R Programming Basics

setwd('~/desktop/coursera')

data <- read.csv("coursera.csv", header=TRUE)

names(data) *# Extract names of dataset*

## [1] "Ozone" "Solar.R" "Wind" "Temp" "Month" "Day"

head(data) *# First 6 rows of dataset*

## Ozone Solar.R Wind Temp Month Day

## 1 41 190 7.4 67 5 1

## 2 36 118 8.0 72 5 2

## 3 12 149 12.6 74 5 3

## 4 18 313 11.5 62 5 4

## 5 NA NA 14.3 56 5 5

## 6 28 NA 14.9 66 5 6

tail(data) *# Last 6 rows of dataset*

## Ozone Solar.R Wind Temp Month Day

## 148 14 20 16.6 63 9 25

## 149 30 193 6.9 70 9 26

## 150 NA 145 13.2 77 9 27

## 151 14 191 14.3 75 9 28

## 152 18 131 8.0 76 9 29

## 153 20 223 11.5 68 9 30

nrow(data) *# Number of observations*

## [1] 153

data[47,1] *# Find the value of ozone in the 47th row*

## [1] 21

sum(is.na(data[,1])) *# Find number of missing values in Ozone column*

## [1] 37

mean(na.omit(data[,1])) *# Find mean of Ozone column excluding missing values*

## [1] 42.12931

ozone <- subset(data, !is.na(Ozone), select = Ozone) *# We will consider this approach instead*

apply(ozone, 2, mean) *# For a data.frame, a margin of 2 indicates columns*

## Ozone

## 42.12931

*# Extract the subset of rows of the data frame where Ozone values are above 31 and Temp values are above 90. What is the mean of Solar.R in this subset?*

solar <- subset(data, Temp>90 & Ozone>31, select=Solar.R)

apply(solar, 2, mean)

## Solar.R

## 212.8

*# Find the mean of "Temp" when "Month" is equal to 6.*

temp <- subset(data, Month==6, select=Temp)

apply(temp, 2, mean)

## Temp

## 79.1

*# Find the maximum ozone value in the month of May (i.e. Month = 5)*

ozone2 <- subset(data, Month==5 & !is.na(Ozone), select=Ozone)

apply(ozone2, 2, max)

## Ozone

## 115

*## Factors*

x <- factor(c("yes", "no", "yes", "no", "no", "no"))

table(x) *## Frequency table*

## x

## no yes

## 4 2

unclass(x) *## See how factors are represented in R. Stripping the class and left with integers.*

## [1] 2 1 2 1 1 1

## attr(,"levels")

## [1] "no" "yes"

Missing values

*## is.na() Used to test objects if they're NA*

*## is.nan() Used to test if they're NaN*

x <- c(1,2,3,NA,10)

is.na(x)

## [1] FALSE FALSE FALSE TRUE FALSE

is.nan(x)

## [1] FALSE FALSE FALSE FALSE FALSE

y <- c(1,2,3,NaN,10)

is.na(y)

## [1] FALSE FALSE FALSE TRUE FALSE

is.nan(y)

## [1] FALSE FALSE FALSE TRUE FALSE

data.frame

z <- data.frame(Dog=c(1,2,3,4), Cat=c("a", "b", "c", "d"))

z

## Dog Cat

## 1 1 a

## 2 2 b

## 3 3 c

## 4 4 d

nrow(z)

## [1] 4

ncol(z)

## [1] 2

Names Attribute

x <- 1:3

names(x)

## NULL

names(x) <- c("a", "b", "c")

m <- matrix(1:4, nrow=2, ncol=2, byrow=TRUE)

dimnames(m) <- list(c("a", "b"), c("c", "d"))

m

## c d

## a 1 2

## b 3 4

Reading tabular data

read.csv("coursera.csv", header=TRUE, sep=",") ## If your data is separated by commas

Reading in Large Datasets

setwd('~/desktop/Ostats')

gestation <- read.table("gestation.txt", nrow=10) *## Read in data of the first 10 rows to get classes*

classes <- sapply(gestation, class) *## Find classes of each column*

tabAll <- read.table("gestation.txt", colClasses=classes) *## Tell R the classes. The intuition behind this is that if you can tell R the types of data that are in each column, then R won't spend extra time and memory figuring them out on its own.*

