# Course:

URL: <https://class.coursera.org/rprog-015/lecture/117>

# Week 1:

R Programming

*> x <- 5*

Here:

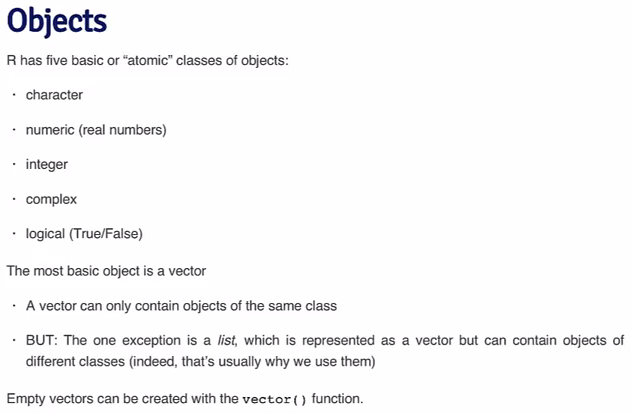
* “x” is the object, it is a numeric vector
* “<-“ is the assignment operator
* Auto printing occurs when you put an objects name

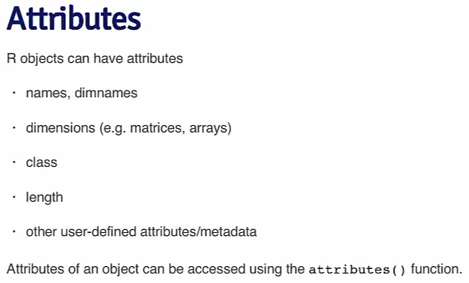
*> x <- 1:20*

Here:

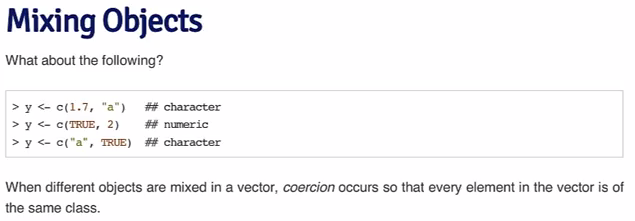
* The colon operator creates an integer sequence

There are 5 basic objects (atomic data types):





Coercion occurs when you mix objects of different classes in the same vector:

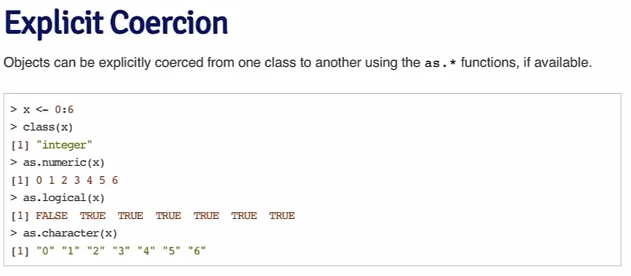


To understand the class and structure of R objects:

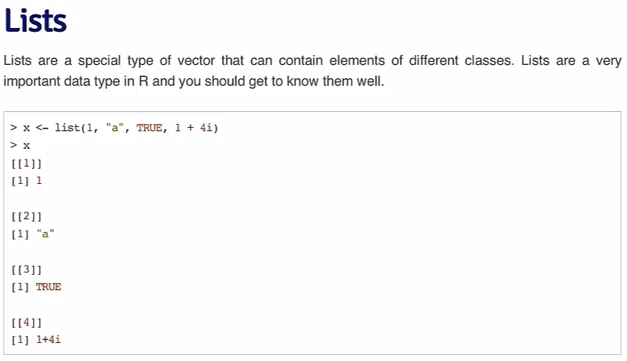
“str(object)”

“class(object)”

