R programming : .

- Knowhow much memory you would need.
- 1.5 mil +120 columns *8bytes /numeric * rule of thumb, twice of this
- $dumping and $dputing - metadata will contain the class, textual data
- textual formats are editable and longer lived, possible to recover it .
- downside- space inefficient
- $dput(y) - output the result constructs a result , with all the metadata. dput(y, file = 'y.r") put metadata into a file
- $dget(y.r) - use the metadata to write back the code in R.
- dump(c(x,y), file=Data.R)
- rm(x,y)
- source)data.R) - to read back into source file

Interface with outside world
file -
gzfile -compressed files
bzfile- compressed files
url

$str(file)
description - name
open - r = read only. w = read and write. -appending.

$con <- file(foo.txt, r)
$data <- read.csv(con) /// x <- readLines(con,10)
$close(con)
$writeLines can be used to write lines.
$con <- url("",,,. "r") ,,, then read lines , and read the html

Subsetting: [- return an object of the same class as the original, can be used to select rmore than one
*** Neumeric index or logical index

[[- list or element, only one single element can be pulled out
$ - used to extract element of list or data frame by NAME.

$x[]
$x[1:4] , subsetting using a numeric index
$x[x >"a"]— subsetting can be done by numeric way as well .
$u <- x > "a" — logical list

Subsetting lists:
$x[[1]] - get a list
$ x$bar - get numbers associated with “bar” same as x[[“bar”]]

$x[c(1,3)] — get multiple elements of a list ,, cannot extract multiple using double bar.
x[[c[1,3)]] - get list element from within another list that exists inside.

Subsetting Matrices:
$- x[1,] first row [1,2] , row column ,,
When matrix is subsetted out if only a single element is asked for, just get
$ x[1,2,drop = False] — subset out

Subsetting Partial Matching:
works with double bracket and $
$ looks for a name in the element with name a
double fracker expects name to be an exact match,
x[[“a” , exact= FALSE]] partially match to find a possible match

Removing NA values:
$bad <- is.na(x) — get which elements are NA
$x[!bad] — get the elements that are not NA.
$ good <- complete.cases(x,y) — get a vector which positions have both elements not missing.
 $x[good] — use to get something and subset out without changing the entire table .

Vectorized operations:
Adds, subtract , corresponding elements together. user <> to get logical symbols for matrix multiply using %*%

y <- matrix(rep(10,4), 2,2) , make y a matrix with all 10s

FUNCTIONS :
can be nested and passed as arguments
formals argument - return sth

& = evaluate all vectors in the c(TRUE , FALSE, TRUE, TRUE)

&& = only evaluate first member of the thing.

Same with OR | , || - two of them.
all and operators are evaluated before OR operations — order of evaluation.

isTRUE(6>4) - takes in an argument and returns true if whats inside the bracket is true

$identical() - Returns true if both are true.

$xor(a, b) - Takes in two arguments, one argument evaluates — If one argument evaluates to TRUE and one argument evaluates to
| FALSE, then this function will return TRUE, otherwise it will return FALSE

 which(ints>7) - returns all those that are true.
$any() - return true if any of them are true.
$all() - true only if all of them are true.

“To understand computations in R, two slogans are helpful: 1. Everything that exists is an object. 2. Everything that happens is a function call.”

- just type function name to see source code for function

— we can make an generate anonymous functions within evaluate(function(x) {x+1}, 6)
evaluate(function(x){x[length(x)]},c(8, 4, 0)) — use length to return last element .

- ... - can be unpacked using args <- list(...) , then use the list to get names of different variables back.
- Binary operators can be created in R . — %[whatever]%

```
"%mult_add_one%" <- function(left, right){ # Notice the quotation marks!
#   left * right + 1
# }
#
```

unclass(d1)- use to check what the value looks like internally when stored in R.

use POSIXlt to store objects to get the min, hour, mon , year,

#t2$min - get min

 str(unclass(t2))— get a more modified compact list.
weekdays(), months(), and
| quarters() — return quarter of the year. **
difftime(Sys.time(), t1, units = 'days') — get difference in different units

anonymous functions ()

LOOP functions

apply() - used to apply to everything in a matrix . not faster than loop apply(x, 2(margin, preserve columns , collapse rows), mean)

some optimized functions:
rowSums: apply(x,1,sum)
rowMeans:'' mean
colSums :apply(x,2,sum)
colMeans : '' mean

lapply() - have a list of objects and want to use that on all — Use in conjunction with SPLIT. lapply(seq, runif, max= , min =) — if you want to specify default values.
$ sapply()
$vapply() and tapply()
$ viewinfo() - get information on loaded dataset.
$ cls_list <- lapply(flags,class) - to apply the function , list apply = apply
as.character(cls_list) - to show as character vector
sapply - simplify apply
flags[, 11:17] - get all rows but only columns 11:17
$sapply(flag_colors, sum, na.rm =TRUE) =— Better and shows in good format. - returns a matrix if multiple one is returned

$lapply(unique_vals, function(elem) elem[2]) — to define functions , return second element of unique_vals.
$vapply() allows you to specify it explicitly . — will give error otherwise.
vapply(flags, unique, numeric(1)) — useful when not working while viewing output at the same time.

