NIBM | POWERING GREAT MINDS

DIGITAL CAMPUS

Introduction to R

By T.Balakumar(MSc,BSc)
Lecturer at NIBM

NIBM WORLD WIDE

POWERUP
WITH THE CHAANGE

Outline

- R Environment
- Getting Started with R and Rstudio
- Getting Help with Features
- R Scripts
- R markdown
- Installing Packages
- Where Does Your Analysis Live
- Data import
- **Basic Syntax**
- **R as a Calculator**
- **Vectors and Assignment**
- **Regular Sequences**
- **rep() fun**

R Environment

R is an integrated suite of software which facilitates for **data manipulation, calculation graphical display and analysis**.

R is an effective data handling and **storage facility**.

It is a **large**, **coherent**, **integrated collection** of intermediate tools for data analysis.

A statistician would find R easy to use for data analysis since **R has built**-in commands for most common statistical procedures.

A software developer would find R convenient for developing a piece of software using **R scripts** and interfacing facilities of R.

The most important feature of R is that it is **open** to everybody.(Free and Open Source software).

Getting Started

R is both a programming language and software environment for statistical computing, which is free and open-source. To get started, you will need to install two pieces of software:

- R, the actual programming language.

Chose your operating system, and select the most recent version, R-4.0.1 for Windows (32/64 bit)

- R studio, an excellent IED for working with R.

Note, you must have R installed to use RStudio. RStudio is simply an interface used to interact with R.

R Installation

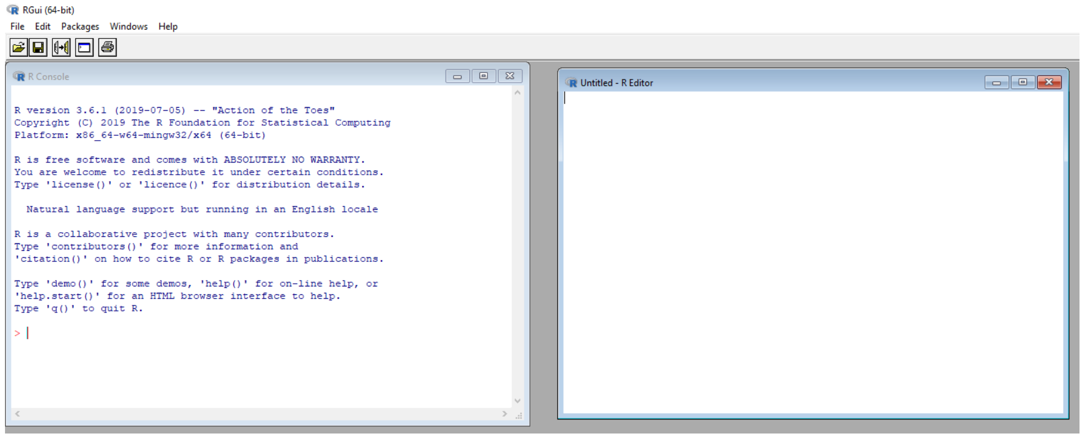
Go to https://cran.r-project.org/ and in the **Getting Started** box, click on **download R**.

Current version is Download R 4.1.2 for Windows (86 megabytes, 32/64 bit)

Once R is installed you will come across this view.

When starting R, the main window with a sub window (R Console) will appear.

In the Console window the cursor is waiting for you to type in some R commands.

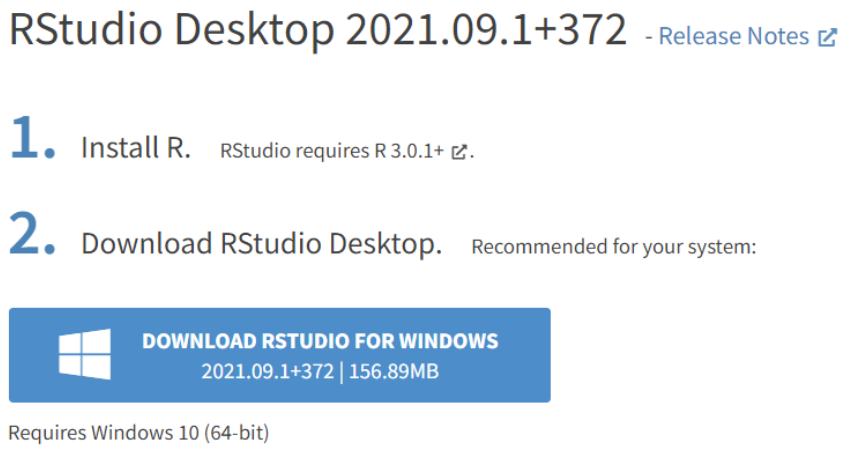


```
R version 3.6.1 (2019-07-05) -- "Action of the Toes"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

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.

  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.

> |
```