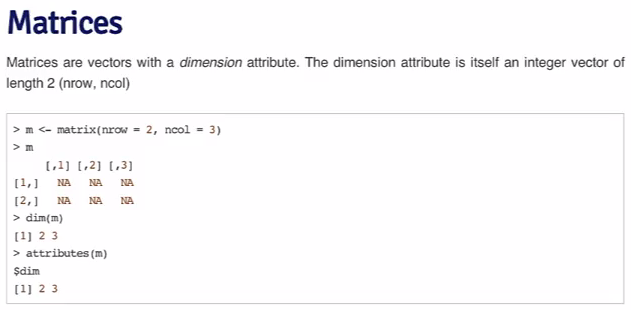
You can explicitly define the class of an object:



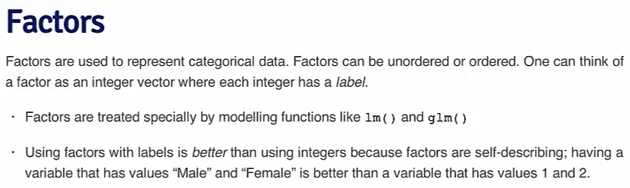
Lists:



Matrices:



Factors:



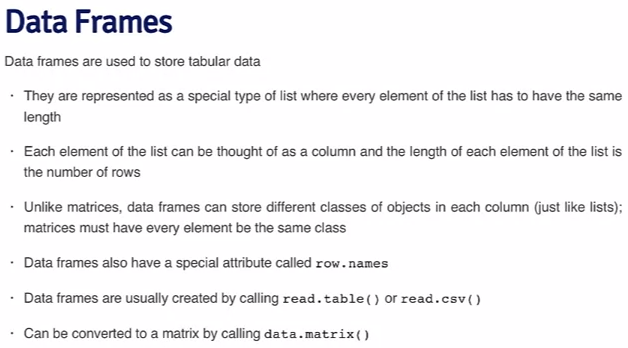
Useful factor functions:

“test <- factor(c(“male”, “male”))”

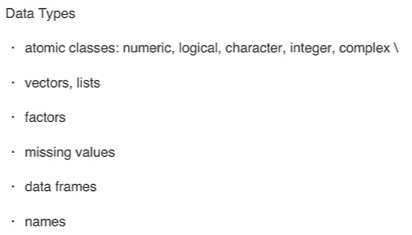
“table(test)”

“unclass(test)”

Data Frames:

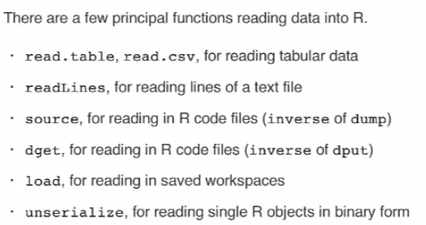


Summary of data types in R:

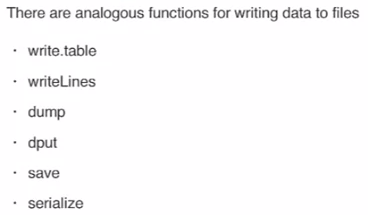


* vectors can only have elements of the same class (lists are an exception to that)
* data frames store tabular data where each column can be of a different class
* all objects can have names (useful for creating self-describing data)

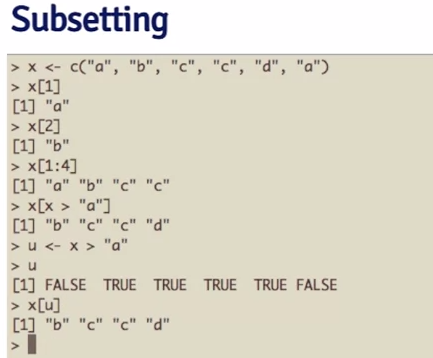
Reading data into R:

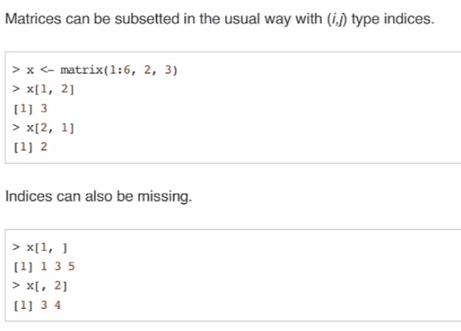


And writing:

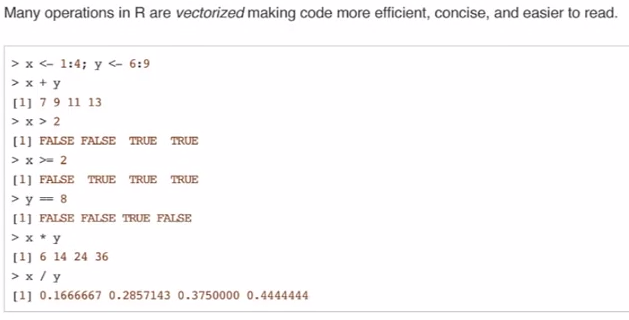


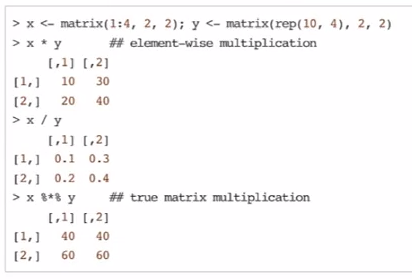
Subsetting:





Vectorized operations:





Learnt from the Quiz:

“hw1 <- read.csv(“hw1.csv”)” – reads the csv file into a dataframe

“hw1$Ozone[47]” – returns the value of Ozone in the 47th row of the dataset hw1

“hw1[is.na(hw1$Ozone), ]” – returns all the rows of hw1 where Ozone is NA

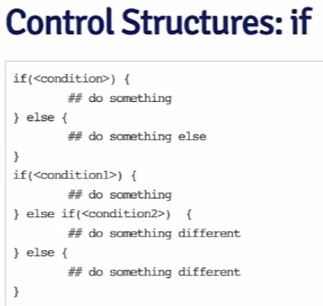
“mean(hw1$Ozone, na.rm = TRUE)” – calculates the mean of Ozone, removing all NA values

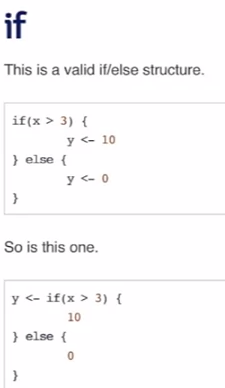
“max(hw1$Ozone[hw1$Month==5], na.rm=TRUE)” – calculates the maximum value of Ozone where Month is May, removing all NA values

“mean(hw1$Solar.R[hw1$Ozone > 31 & hw1$Temp > 90], na.rm=TRUE)” – calculates the mean of Solar.R where Ozone is greater than 31 and Temp is greater than 90, removing all NA values

# Week 2

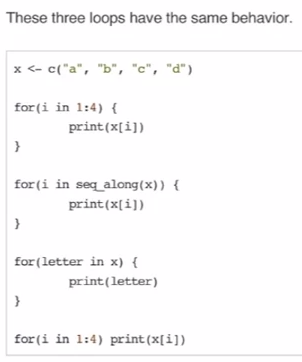
Control Structures: IF



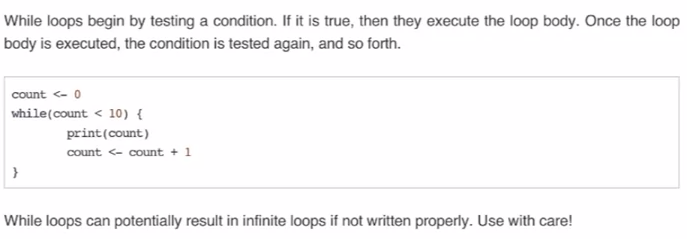


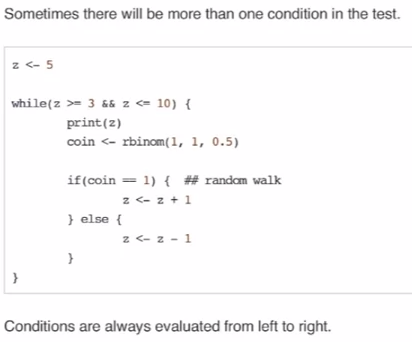
Note: “else” is not mandatory, if the if is false, you can just do nothing