$tapply() apply a function to all non empty groups table apply.

$table(flags$landmass) - look at how many fall into each category. ** group by category and count
$tapply(flags$animate, flags$landmass, mean, SIMPLIFY = TRUE) - apply mean to animate separately within each of the six land masses.
$mapply(, Moreargs - , SIMPLiFY =) - multivariate version of lapply.. for two or more different data , take multiple args ,, ** apply a function in parallel over multiple sets of argument. mapply(rep, 1:4 , 4:1) , 1111,222,33,4 ..

runif - no of uniform random variables generator. , vectorize functions ,,

split() , x, f,
lapply(split(x,f), mean) - use split and then apply, almost the same as tapply() .

gl(3,10 - create e levels each repeats 10 times

interaction(f1, f2) , combine all the levels in level 1 with al those on 2.
concatenate .
— some empty levels and different ones. drop = TRUE , drop empty levels.

{{levels interaction factor , split ** redo!} } —

DEBUGGING TOOLS in R - fix after finding a problem.
message - execution will still continue
warning - ignore sth ,
error - fatal problem, stops execution of problem.
condition - generic condition , something unexpected can occur.
- Get a warning back after the

$invisible(x) - does not print the returned value/no autoprinting

debugging tools: traceback - look at how many functions you are in and when the error occurred , debug - flags a function of debug mode; will suspend execution and look at each expression in detail and look at it,. , browser - when execution of function is suspended , will run the function upntil that function and then finishes , trace- use a snipped of code. , recrover - when you get a error you get a message, execution of that funciton stops, recover is a error handler function; will stop right where the error occurred and work on it.

traceback() , tells you exactly where the error occurred. call immediately after the error occurs, call right away .

debug() , first print out everything, brings out the browser function. n for next, runs light by line, get to the line where the error occurs, will know where the error functions

$options(error = recover), browse the environment on a particular function, see what is going on in each function. Only error stops execution.

WEEK 4

$str = compactly display the internal structure of a R object. look at an object , look at its summer, useful for displaying large lists. Displays one line of output. Answer whats in his object?

Similar to @summary() , head()

Simulation : Rnorm , Dnorm, Norm , Rpois,
Rnorm - generate random Normal Variables. , rnom(10)
Dnorm , evaluate normal probability density
Pnorm evaluate cumulative distribution function
Rpois - generate random Passion

d - density
r- random number generation
p - cumulative distribution
q- quantile function

dbirn(x, mean = , sd = , log =)
Default , log = false, sd = 1,
Rnorm, Pnorm , lower.tail , log, p
Generate numbers (number, sd, norm).

$set.seed(1) - pass seed to , if seed is reset, set the sequence to go back to something, allows for you to reproduce something , you want to generate something to reproduce it .

rpois(10,1) , will be integers, rate of 1, Possion distribribution, rate/mean ,

Generating for linear model ,
x <- Rnorm(100) , rbinorm ,
e<- norm (100,0,2)
summary(U)
plot(x,y)

sample(replace=TRUE) , gets repetitions as well .
- useful to draw random samples from specific functions.

set seed every time you generate something to be able to reproduce results later.

PROFILER - useful to see what is going on , figure out why something is taking a lot of time.
profiling is better than guessing, optimize after coding.
First code then do a performance analysis

$system.time(svd(x)) , takes a time and , evaluate and get time until error has occurred, svd uses multicores, splits computation time across.
Elapsed time and user time- , system time, look at different ones.
Basic R program does not use multiple cores.
Can wrap a whole function in system time and get the time.

$Rprof() . summaryRprof() .
Do not use system.time() and Rprof() together, not designed to be worked together.
Rprof keeps tack of the call stack for default 0.02 seconds. if less than that useless. but if it is slower. look at callstack .

summaryRprof() , looks at how much time is spent in each functions.
Normaize the data using by.total (total time spent on function) and by.self() . top level functions usually call the helper functions that do all the work , not useful , if you want to look at top level after subtracting out all the lower level functions.
by.self subtracts out the lower level functions, look at which functions are taking what about of computation and investigate.

Assignments:
$object.size
$ names() - get names of columns
$dim () , ger rows X column, no of observations and no of columns
$table(plants$Active_Growth_Period) - get summary of individual column

$sample(1:6, 4, replace = TRUE)
flips2 <- rbinom(100,size=1 ,prob=0.7)
replicate(100, rpois(5, 10)) , generate 5 numbers with a mean of 10 , a 100 times
$ (rexp()), chi-squared (rchisq()), gamma (rgamma()), , other probability distributions in R.

lattice, ggplot2 and ggvis..
http://varianceexplained.org/r/teach_ggplot2_to_beginners/
plot(x = cars$speed, y = cars$dist)
plot(cars,xlim = c(10, 15)) - change limit for X axis.
col = 2 , plot in red,

hist(mtcars$mpg)
boxplot(formula = mpg ~ cyl, data = mtcars)