R-studio installation

There is also an integrated development environment available for R that is built by RStudio. It really likes this IDE since it has a nice editor with syntax highlighting, there is an R object viewer, and there are a number of other nice features that are integrated.

https://rstudio.com/products/rstudio/download/



RStudio Desktop 2021.09.1+372 - Release Notes

1. Install R. RStudio requires R 3.0.1+.
2. Download RStudio Desktop. Recommended for your system:

Requires Windows 10 (64-bit)

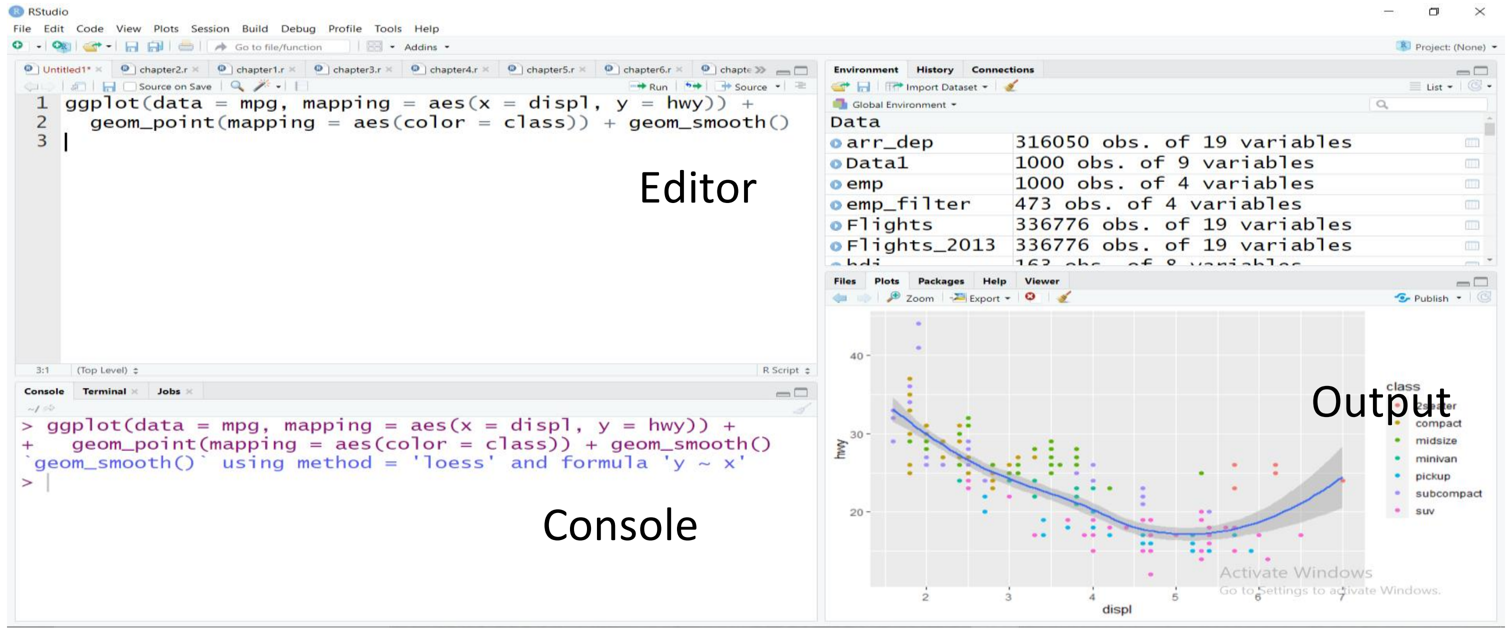
Getting Help with Features

To get documentation about a function in R, simply put a question mark in front of the function name and RStudio will display the documentation, for example:

- ?log
- ?sin
- ?paste
- ?lm

Workflow: Scripts

To give yourself more room to work, it’s a great idea to use the script editor. Open it up either by clicking the File menu and selecting New File, then R script, or using the keyboard shortcut Cmd/Ctrl-Shift-N. Now you’ll see four panes:



```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point(mapping = aes(color = class)) + geom_smooth()
```

```
> ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
+   geom_point(mapping = aes(color = class)) + geom_smooth()
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
>
```

Data

arr_dep	316050 obs. of 19 variables
Data1	1000 obs. of 9 variables
emp	1000 obs. of 4 variables
emp_filter	473 obs. of 4 variables
Flights	336776 obs. of 19 variables
Flights_2013	336776 obs. of 19 variables
hdi	163 obs. of 8 variables

Installing Packages in R

- To install a package, use the install.packages() function.
- Once a package is installed, it must be loaded into your current R session before being used.
- Once you close R, all the packages are closed. The next time you open R, you do not have to install the package again, but you do have to load any packages

Where Does Your Analysis Live?

R has a powerful notion of the working directory. This is where R looks for files that you ask it to load, and where it will put any files that you ask it to save.

- You can print this out in R code by running **getwd()**
- You can also set the working directory in R: **setwd(“path")**

R markdown

- R Markdown provides a unified authoring framework for data science, combining your code, its results, and your prose commentary
- File-New file – R markdown
- To produce a complete report containing all text, code, and results, click “Knit” or press Cmd/Ctrl-Shift-K. This will display the report in the viewer pane, and create a self-contained HTML file that you can share with others.

Data import - readr’s functions

Assume that the files are in your current working directory.

read_csv() reads comma-delimited files

- read.csv(“file.csv”, header = T)

read_delim() reads in files with any delimiter

- read.delim(“file.txt”, header = T)

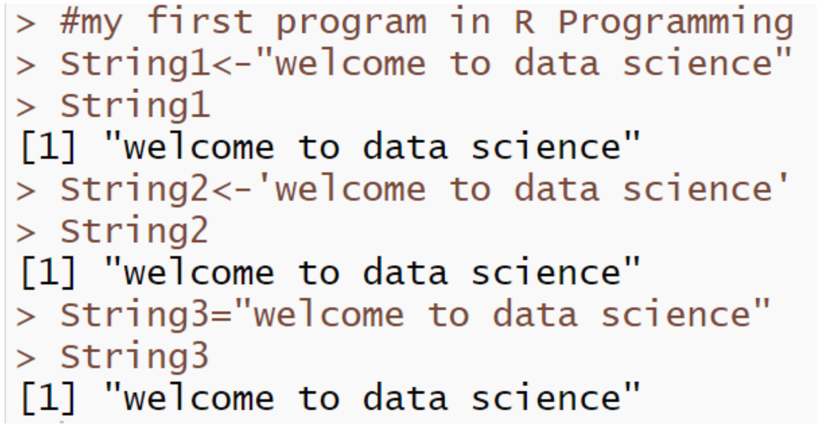
File.choose

read.csv(file.choose(), header = T)

Normally R would like us to specify the path to find but the handy command to know in R is the “file.choose” rather than specifying the path to find this file.

Basic Syntax

Type the following in your R Studio



```
> #my first program in R Programming
> String1<-"welcome to data science"
> String1
[1] "welcome to data science"
> String2<-'welcome to data science'
> String2
[1] "welcome to data science"
> String3="welcome to data science"
> String3
[1] "welcome to data science"
```

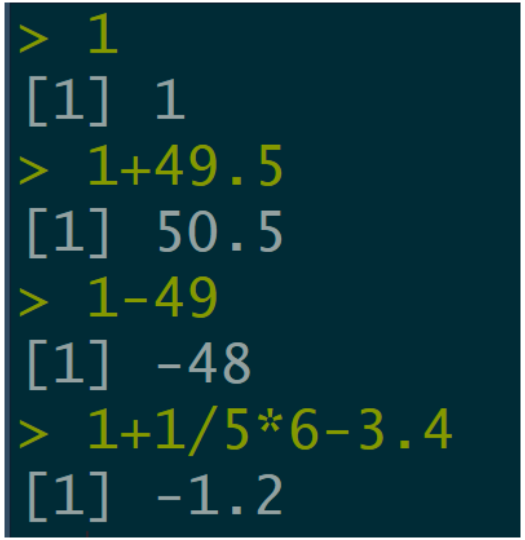
Variable assignment is done using the assignment operator <- or =.