For:

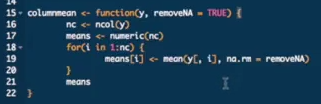


While loops:

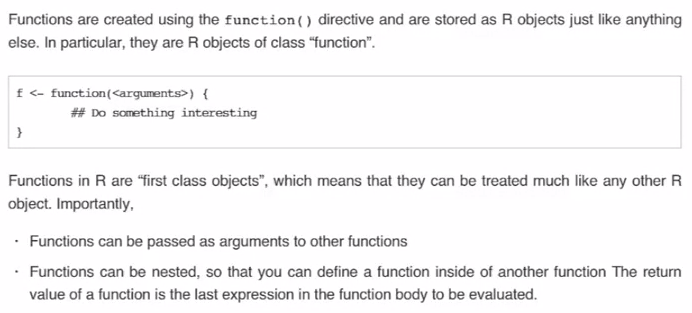




Creating functions:

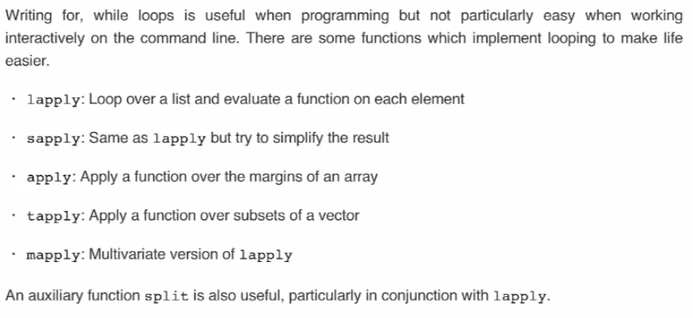


Functions:



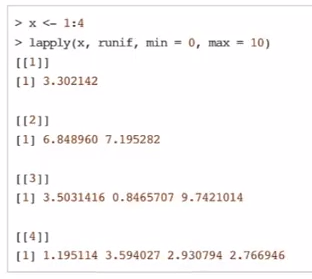
# Week 3

Looping:

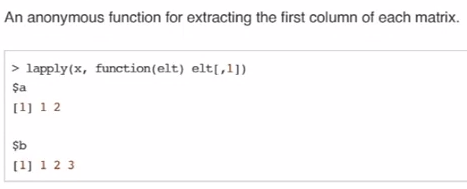


lapply always returns a list

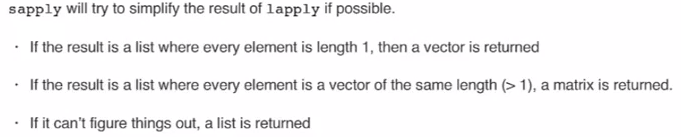
you can apply arguments to the lapply function:



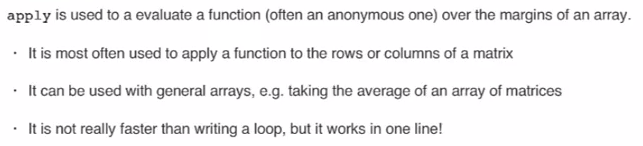
Anonymous functions only exist within the context of the lapply:



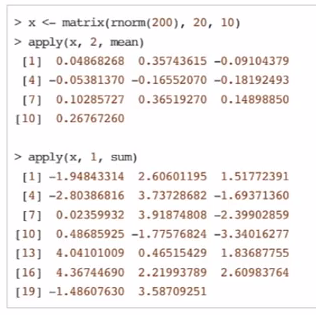
Sapply simplifies the result (where possible):

