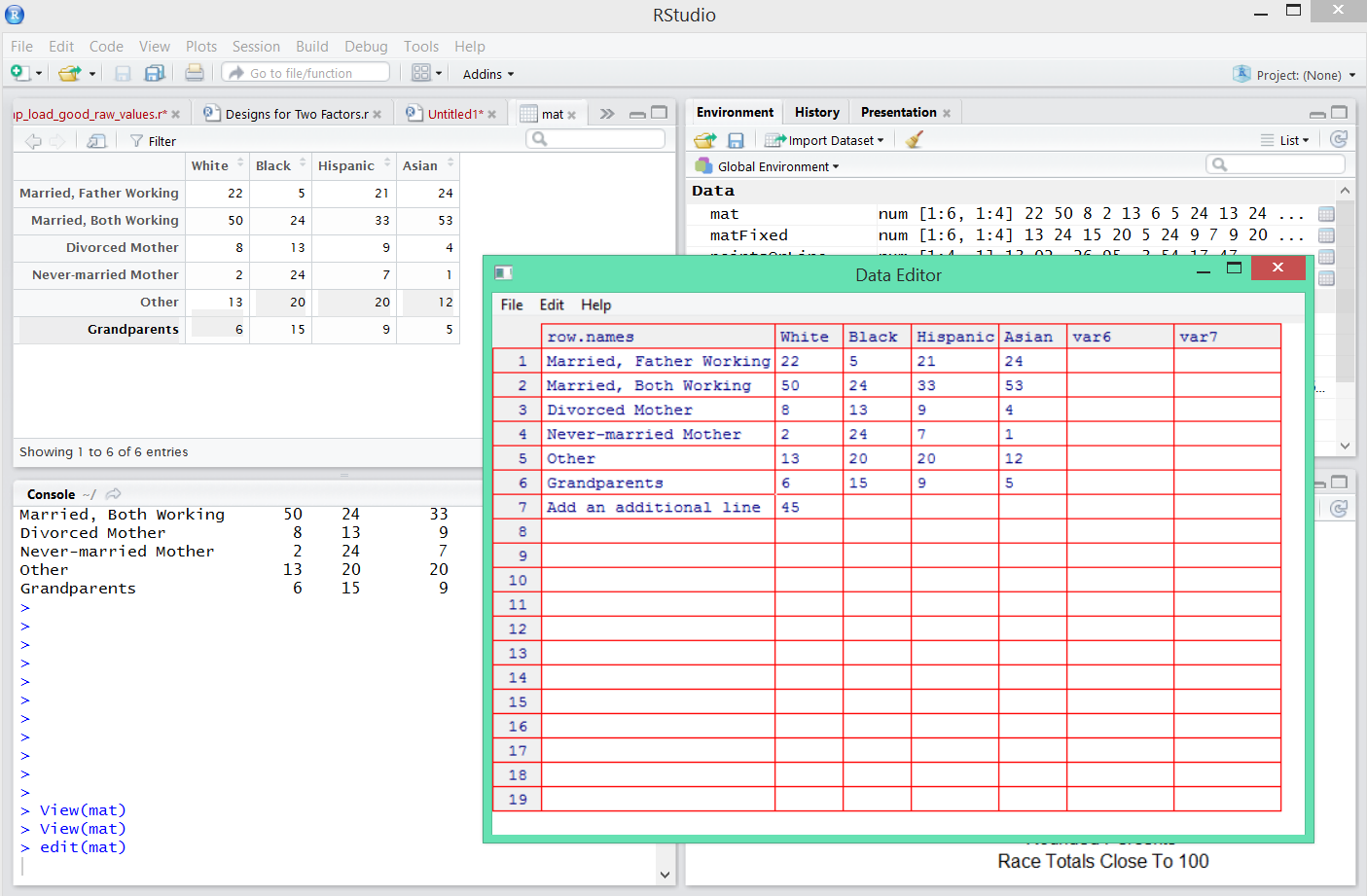


**RStudio IDE cheat sheet**

<https://www.rstudio.com/wp-content/uploads/2016/01/rstudio-IDE-cheatsheet.pdf>

**Why do you have to use RStudio**

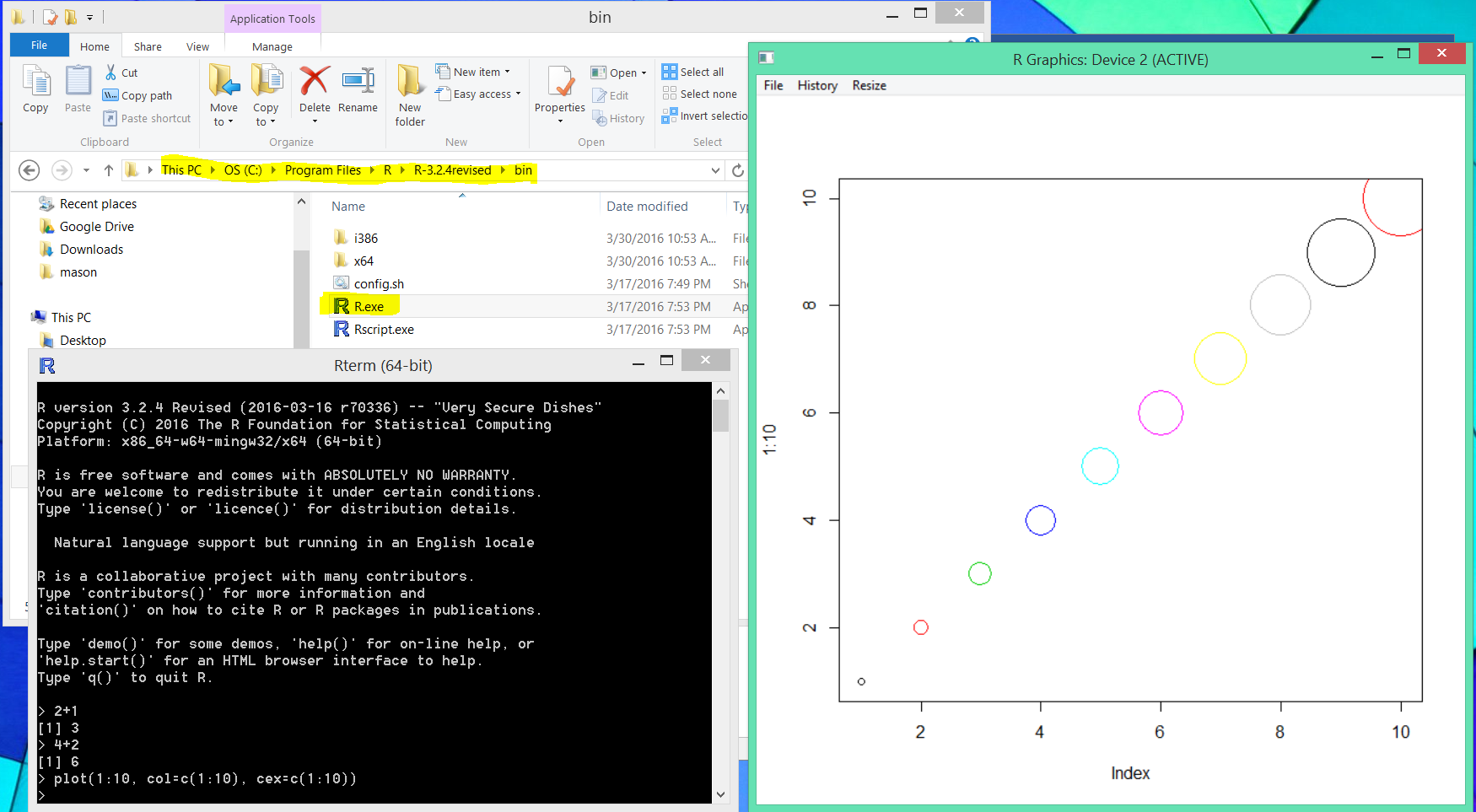
1. A graphical workspace
2. Full-featured text editor
3. Tab-completion of filenames, function names and arguments
4. Variable inspection
5. Features, features, features