Comments are ignored by the interpreter while executing your actual program.

A single comment is written using #.

Basic calculations

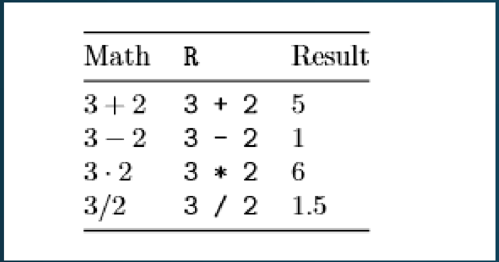
R as a Calculator



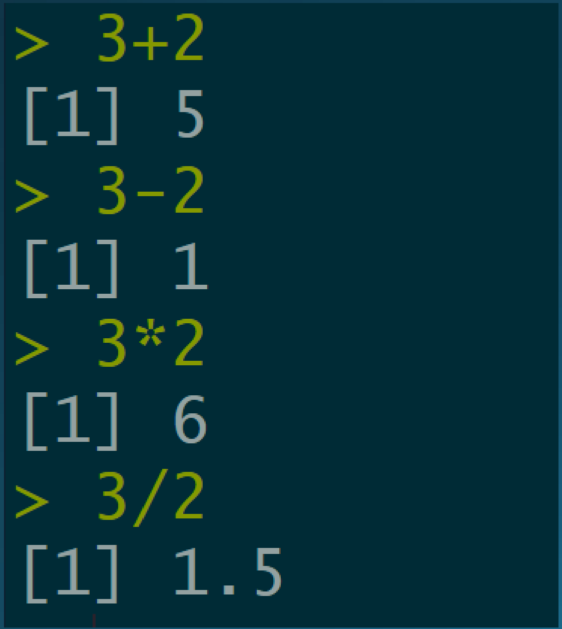
```
> 1
[1] 1
> 1+49.5
[1] 50.5
> 1-49
[1] -48
> 1+1/5*6-3.4
[1] -1.2
```

- The simplest thing that R can do is evaluate arithmetic expressions.
- R can be used as a powerful calculator by simply type your arithmetic expression.
- R will evaluate the expressions and respond with the result.
- R will normally execute your arithmetic expression by evaluating each item from left to right taking in account the precedence of operators.

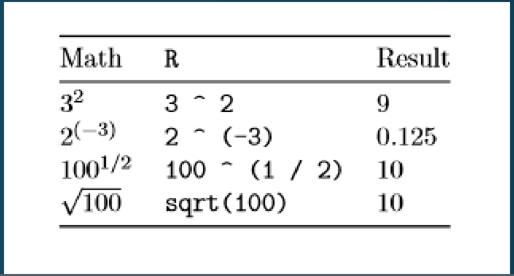
Addition, Subtraction, Multiplication , Division and Exponents



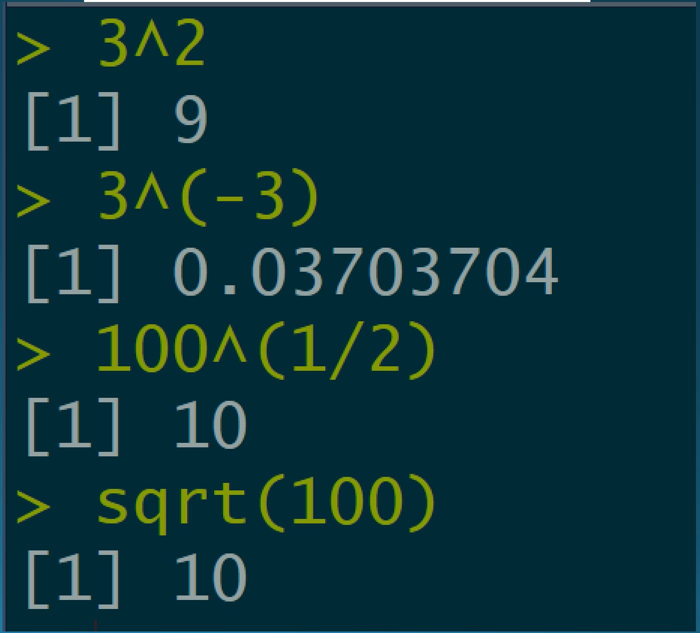
Math	R	Result
$3+2$	`3 + 2`	5
$3-2$	`3 - 2`	1
$3\cdot 2$	`3 * 2`	6
$3/2$	`3 / 2`	1.5



```
>  3+2
[1] 5
>  3-2
[1] 1
>  3*2
[1] 6
>  3/2
[1] 1.5
```