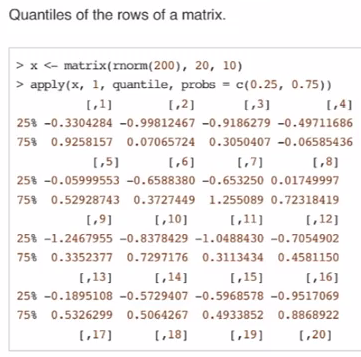


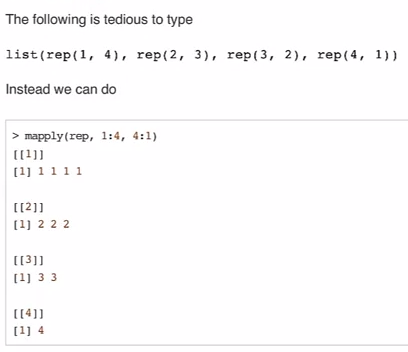
Apply is, again, usually leveraging anonymous functions and most often over a 2-dimensional matrix (most common form of array):



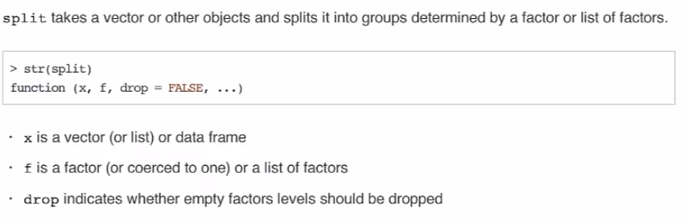
The margin tells the function which dimension to use – in the following, margin “2” means to preserve the columns, but collapse all the rows:



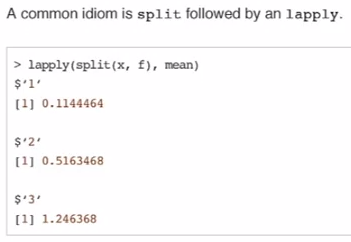


Mapply applies a function to multiple lists  


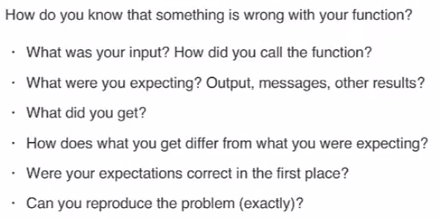
Split:

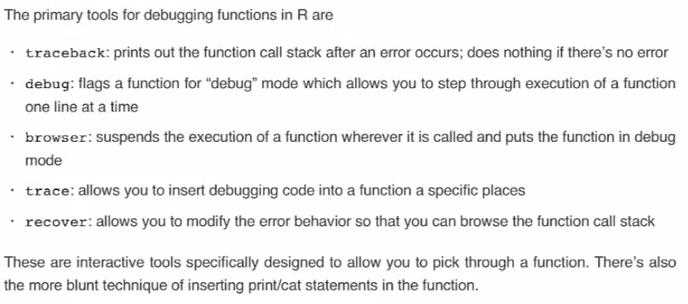


Commonly used within lapply:



Debugging steps:





# Week 4

Sample:

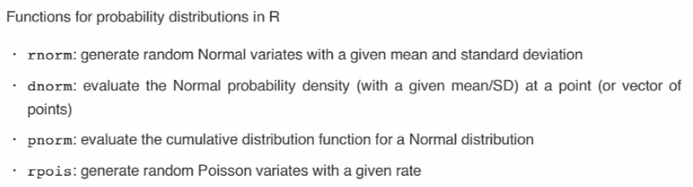
let's select 100 elements at random from these 2000 values

