1. **Write a R program using control operators to test whether following values are prime numbers or not by providing a PRIME or NOT PRIME message as output: A. 103 B. 82 C. 179**

f.checkPrime<- function(x)

{

i<-2

while(i<=x)

{

if(x%%i==0 & x!=i)

{

print(paste(x,"is a Non-prime number", sep = " "))

break()

}

else if(x==i)

{

print(paste(x,"is a Prime number", sep = " "))

break()

}

i<-i+1

}

}

> f.checkPrime(103)

[1] "103 is a Prime number"

> f.checkPrime(82)

[1] "82 is a Non-prime number"

> f.checkPrime(103)

[1] "103 is a Prime number"

1. **Write a R program using control operators to identify letter u and a both occur in the following words: 1. above 2. unit 3. Under**

> f.findLetterUA<-function(x)

+ {

+ return(grep("u",x, ignore.case = T) & grep("a",x, ignore.case = T))

+

+ }

> f.findLetterUA("above")

logical(0)

> f.findLetterUA("unit")

logical(0)

> f.findLetterUA("Under")

logical(0)

> f.findLetterUA("TestaUnder")

[1] TRUE

**3. Write a function that to calculate BMI (Body Mass Index):**

**BMI for a person is defined as their body mass divided by the square of their height**

**The weight is in kilograms and the height in meters or**

**(The weight can be in pounds and the height in inches)\* 703**

**Check your BMI :**

f.getBMI <- function(weight, height) {

BMI<-weight/height^2

BMICategory<-NULL

if(BMI<15)

{

BMICategory<-"Very severely underweight"

}

else if(BMI<16.0 & BMI>=15)

{

BMICategory<-"Severely underweight"

}

else if(BMI<18.5 & BMI>=16)

{

BMICategory<-"Underweight"

}

else if(BMI<25 & BMI>=18.5)

{

BMICategory<-"Underweight"

}

else if(BMI<30 & BMI>=25)

{

BMICategory<-"Overweight"

}

else if(BMI<35 & BMI>=30)

{

BMICategory<-"Obese Class I (Moderately obese)"

}

else if(BMI<40 & BMI>=35)

{

BMICategory<-"Obese Class II (Severely obese)"

}

else if(BMI>=40)

{

BMICategory<-"Obese Class III (Very severely obese)"

}

return(c(BMI,BMICategory))

}

> f.getBMI(75,1.72)

[1] "25.3515413737155" "Overweight"

**4. Write a function called sum\_of\_cubes, that calculates the sum of cubes of the first n natural numbers :**

**if we have two numbers : 1, 2 then sum of squares is 9 ( 1^3 + 2^3)**

**if we have three numbers : 1, 2, 3 then sum of squares is 36 ( 1^3 + 2^3 + 3^3)**

sum\_of\_cubes<-function(x)

{

numcub<-1

while(x>1)

{

numcub<-x^3+numcub

x=x-1

}

return(numcub)

}

> sum\_of\_cubes(1)

[1] 1

> sum\_of\_cubes(2)

[1] 9

> sum\_of\_cubes(3)

[1] 36

**5. Write a function to calculate the mode (highest frequency) of the following vector:**

**x = c(2,3,3,4,4,5,6,7,9,10)**

get\_Mode<-function(x)

{

temp <- table(as.vector(x))

return(names(temp)[temp == max(temp)])

}

> x = c(2,3,3,4,4,5,6,7,9,10)

> get\_Mode(x)

[1] "3" "4"

**6. Write a function to calculate the no. of prime numbers of the following vector :**

**x = c(2,2,3,3,4,5,7,11,15,19,24,29)**

checkPrime<- function(x)

{

i<-2

isPrime<-NULL

while(i<=x)

{

if(x%%i==0 & x!=i)

{

isPrime<-0

break()

}

else if(x==i)

{

isPrime<-1

break()

}

i<-i+1

}

return(isPrime)

}

num\_primeNum\_in\_Vector <- function(x)

{

counter<-0

for(i in x)

{

counter<-counter+checkPrime(i)

}

return(counter)