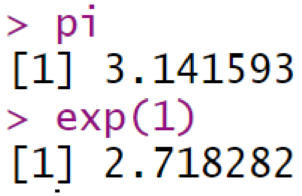
Math	R	Result
3^2	`3 ^ 2`	9
$2^{(-3)}$	`2 ^ (-3)`	0.125
$100^{1/2}$	`100 ^ (1 / 2)`	10
$\sqrt{100}$	`sqrt(100)`	10



```
>  3^2
[1] 9
>  3^(-3)
[1] 0.03703704
>  100^(1/2)
[1] 10
>  sqrt(100)
[1] 10
```

Mathematical Constants

Math	R	Result
π	`pi`	3.1415927
e	`exp(1)`	2.7182818

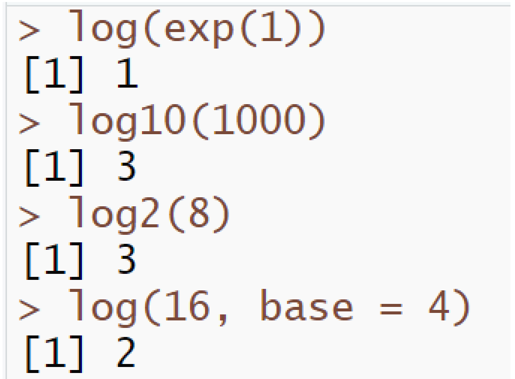


```
> pi
[1] 3.141593
> exp(1)
[1] 2.718282
```

Logarithms

Note that we will use ln and log interchangeably to mean the natural logarithm. There is no ln() in R, instead it uses log() to mean the natural logarithm.

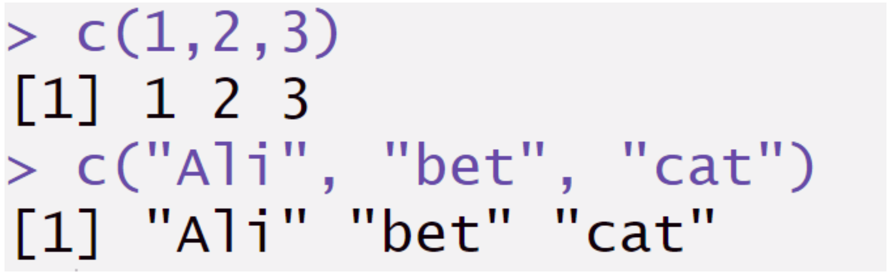
Math	R	Result
$\log(e)$	`log(exp(1))`	1
$\log_{10}(1000)$	`log10(1000)`	3
$\log_2(8)$	`log2(8)`	3
$\log_4(16)$	`log(16, base = 4)`	2



```
> log(exp(1))
[1] 1
> log10(1000)
[1] 3
> log2(8)
[1] 3
> log(16, base = 4)
[1] 2
```