*## Let's say we have a data.frame with 1,500,000 rows and 120 columns. How many gigs of memory will it take up?*

1500000\*120\*8 *## There are 8 bytes per numeric*

## [1] 1.44e+09

(1500000\*120\*8)/(2^20) *## There are 2^20 bytes per MB*

## [1] 1373.291

((1500000\*120\*8)/(2^20))/1000 *## There are 1000 MBs per GB*

## [1] 1.373291

Subsetting

*# [always returns an object of the same class as the original]*

*# [[used to extract elements of a list of data.frame; it can only be used to extract a single element and the class of the returned object will not necessarily be a list or data.frame]]*

*# $ used to extract element from list or data.frame by name*

x <- c("a", "b", "c", "c", "d", "a")

x[1]

## [1] "a"

x[1:4]

## [1] "a" "b" "c" "c"

x[x > "a"] *## This will return a character vector*

## [1] "b" "c" "c" "d"

u <- x > "a" *## This will return a logical vector*

u

## [1] FALSE TRUE TRUE TRUE TRUE FALSE

Lists

x <- list(dog=c(1,2,3,4), cat=3.14, squirrel="Hello")

x[1] *## Returns a list ALWAYS*

## $dog

## [1] 1 2 3 4

x[[1]] *## Returns a vector*

## [1] 1 2 3 4

x[c(1,3)] *## Extract multiple elements, returns a list since it's [ ]*

## $dog

## [1] 1 2 3 4

##

## $squirrel

## [1] "Hello"

x[[c(1,3)]] *## Returns a single element; same as below*

## [1] 3

x[[1]][[3]]

## [1] 3

name <- "dog" *## Same as below*

x[[name]]

## [1] 1 2 3 4

x[["dog"]]

## [1] 1 2 3 4

x$dog

## [1] 1 2 3 4

Matrices

x <- matrix(1:6, 2, 3) *## A matrix from 1 to 6 with 2 rows and 3 columns*

x[1,2] *## First row 2nd column*

## [1] 3

x[1,] *## First row*

## [1] 1 3 5

x[,2] *## Second column*

## [1] 3 4

*## By default, when a single element is retrieved from a matrix, it is returned as a vector of length 1 rather than a 1x1 matrix. This can be turned off with the following:*

x[1,2, drop=FALSE]

## [,1]

## [1,] 3

x[1, , drop=FALSE]

## [,1] [,2] [,3]

## [1,] 1 3 5

Removing NA

z <- c(1, 2, NA, 3, NA, 4)

bad <- is.na(z)

z[!bad] *## ! means not*

## [1] 1 2 3 4

w <- c("a", "b", NA, "c", NA, "d")

good <- complete.cases(z,w) *## Tells us which combinations are not true*

good *## i.e., 1 and "a" are good, 2 and "b" are good, and so forth...*

## [1] TRUE TRUE FALSE TRUE FALSE TRUE

z[good]

## [1] 1 2 3 4

w[good]

## [1] "a" "b" "c" "d"

*## Consider the coursera dataset*

head(data)

## Ozone Solar.R Wind Temp Month Day

## 1 41 190 7.4 67 5 1

## 2 36 118 8.0 72 5 2

## 3 12 149 12.6 74 5 3

## 4 18 313 11.5 62 5 4

## 5 NA NA 14.3 56 5 5

## 6 28 NA 14.9 66 5 6

good <- complete.cases(data) *## The logical vector here tells me which rows are complete*

data[good,][1:6,]

## Ozone Solar.R Wind Temp Month Day

## 1 41 190 7.4 67 5 1

## 2 36 118 8.0 72 5 2

## 3 12 149 12.6 74 5 3

## 4 18 313 11.5 62 5 4

## 7 23 299 8.6 65 5 7

## 8 19 99 13.8 59 5 8

Vectorized matrix operations

x <- matrix(1:4,2,2); y<-matrix(rep(10,4),2,2) *## y is a matrix of all 10's, i.e., 4 10's*

x; y

## [,1] [,2]

## [1,] 1 3

## [2,] 2 4

## [,1] [,2]

## [1,] 10 10

## [2,] 10 10

x\*y *## Element-wise multiplication*

## [,1] [,2]

## [1,] 10 30

## [2,] 20 40

x %\*% y *## True matrix multiplication*

## [,1] [,2]

## [1,] 40 40

## [2,] 60 60