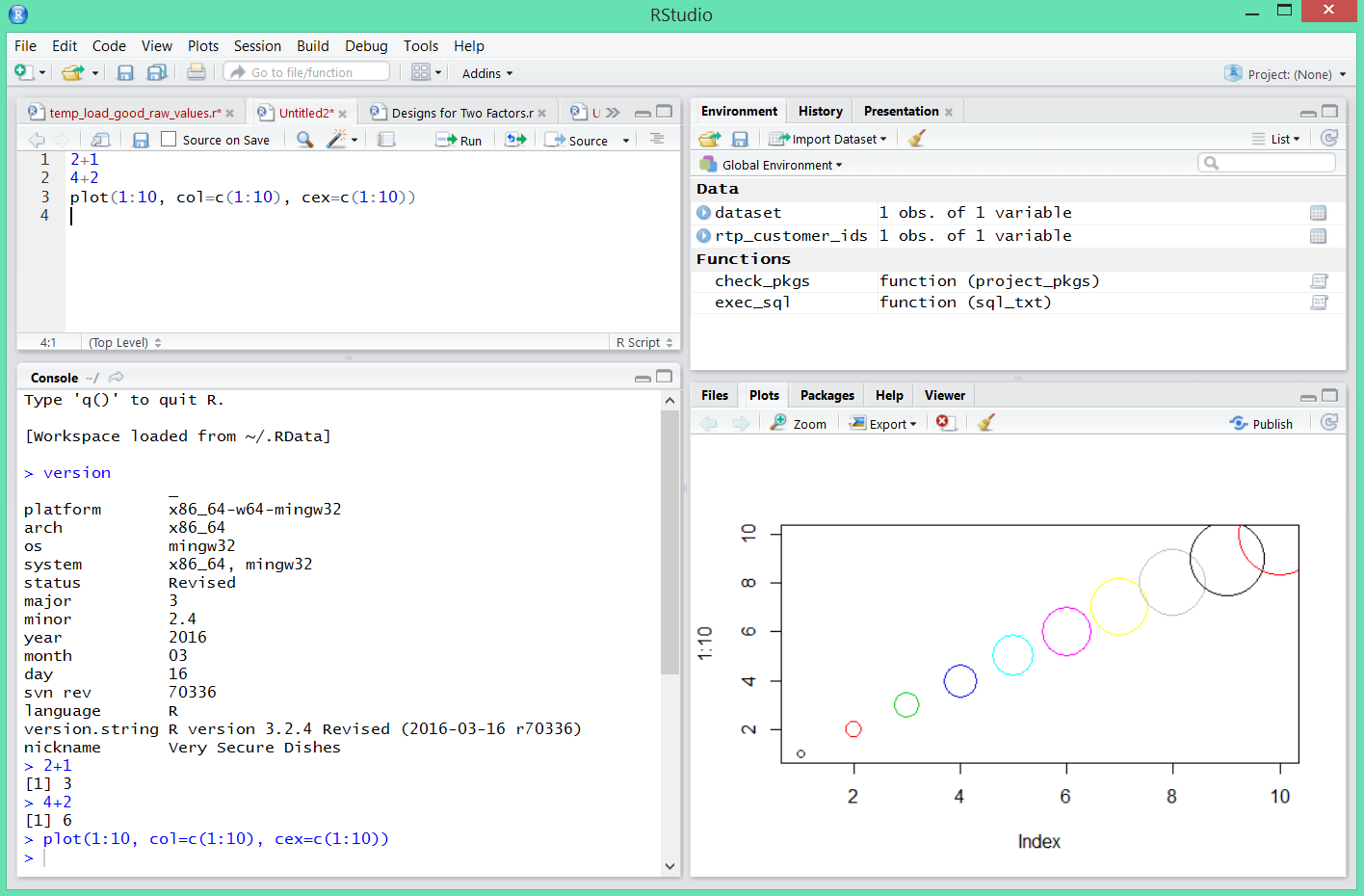


**Lab #1: Take a look at the RStudio IDE cheat sheet and try some of the command not covered**

<https://www.rstudio.com/wp-content/uploads/2016/01/rstudio-IDE-cheatsheet.pdf>

**R Software**

**A: Without R Studio**

B: With RStudio (

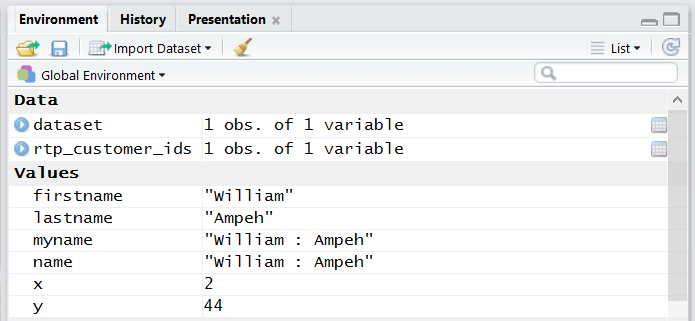
**1: How to use R as a calculator**

**Start RStudio**

* Start RStudio (Windows key, RStudio <Enter>)
* Key in as many mathematical expressions as can and note the results. You may use the following example:
  + 1 + 1
  + 2 + 5
  + sqrt(25)
  + x = 2
  + x + 3
  + factorial(6)
  + 6\*5\*4\*3\*2\*1
  + 45 + <Press the Enter key>

10

* + 5^3
  + y = 34 + 10
  + firstname = 'William'
  + lastname = 'Ampeh'
  + myname = paste(firstname, ' : ' , lastname)
  + myname
* Observe the variables x and y from the **Environment** window
* Also take note of the data type of ***x***, ***y*** and ***myname***



**Note**: Your window entries may differ from the one show above

**2: Use Washington post article to describe the following concepts of observations and variables**

****

**Class discussion (5 minutes)**:

List what you thing will be the observation and the variables in the above picture

Observation:

Variables:

Values:

The two factors in the above plot are:

* race with 4 levels and
* family type with 6 levels.

An example of a family type is "Married, both working".

There are 24 values (***percents***) that correspond to the product set of:

* 4 races and
* 6 family types.

The family type (***percents***) are percent of the total for each race.

Note: The percent totals for each race turn out to be 99 or 101 due to rounding to integers.

**3: R data types: vectors, arrays, lists, matrices, data frame and factors**

**A: Vectors**: A vector is the most basic data structure in R. All values are vectors.

* Numeric: Vector of length 1

x = 22

y = 50

z = 3.14

typeof(x)

typeof(y)