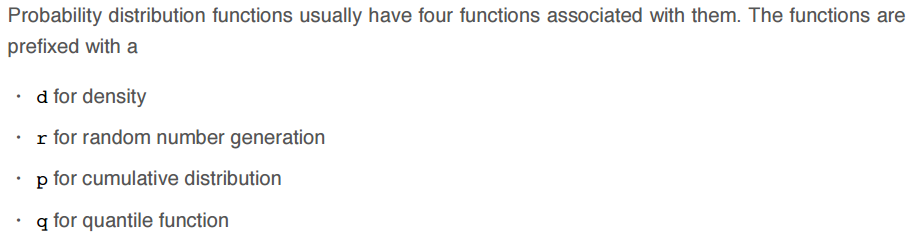
| (combining y and z) such that we don't know how many NAs we'll wind up with

| or what positions they'll occupy in our final vector -- my\_data <-

| sample(c(y, z), 100).

Generating random numbers from given probability distributions:



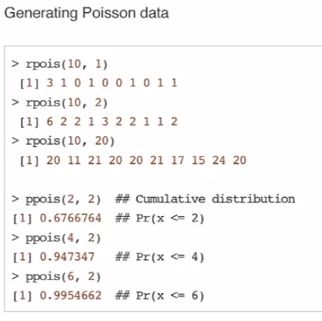


Computers actually create pseudo-random numbers, it’s therefore important to “set.seed” to ensure reproducibility…

With Poisson, the numbers are integers.

- You set the rate as part of the function (rate is the same as the mean?)

- The ppois function returns the probability of a number being less than or equal to a number, given a certain rate

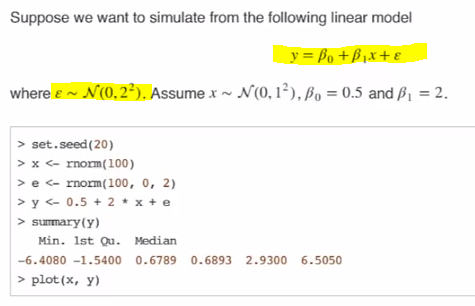


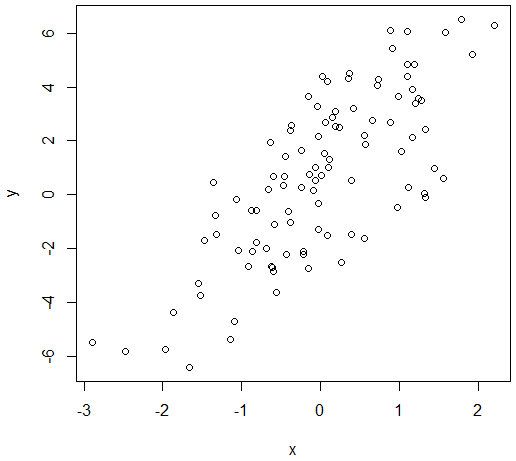
From a linear model:

- A linear model with a single predictor “x”

- It has noise via epsilon which has a mean of zero and a standard deviation of 2

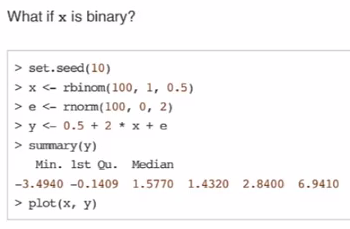
- There are two regression coefficients; intercept (beta zero) and slope coefficient (beta 1)

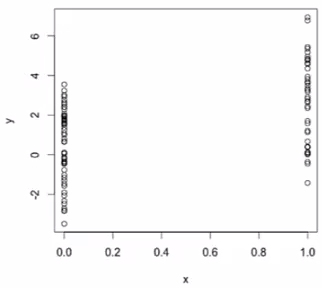




What is x was binary (e.g. gender, or treatment vs. control)?

