Datatypes and Basic Operations

Sunday, December 20, 2015 7:33 PM

"Take only pictures, leave only footprints."

Everything that exists is an object. Everything that happens is a function call.

Data Types in R

Common data structure in R is the vector. Vectors come in two different flavors: atomic vectors and lists. An atomic vector contains exactly one data type, whereas a list may contain multiple data types.

Atomic Class objects

- Character
- Numeric(Real Numbers) -> 1L. Inf, -Inf and NaN is also number
- Integer -> Array
- Complex -> a+ib
- Logical -> TRUE,FALSE,NA

Basic Object

Vector --> Same class objects

List --> Can have any class

Class() will return the class of the object

Each object have an attribute

- Name, dimnames
- dimensions
- class
- length
- other user-defined attributes/metadata

Commands are called expressions. <- is used for assignment

: operator is used to create from 0 to 20

```
x<- 1:20
```

c is used to create vectors. c can be remembered as concatenate.

Using as.Character can convert to integer.

When mixing different class, least common denominator. Coercion occurs.

Character > Complex > Numeric > Integer > logical

Matrix

```
m<- matrix(nrow=2,ncol=3)
```

attribute(m) returns dimensions.

m<-1:10, then dim(m) <- c(2,5) : Creates a vector

cbind and rbind

```
x<-1:3
y<-10:12
```

cbind(x,y) and rbind(x,y)

Essentially cbind and rbind tells the order of a matrix creating. when rbind, vectors are written in row wise, while cbind writes in column wise.

List

can be created by command list

creates a collection of values with its native form

Factor

Integer vector with a label so that the values will be self-describing.

```
x<- factor(c("yes","no","yes","no","yes","no"), Levels=c("yes","no"))
```

table(x) returns number of levels in x
unclass(x) will strips out class

Missing Values

- is.na() is used if there is an Na
- is.nan() for NaN
- Na values can have class also, so there are integer NA and character NA
- Na.rm()

Data Frames

- For tabular data
- Its a special type of list but every coloum has value and can have different class
- It can be created by read.table() or read.csv()
- can be converted to matrix by data.matrix()
- x<- data.frame(foo=1:5,bar=c(T,T,F,F))
- row.names() will give the name of the rows. Like row number in excel
- nrow() and ncol() for row and coloumn

Names

- Assigns names to values
  names(x) <- c("foo","bar","norf")
- Also list also can have names
  x<-list(a=1,b=2)
- For matrix, dimnames(matrix)<-list(c("a","b"),c("c","d")

Reading and Writing Data

Tabular	read.data,read.csv	write.table
read lines	readlines	writelines
Reading R code file	source	dump
For reading R code files	dget	dput
Saved Workspace	load	save
For reading single R objects in Binary Form	unserialize	serialize

Read.table

The `read.table` function is one of the most commonly used functions for reading data. It has a few important arguments:

- `file`, the name of a file, or a connection
- `header`, logical indicating if the file has a header line
- `sep`, a string indicating how the columns are separated
- `colClasses`, a character vector indicating the class of each column in the dataset
- `nrows`, the number of rows in the dataset
- `comment.char`, a character string indicating the comment character
- `skip`, the number of lines to skip from the beginning
- `stringsAsFactors`, should character variables be coded as factors?

Look Read help file
comment.char='' if no comment in the file
dput and dump can be helpful incase the data in R needs to be stored. MetaData with the data

will be stored.

Reading in Larger Datasets with read.table

- Use the `colClasses` argument. Specifying this option instead of using the default can make 'read.table' run MUCH faster, often twice as fast. In order to use this option, you have to know the class of each column in your data frame. If all of the columns are "numeric", for example, then you can just set `colClasses = "numeric"`. A quick an dirty way to figure out the classes of each column is the following:

```
initial <- read.table("datatable.txt", nrows = 100)
classes <- sapply(initial, class)
tabAll <- read.table("datatable.txt",
                     colClasses = classes)
```

Size of data: Row*Column*size in bytes of object/2^20 MB

Interfaces with Outside world

file = Connection to a file
gzfile = Compressed gzip
bzfile = Compressed bzip2 algorithm
url = webpage

Sub-setting

Extract subsets of R objects

- [--> Returns the object of same type. Canbe used to select more than one element
  - x[1:4]
  - x[x>"a"] returns all the elements greater than a. but u<-x>"a" will be logical output in u.
  - x[1,] will return row1, x[,1] will return column 1.
  - > s<-z[1,2,drop=TRUE]
    > typeof(s) "double"
    > s<-z[1,2,drop=FALSE]
    > typeof(s) "list"
- [[--> For list or data frame. not necessarily be a list or data frame.
  - norf<-list(foo=1:4,bar=0.6)
  - norf[["bar"]]
  - norf[[c(1:3)]]
  - matrix[1,] row
  - -neagtive index can give all the values except that index
- $ --> To extract elements from a list by name.
  - norf$foo or norf$foo[1]

Partial Matching and NA Values

- a[["a",exact=FALSE) gives the closest.
- bad<-is.na(x) and x[!bad] can remove bad values.
- good<-complete.cases(h) gives a vector without NA values
  - Then h[good,][1:6,] will return a vector that has no NA values

IF-ELSE

if-else if-else
y<- if(x>3){3} else{7}

Functions

- By setting value in argument, creates the default value for the argument in case the argument is

missing.
- Also with argument name and partial argument name,it doesn't matter in which order we call the function arguments.
- using eclipses (...), it is possible to combine the arguments
- Then inside the function, unpack the arguments using the below commands

```
args <- list(...)
arg1<-args[["arg1"]]
```

lapply() - List apply

lapply() function takes a list as input, applies a function to each element of the list, then returns a list of the same length as the original one.

```
lapply(variable,function)
lapply(unique_vals, function(elem) elem[2])
```

sapply() - simplify2array lapply()

Same as lapply but a vector will be the output instead of list

vapply()

sapply() tries to 'guess' the correct format of the result, vapply() allows you to specify it explicitly.