lapply(c(value, myPi, pi), typeof ) *#see the class of each of the variables*

* Character: Non-numeric values

string = “FRB 1801K-street”

Logic: True or False

t = 2 <4

typeof(t)

class(t)

* **Factors** Factors (categorical data) and dates are built on top of integers

x <- factor( c("Yes", "No", "No", "Yes", "Yes", "Maybe") )

x

classof(x)

s <- factor( rep(c("Male", "Female"), times=c(3, 4)) )

s

**B: List: Generic vector containing other objects of the same type**

xList <- list(1, 2, 3)

v = c("a","b",1,2,3,TRUE)

Discussion: 1: Display the value of v

2: What is the ***type*** and ***class*** of v?

---

**C: Matrix: A collection of data elements arranged in a 2-D rectangular layout**

The data elements of a matrix must be of the same basic type.

A = matrix(

c(1:12), *# the data elements*

nrow=3, *# number of rows*

ncol=4, *# number of columns*

byrow = TRUE *# fill matrix by rows*

)

A *# print A*

**Another Matrix**

column\_1 <- c(22, 50, 8, 2, 13, 6)

column\_2 <- c( 5, 24, 13, 24, 20, 15)

column\_3 <- c(21, 33, 9, 7, 20, 9)

column\_4 <- c(24, 53, 4, 1, 12, 5)

mat <- cbind(column\_1, column\_2, column\_3, column\_4)

mat

An element at the mth row, nth column of A can be accessed by the expression A[m, n].

mat[3, 1] *# element at 3nd row, 1rd column*

mat[1, ] *# all elements of the first row*

mat[ , 2] *# second column elements*

We can also extract more than one rows or columns at a time.

mat[ ,c(1,4)] *# the 1st and 4rd columns*

Discussion: Create a new matrix using:

1: the first 2 columns of mat

2. last 2 rows of mat

3: the 3rd and 4th column of mat

4: extract the number **53** from the **mat** matrix

5: find the sum of the data values in row 2 of mat

6: find the mean and standard deviation of the data values 2nd column of mat

**D: A data frame a list of vectors of equal length.**

For example, the following variable df is a data frame containing three vectors a, b, c.