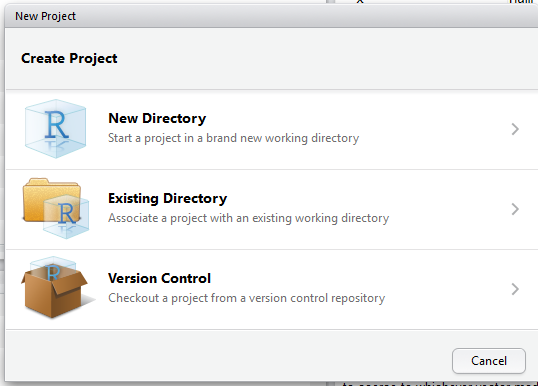
}

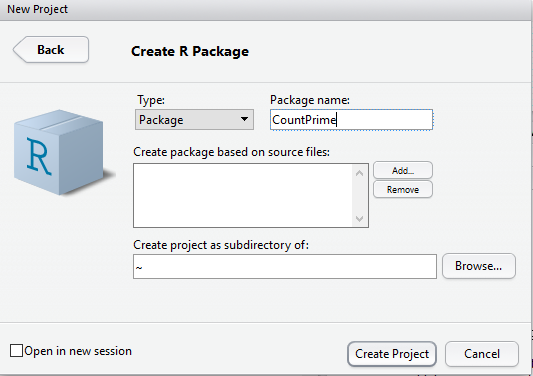
> x = c(2,2,3,3,4,5,7,11,15,19,24,29)

> num\_primeNum\_in\_Vector(x)

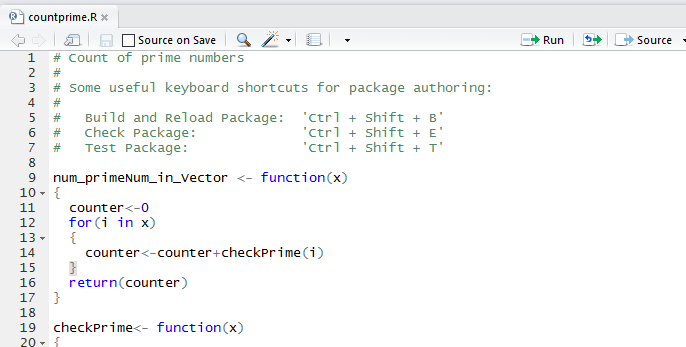
[1] 9

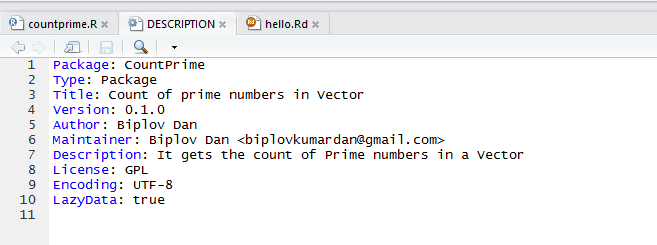
7. Create a R package for calculating the count of prime numbers , name it as “CountPrime

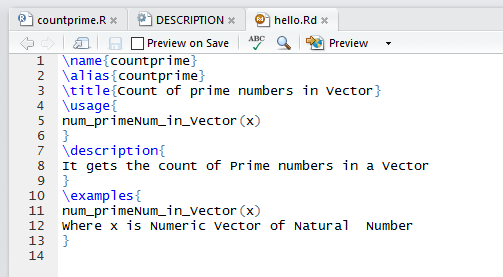


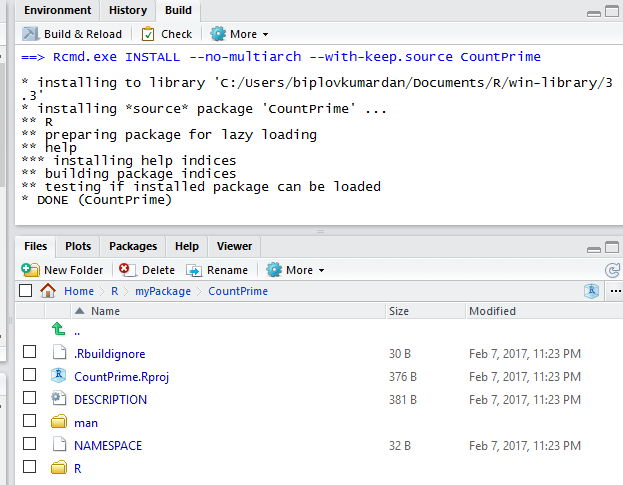


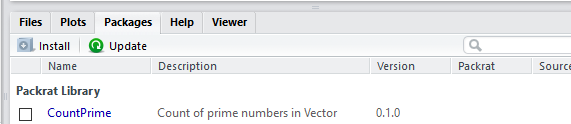


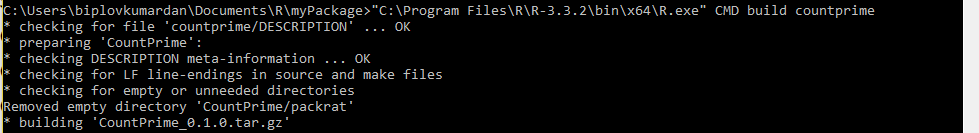


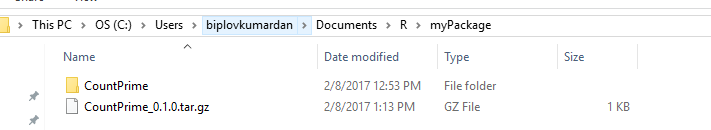












**8. Perform below operations using Data.frame and Data.table**

**a. Load 2 files (.csv) and show it on screen ( F1 – empno, deptid,mgr\_id , F2 – empno, sal, DOJ)**

> library(readr)