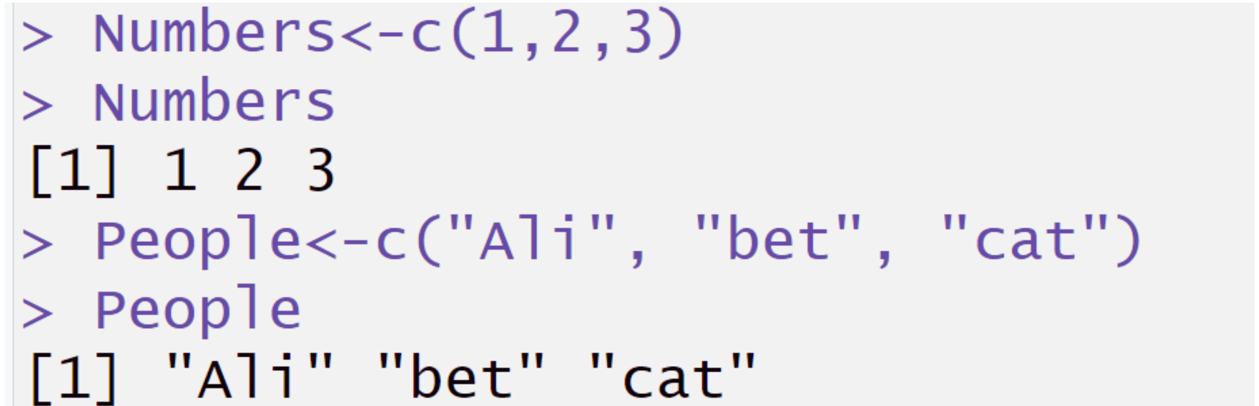
Vectors and Assignment

Possibly the most common way to create a vector in R is using the c() function, which is short for "combine."" As the name suggests, it combines a list of elements separated by commas. c(1, 3, 5)



```
> c(1,2,3)
[1] 1 2 3
> c("Ali", "bet", "cat")
[1] "Ali" "bet" "cat"
```

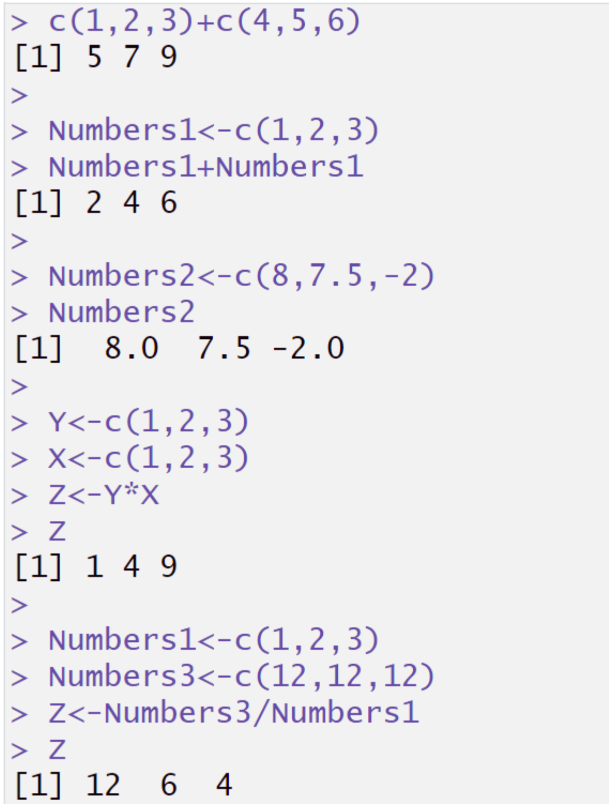
By using the assignment operator to assign the vector to a variable.



```
> Numbers<-c(1,2,3)
> Numbers
[1] 1 2 3
> People<-c("Ali", "bet", "cat")
> People
[1] "Ali" "bet" "cat"
```

Vectors and Assignment

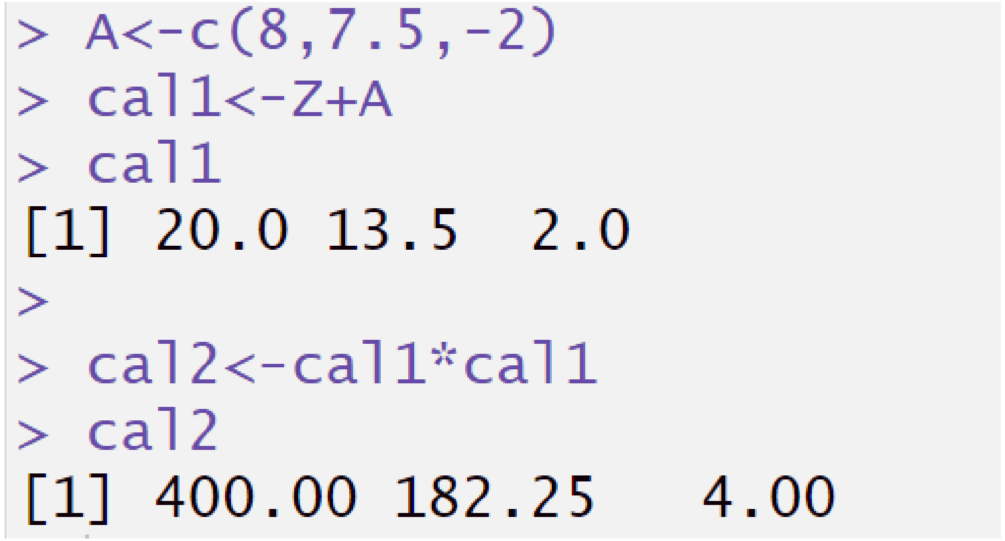
Simple arithmetic operations can be performed with vectors.



```
> c(1,2,3)+c(4,5,6)
[1] 5 7 9
>
> Numbers1<-c(1,2,3)
> Numbers1+Numbers1
[1] 2 4 6
>
> Numbers2<-c(8,7.5,-2)
> Numbers2
[1]  8.0  7.5 -2.0
>
> Y<-c(1,2,3)
> X<-c(1,2,3)
> Z<-Y*X
> Z
[1] 1 4 9
>
> Numbers1<-c(1,2,3)
> Numbers3<-c(12,12,12)
> Z<-Numbers3/Numbers1
> Z
[1] 12  6  4
```

