# Slide # 29

matrix(c(1:5, NA, NA, NA),nrow=2,byrow=T)

matrix(c(1:5, rep(NA,3)),nrow=4,byrow=T)

# Slide # 30

# To Locate the Current Working Directory

getwd()

Company.employees <- read.table("EmployeeSales.txt", TRUE, sep = " ", quote="\"")

Company.employees

write.table(Company.employees,"EmployeeSales2.txt")

# Slide # 31

employees <- read.table("EmployeeSales.txt", TRUE, quote="\"",

na.strings = c(“NA”,””), strip.white = TRUE,

blank.lines.skip = TRUE)

print(employees)

# Slide # 32

employeeNames <- c("Employee\_ID", "First Name", "Last Name", "Education", "Profession","Salary","Sales")

employees <- read.table("EmployeeSales.txt", TRUE, quote="\"",

strip.white = TRUE, skip = 3,

as.is = c(TRUE, TRUE, FALSE, FALSE, TRUE),

col.names = employeeNames,

blank.lines.skip = TRUE)

print(employees)

print(str(employees))

# Slide # 34

# first row contains variable names

# row.names assigns the variable id to row names

# If we do not specify row.names then it would create another running serial number to it

mydata <- read.csv("empdata.csv", header=TRUE, row.names="Employee\_ID")

mydata

#Write to Comma Delimited Text File

write.csv(mydata, "MyEmpData.csv")

# Slide # 35

write.xlsx(mydata, "EmployeeSales.xlsx", row.names= F)

setwd("C:/Users/Atin/Desktop/RProgramming/WorkingD")

install.packages("xlsx")

library(xlsx)

mydata <- read.xlsx("EmployeeSales.xlsx", 1)

mydata

# read in the worksheet named mysheet

mydata <- read.xlsx("EmployeeSales.xlsx", sheetName = "Sheet 1")

mydata

# Slide # 36

readClipboard() #Only on Windows

read.table(file="clipboard", sep="\t")

# writeClipboard example

x <- "hello world"

#copies the string "hello world" to the clipboard as expected.

writeClipboard(x)

#However the below code produces the error message

x <- 3.14

writeClipboard(x)

# Slide # 37

install.packages("psych")

library(psych)

My\_data= read.clipboard.csv(header=TRUE,sep=',')

My\_data

print(read.clipboard.tab(header=TRUE,sep='\t'))

# Slide # 38

my\_data= read.clipboard.csv(header=TRUE,sep=',')

saveRDS(my\_data, file = "my\_data.rds")

rm(my\_data)

my\_data

my\_data = readRDS(file = "my\_data.rds")

my\_data

# Slide # 39

# Save a single object to a file

saveRDS(mtcars, "mtcars.rds")

# Restore it under a different name

my\_data <- readRDS("mtcars.rds")

#Save multiple objects to a file

a=1;b=2;c=3;d="Sumit";e=T

a;b;c;d;e

save(a, file = "data.RData") # Saving on object in RData format

# Save multiple objects

save(b,c,d,e, file = "data.RData")

rm(b);rm(c);rm(d);rm(e)

b;c;d;e

load("data.RData") # To load the data again

b;c;d;e

# Slide # 40

# Start writing to an output file

sink('analysis-output.txt')

a=1;b=2;c=3;d="Sumit";e=T

a;b;c;d;e

sink() # return out back to termincal

a;b;c;d;e

# Slide # 44

mydata = read.csv("Sample Data in R Programming-Dplyr.csv")

#The sample\_n function selects random rows from a data frame. The second parameter of the function tells R the number of rows to select.

sample\_n(mydata,3)

#The sample\_frac function returns randomly N% of rows. In the example below, it returns randomly 10% of rows.

sample\_frac(mydata,0.1)

# Slide # 45

#Remove Duplicate Rows based on all the variables (Complete Row)

distinct(mydata)

#Remove Duplicate Rows based on a variable

distinct(mydata, Index, .keep\_all= TRUE)

#The code below selects variables "Index", columns from "State" to "Y2008".

select(mydata, Index, State:Y2008)

# Slide # 46

#The minus sign before a variable tells R to drop the variable.

select(mydata, -Index, -State)

#The starts\_with() function is used to select variables starts with an alphabet.

select(mydata, starts\_with("Y"))

#Selecting Variables contain 'I' in their names

select(mydata, contains("I"))

#The code below keeps variable 'State' in the front and the remaining variables follow that.

select(mydata, State, everything())

# Slide # 47

#In the following code, we are renaming 'Index' variable to 'Index1'.

rename(mydata, Index1=Index)

#Suppose you need to subset data. You want to filter rows and retain only those values in which Index is equal to A.

filter(mydata, Index == "A")

#Suppose you need to apply 'AND' condition. In this case, we are picking data for 'A' and 'C' in the column 'Index' and income greater than 1.3 million in Year 2002.

filter(mydata, Index %in% c("A", "C") & Y2002 >= 1300000 )

#The 'I' denotes OR in the logical condition. It means any of the two conditions.

filter(mydata, Index %in% c("A", "C") | Y2002 >= 1300000)

#The "!" sign is used to reverse the logical condition.

filter(mydata, !Index %in% c("A", "C"))

# Slide # 48

#Calculating mean and median for the variable Y2015.

summarise(mydata, Y2015\_mean = mean(Y2015), Y2015\_med=median(Y2015))

#sorting data by multiple variables.

arrange(mydata, Index, Y2011)

# Slide # 49

df <- data.frame( c( 183, 85, 40), c( 175, 76, 35), c( 178, 79, 38 ))

names(df) <- c("Height", "Weight", "Age")

# All Rows and All Columns

df[,]

# First row and all columns

df[1,]

# First two rows and all columns

df[1:2,]

# Slide # 50

# First and third row and all columns

df[ c(1,3), ]

# First Row and 2nd and third column

df[1, 2:3]

# First, Second Row and Second and Third Column

df[1:2, 2:3]

# Slide # 51

#Although R can calculate accurately to up to 20 digits, you don’t always want to use that many digits.

format(22/7,nsmall=20)

#To round a number to two digits after the decimal point, use the round() function as follows:

round(22/7,digits=2)

# Slide # 52

#You also can use the round() function to round numbers to multiples of 10, 100, and so on. For that, you just add a negative number as the digits argument:

round(-123.456,digits=-2)

#If the first digit that is dropped is exactly 5, it rounds to the nearest even number.

#Both the 2 functions below same output - 2

round(1.5)

round(2.5)

# and below function returns -4

round(-4.5)

# Slide # 54

a <- "apple"

b <- "banana"

# Put a and b together, with a space in between:

paste(a, b)

# Slide # 55

# With no space, use sep="", or use paste0():

paste(a, b, sep="")

# With a comma and space:

paste(a, b, sep=", ")