a = 1:3 *#same as c(1, 2, 3) or seq(from=1, to=3, by=1)*

b = c("aa", "bb", "cc") *# 3 character sets*

c = c(TRUE, FALSE, TRUE) *# 3 logic values*

mydf = data.frame(a, b, c) *# mydf is a data frame*

head(mydf, n=2)

m.df <- data.frame(column\_1 = c(22, 50, 8, 2, 13, 6),

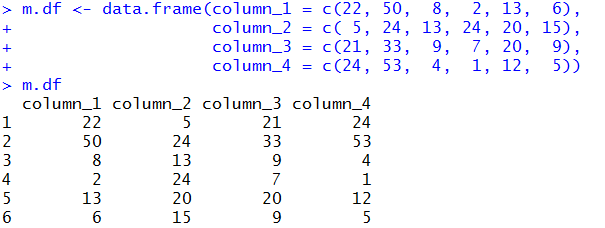
column\_2 = c( 5, 24, 13, 24, 20, 15),

column\_3 = c(21, 33, 9, 7, 20, 9),

column\_4 = c(24, 53, 4, 1, 12, 5))

m.df

**In RStudio**



Discussion: Create a data frame using the data values from the **mat** matrix

1: the first 2 columns of mat

2. last 2 rows of mat

3: the 3rd and 4th column of mat

4: extract the number **53** from the **mat** matrix

5: find the sum of the data values in row 2 of mat

6: find the mean and standard deviation of the data values 2nd column of mat

**Build-in Data Frame**

R has a number of built-in data frames, for example, here is a built-in data frame in R, called mtcars.

head(mtcars)

## mpg cyl disp hp drat wt qsec vs am gear carb

## Mazda RX4 21.0 6 160 110 3.90 2.620 16.46 0 1 4 4

## Mazda RX4 Wag 21.0 6 160 110 3.90 2.875 17.02 0 1 4 4

## Datsun 710 22.8 4 108 93 3.85 2.320 18.61 1 1 4 1

## Hornet 4 Drive 21.4 6 258 110 3.08 3.215 19.44 1 0 3 1

## Hornet Sportabout 18.7 8 360 175 3.15 3.440 17.02 0 0 3 2

## Valiant 18.1 6 225 105 2.76 3.460 20.22 1 0 3 1

**Data frame slicing using Numeric Indexing**

To display the cell value from the first row, second column of mtcars.

mtcars[1, 2]

**Data frame slicing using Name Indexing**

Moreover, we can use the row and column names instead of the numeric coordinates.

mtcars["Mazda RX4", "cyl"]

## [1] 6

**Data frame slicing using Logical Indexing**

finally, we can also retrieve rows with a logical index vector

G = mtcars$gear == 3 *#[,10] gear Number of forward gears*

head( mtcars[G, ] )

## mpg cyl disp hp drat wt qsec vs am gear carb

## Hornet 4 Drive 21.4 6 258.0 110 3.08 3.215 19.44 1 0 3 1

## Hornet Sportabout 18.7 8 360.0 175 3.15 3.440 17.02 0 0 3 2

## Valiant 18.1 6 225.0 105 2.76 3.460 20.22 1 0 3 1

## Duster 360 14.3 8 360.0 245 3.21 3.570 15.84 0 0 3 4

## Merc 450SE 16.4 8 275.8 180 3.07 4.070 17.40 0 0 3 3

## Merc 450SL 17.3 8 275.8 180 3.07 3.730 17.60 0 0 3 3

R Variable names

A **syntactically valid name** consists of letters, numbers and the dot or underline characters and starts with a letter or the dot not followed by a number. Names such as “.2way” are not valid, and neither are the reserved words.

R’s **make.names()** function can be used to check and/or create valid variable names

*# ?make.names*

make.names(c(".2pi"))

## [1] "X.2pi"

Discussion: Try forming as many variable names as you can using R’s ***make.name***s function

The Mosaic package

The mosaic package is designed to help simplify the interface for R users, while allowing them to undertake sophisticated statistical analyses.

*#install.packages("mosaic")*

**library**(mosaic)

( x <- 1:10 )

( xmean <- mean(x) ) #new (x <- 1) # assign and display

( xsd <- sd(x) )

# display functions in this package

ls("package:mosaic")

#

#help(mosaic) #Homework Q5.b

# Use the summary function

summary()

**End of session**