Vectors and Assignment

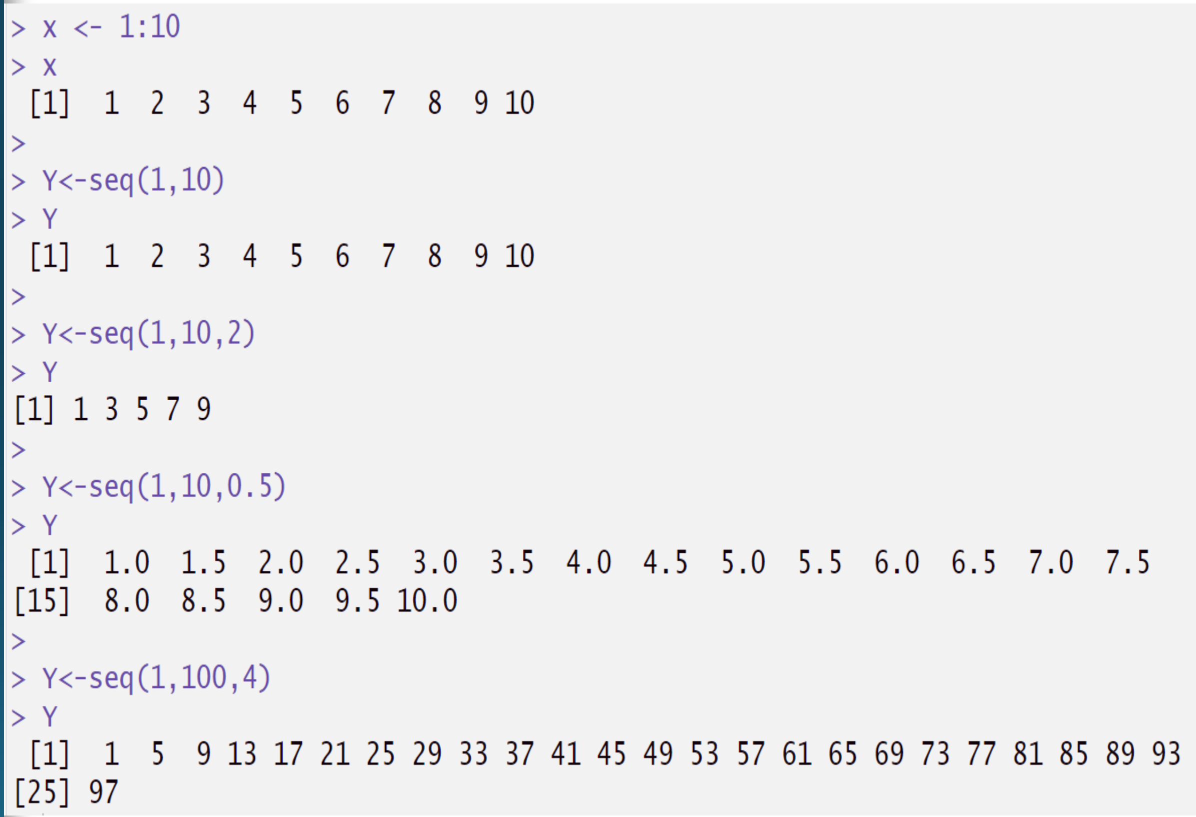
- The outcome of an arithmetic calculation can be given as an identifier for later use.



```
> A<-c(8,7.5,-2)
> cal1<-Z+A
> cal1
[1] 20.0 13.5  2.0
>
> cal2<-cal1*cal1
> cal2
[1] 400.00 182.25   4.00
```

Regular Sequences

- **A regular sequence is a sequence of numbers or characters that follows a fixed pattern.**
- **These are useful for selecting portions of a vector and in generating values for categorical variables.**
- **We can use a number of different methods for generating sequences.**
- **The seq() function allows a greater degree of sophistication in generating sequences.**



```
> x <- 1:10
> x
 [1]  1  2  3  4  5  6  7  8  9 10
>
> Y<-seq(1,10)
> Y
 [1]  1  2  3  4  5  6  7  8  9 10
>
> Y<-seq(1,10,2)
> Y
[1] 1 3 5 7 9
>
> Y<-seq(1,10,0.5)
> Y
 [1]  1.0  1.5  2.0  2.5  3.0  3.5  4.0  4.5  5.0  5.5  6.0  6.5  7.0  7.5
[15]  8.0  8.5  9.0  9.5 10.0
>
> Y<-seq(1,100,4)
> Y
 [1]  1  5  9 13 17 21 25 29 33 37 41 45 49 53 57 61 65 69 73 77 81 85 89 93
[25] 97
```

Regular Sequences…

#If we want to create a sequence that isn't limited to integers and increasing by x at a time, we can use the seq() function.

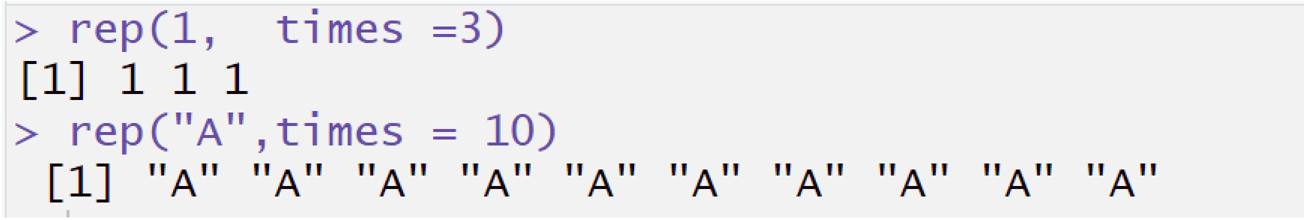
- seq(from = 1.5, to = 4.2, by = 0.1)

#note here that the input labels from, to, and by are optional.

- seq(1.5,4.2,0.1)

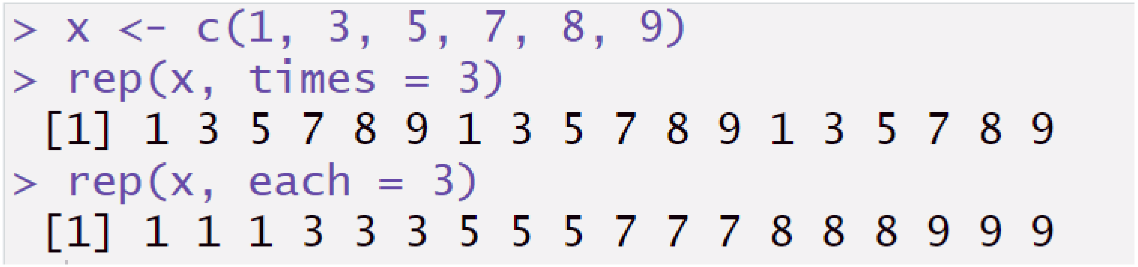
rep()

Another common operation to create a vector is rep(), which can repeat a single value a number of times



```
> rep(1,  times =3)
[1] 1 1 1
> rep("A",times = 10)
 [1] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
```

The rep() function can be used to repeat a vector some number of times.



```
> x <- c(1, 3, 5, 7, 8, 9)
> rep(x, times = 3)
 [1] 1 3 5 7 8 9 1 3 5 7 8 9 1 3 5 7 8 9
> rep(x, each = 3)
 [1] 1 1 1 3 3 3 5 5 5 7 7 7 8 8 8 9 9 9
```