> F1 <- read\_csv("~/R/myPackage/workspace/F1.csv")

Parsed with column specification:

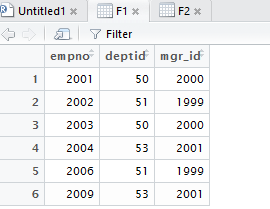
cols(

empno = col\_integer(),

deptid = col\_integer(),

mgr\_id = col\_integer()

)



> library(readr)

> F2 <- read\_csv("~/R/myPackage/workspace/F2.csv")

Parsed with column specification:

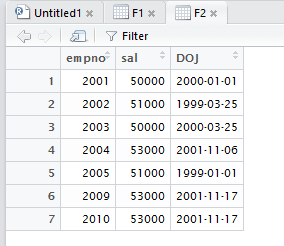
cols(

empno = col\_integer(),

sal = col\_double(),

DOJ = col\_date(format = "")

)



**b. Perform equi join**

> merge(F1,F2,by="empno")

empno deptid mgr\_id sal DOJ

1 2001 50 2000 50000 2000-01-01

2 2002 51 1999 51000 1999-03-25

3 2003 50 2000 50000 2000-03-25

4 2004 53 2001 53000 2001-11-06

5 2009 53 2001 53000 2001-11-17

**c. Perform left outer join**

> merge(F1,F2,by="empno", all.x = T)

empno deptid mgr\_id sal DOJ

1 2001 50 2000 50000 2000-01-01

2 2002 51 1999 51000 1999-03-25

3 2003 50 2000 50000 2000-03-25

4 2004 53 2001 53000 2001-11-06

5 2006 51 1999 NA <NA>

6 2009 53 2001 53000 2001-11-17

**d. Perform right outer join**

merge(F1,F2,by="empno", all.y = T)

empno deptid mgr\_id sal DOJ

1 2001 50 2000 50000 2000-01-01

2 2002 51 1999 51000 1999-03-25

3 2003 50 2000 50000 2000-03-25

4 2004 53 2001 53000 2001-11-06

5 2005 NA NA 51000 1999-01-01

6 2009 53 2001 53000 2001-11-17

7 2010 NA NA 53000 2001-11-17

**e. Perform full outer join**

> merge(F1,F2,by="empno", all=T)

empno deptid mgr\_id sal DOJ

1 2001 50 2000 50000 2000-01-01

2 2002 51 1999 51000 1999-03-25

3 2003 50 2000 50000 2000-03-25

4 2004 53 2001 53000 2001-11-06

5 2005 NA NA 51000 1999-01-01

6 2006 51 1999 NA <NA>

7 2009 53 2001 53000 2001-11-17

8 2010 NA NA 53000 2001-11-17

**f. Perform filter operation –Eg find all the rows for which col1 is null**

> library(sqldf)

> sqldf("Select \* from F1 left join F2 on F1.empno=F2.empno where deptid>50")

empno deptid mgr\_id empno sal DOJ

1 2002 51 1999 2002 51000 1999-03-25

2 2004 53 2001 2004 53000 2001-11-06

3 2006 51 1999 NA NA <NA>

4 2009 53 2001 2009 53000 2001-11-17

**g. Perform group by , sum, average operation**

> sqldf("Select deptid,sum(sal) salsum,avg(sal) salavg from F1 left join F2 on F1.empno=F2.empno

+ group by deptid")

deptid salsum salavg

1 50 100000 50000

2 51 51000 51000

3 53 106000 53000

**h. Perform (A “-“ B) operation**

**i. Create a derived column (empname in F2) – do some data transformation on that**

**j. Create a “working” test that example**

**Write output for each in separate files (eg- equijoin.csv ,leftoutjoin.csv)**

**9. Create R functions for the following operations**

**a. Find out unique combinations of data based on a particular column or group of columns.**

**i. Example:**

**– Select count(distinct stdid) from student group by classid**

**- Select count(distinct stdid) from student group by classid, sectionid**

**• Write a function to find out duplicate entries based on a key or group of keys**

**10. Create R functions for the following operations**

**a. Find out if there are any nulls in a dataset or in some specific number of columns**

**b. Write a function to read data from hdfs and dump it back to hdfs**