If the result doesn't match the format you specify, vapply() will throw an error, causing the operation to stop.

This can prevent significant problems in your code that might be caused by getting unexpected return values from sapply()

vapply(flags, unique,numeric(1)) is looking for an output of numeric in size 1 but the return is different.

tapply()

split your data up into groups based on the value of some variable, then apply a function to the members of each group.

tapply(flags$animate, flags$landmass, mean): Take mean of animate after grouping with landmass.

tapply(flags$population,flags$red, summary) : Similar

Looking at a Data

```
> class(ab) : [1] "character"
> table(ab) : ab
      C D E G H J K L N O P Q R T U V W X Z
      2 1 1 1 1 1 2 1 1 2 1 1 2 2 1 1 1 1 3
> summary(ab)
          Length     Class      Mode
            26 character character
> range(ab)
      [1] "C" "Z"
> str(ab)
       chr [1:26] "R" "C" "O" "J" "Z" "P" "Z" "Z" "O" "T" "R" "D" "X" "K" "N" "C" "L" "Q" "U" ..
>names() and object.size()
```

Simulation

Sample() - Creates a random sample of data. Replace argument is for considering the values got in previous sample in this sample.

LETTERS[1:26] - Variable with letters

Binomial Distribution: look google

```
rbinom(1,size = 100,prob=0.5)
```

Normal Distribution: Mean 0 and standard deviation 1

```
rnorm(10)
```

Poison Distribution: Goog Central Limit Theorem

```
replicate(100, rpois(5, 10))
```

hist: Histogram

Also rexp(),ci-squared(rchisq()),gamma(rgamma())

Cheat Sheet

Sunday, December 20, 2015 7:36 PM

Commands	Job	Remarks
ls()	list the variables in the workspace	
list.files()	List the files in the directory	
getwd()	Get working directory	
args(fun_name)	Get the arguments in a function	
setwd(dir)	Set the path as working directory	
file.exists()	Check a file exist	
file.info()	Info about a file	Using $ can get individual value
file.copy,delete,	File and directory operations	dir.create(file.path("testdir2","testdir3"),recursive = TRUE)
unlink(dir)	Delete a directory	
seq(from,to,by=)	Creating a sequence	
seq_along()	For sequence with a length of another vector	
rep(0,times=40) rep(c(0,1,2),each=40)	vector that is 40 0s If each=40, it will be 0,0,0...,111..,222...	
paste(Variable,collape=' ')	For character vector to print the elements	paste(LETTERS,1:26,sep='-')
rnorm()	creates normal distribution of a value	
sample	creating a given number of sample from a given values(can be combined using c())	
identical()	Checks 2 vectors are identical	
isTRUE	To check a logical operation is True	
which()	takes a logical vector as an argument and returns the indices of the vector that are TRUE	
any,all()	checks if any or all condition of the logical is	any(ints<0)

	true	
sys.Date()	Date	
range()	Gives the matrix of min and max of a variable or a list	
summary()	Gives out useful summary details of a given objects	
table()	Will create the summary in a column in a data frame	
object.size()	returns the size of the object	
names()	Returns the names of the columns	
head,tail	First and last rows of data frame	
str()	Structure.combines many of the features of the other functions you've already seen, all in a concise and readable format	
hist	Histogram	
unclass()	To remove a class to see what internally it looks like	
as.POSIXlt(Sys.time())	To save as posix clock	
weekdays(),months(),quarters(),	Months	
strptime()	To convert the time to understand	
difftime	Time difference, unit can be days.. etc	
plot	Ploting a value	

R experiments

Sunday, December 20, 2015 7:38 PM

1) To find the row with any condition satisfied:
Ceo[which(Ceo$Worker_Sal==max(Ceo$Worker_Sal)),]
2) Creating a column in a data frame based on a calculation :
Ceo[,"Rev_Ratio"]<-c(as.integer(Ceo$Revenue/Ceo$Salary*100000))
3) To print last 5 rows with the given condition satisfied:
Ceo[tail(order(Ceo$Revenue),5),]
4) To have control over the decimal of the calculated value:
signif(Ceo$Revenue/Ceo$Salary*1000000,digits=2)
5) To print the data in a definte criteria:
Ceo[tail(order((Ceo$Revenue<30),decreasing = FALSE),10),]
Assign this to a temp variable and then sort that with required column.
or To sort and order a subset: Com[order(Com$f_RatRevToNoOfEmp),[sort(Com $f_RatRevToNoOfEmp)<10,]
6) To do a math or arithmetic of a subset :
sum(Com[Com$c_Category=='Oil and gas',]$d_Revenue)
7) vapply(flags,unique,numeric(1))
8) tapply(flags$population, flags$red, summary)
9) Regular expression to find something:
grep(".*Mean.*|.*Std.*", names(completeData), ignore.case=TRUE)
10) To factor a column based on a vector:
extractedData$Activity <- as.character(extractedData$Activity)
for (i in 1:6){ extractedData$Activity[extractedData$Activity == i] <-
as.character(activityLabels[i,2]) }
11) To find and replace something
names(extractedData)<-gsub("Acc", "Accelerometer", names(extractedData))
12) When plot, using las=2 parameter can invert the axis labels by 90 Degree
13) When using data table, View can give the complete data of variable
14) Ploting to a PDF
t(RunData$Test_Time,RunData$Delta,pch=16)
> pdf("CurrentDelta.pdf", width=40, height=15)
> plot(RunData$Test_Time,RunData$Delta,pch=16,xlab="Time",ylab="Current Delta")
15)