- We can generate binary numbers with the binomial distribution



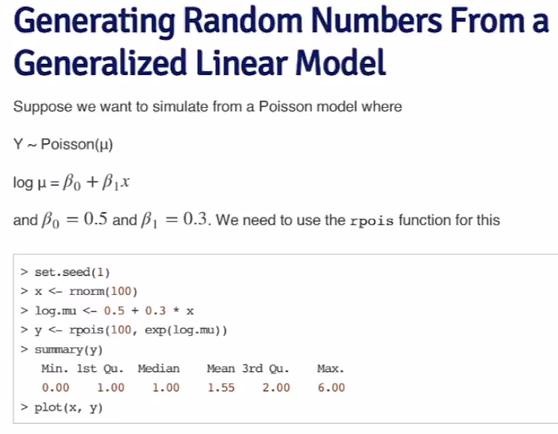


Poisson distribution:

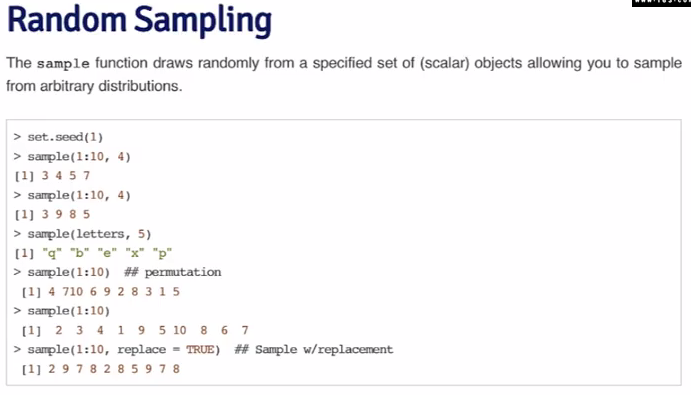
- count rather than continuous variables

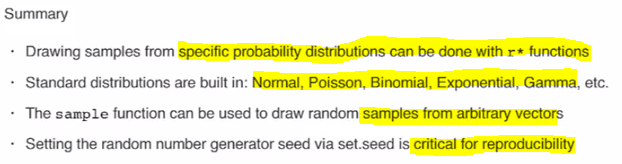
- Outcome of Y follows a poisson distrubtion

- The log of mu follows a linear model

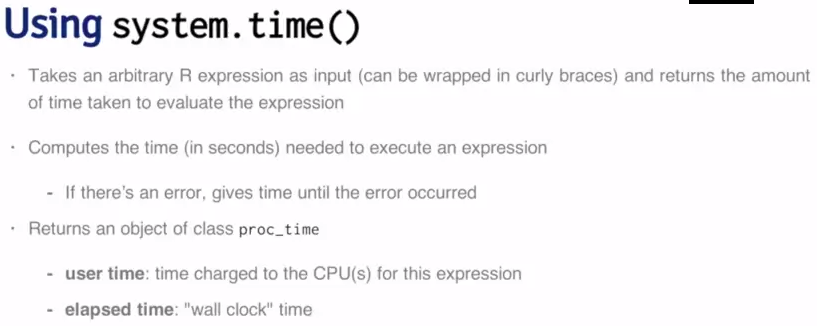


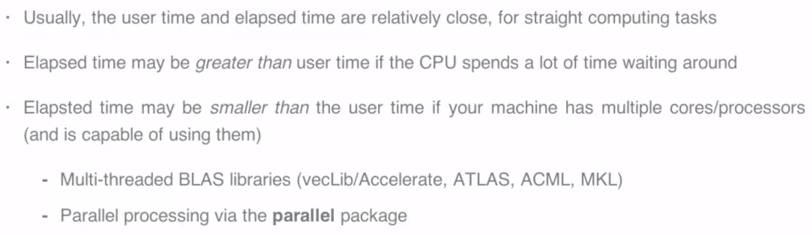
Sampling:





R Profiler – for performance tuning, optimization





Most computers are multi-core, but R doesn’t use them; some of the libraries can

You can see network time in the below call, so elapsed time is high:

