```
# P0 841
# 9/10/2014
# Lab 1: Introduction to R

#BASIC OPERATIONS

x<-9
x
y<-5
y
x+y
z<-x+y
z
z<-x*y
z
z<-x/y
z
z<-x^2
z
z<-x+y*x
z
z<-(2*x + ((x+y)*x))^2 #Remember PEMDAS
z^2
w<-2
w

ls() # List the objects you have created

#**********************************************************************

#VECTORS

a<-c(1,2,3,4,5)
a
a*20
a^2
a<-c(x,y,z)
a
a<-1:10 #1 through 10; renaming a here will override previously-defined a
a

a<-rep(4,10) #4 repeated ten times
a
a<-rep(c(1,2,3),3) #repeat the sequence within the parentheses however many
 times
a
a<-seq(3,15,3) #make a vector from the numbers between 3 and 15 by threes;
 could also do a<-seq(1,10,2) to make a vector from the numbers from one to
```

```
 ten by twos, starting with the first nunber (one)
a
seq(1,10,2)
?rep
help(seq)

length(a) # number of elements in vector a

#Exercise: Create a vector containing all of the even integers from 2 to 20.
 How many ways can you do it? How about the odd integers from 1 to 19?

x<-c(2,4,6,8,10,12,14,16,18,20)
x
x<-rep(c(2,4,6,8,10,12,14,16,18,20),1)
x
x<-seq(2,20,2)
x
x<-rep(seq(2,20,2),1)
x

y<-c(1,3,5,7,9,11,13,15,17,19)
y
y<-rep(c(1,3,5,7,9,11,13,15,17,19),1)
y
y<-seq(1,19,2)
y
y<-rep(seq(1,19,2),1)
y

#*************************************************************************

#INDEXING

b<-x[1] #First element
b
c<-x[5]
c
b/c
x[4] #4th element
x[2:4] #2ns, 3rd and 4th element
x[c(1,3,5)] #Make a vector from the 1st, 3rd and 5th elements
x[-3] #Everything BUT element 3
x[1]<-100 #Change the first element of x to 100
x
x[1]<-NA #Make the first element NA
x

x[6] #What is the 6th element of x?
```

```r
x[6]<-9 #Replace the 6th element of x with 9

#Correcting an element
#Say you accidentally type 12 instead of 6 in the following sequence:
x<-c(1,2,3,4,5,12,7,8,9,10)
x
x[x=6]

#Identifying and replacing elements (per Bill's question):

x
x==12 #You know you mis-entered a value of 12 when you should have entered a
 value of 6. Identify where the mistake is: Where in this vector does x = 12?
x[x==12] #Which elements fulfill the condition that x = 12?
sum(x==12) #How many of the vector's elements fulfill the condition that x=12?

#Replace that element with the correct value, 6
x[x==12]<-6 #For any element wherein x=12, replace it with 6
x

#*********************************************************************

#MATRICES

m<-matrix(11:20,ncol=2,nrow=5) #matrix with numbers 11-20, 2 columns, 5 rows,
 organized by column first (default setting)
m
m<-matrix(11:20,ncol=2,nrow=5,byrow=TRUE) #organized by row first
m
m[5,] #[Row, Column] a specific row and all columns
m[,2] #[Row, Column] a specific column and all rows
m[5,2] #[Row, Column] a specific row and a specific column--i.e., an element

c(m,x) #Turns matrix into a vector -- be careful with this command!
x
#To combine columns, use cbind()

m1<-cbind(x,m) #combines by column
m1

#To combine rows, use rbind()
m
22:23
m2<-rbind(m,22:23)
m2

#*********************************************************************
```

```
#MATRIX MATH

m
n<-matrix(1:10,nrow=5,ncol=2)
n
m+n #adds matrices; adds the element in m to the element in n to create a new
 element n+m
m+10 #adds ten to every element
x
m+x #adds x to every element
x1<-c(x,1) #adds 1 to the end of vector x
x1
m+x1 #Error message will appear if you try to add a vector/matrix to a vector/
 matrix of a different length or dimension. For this kind of matrix math, both
 (or all) matrices must be conformable, i.e., of the same dimesionality.

dim(m) #dimensions - 5 rows, 2 columns
dim(n)
m*n #Multiply element in m by element in n
m/n #Divide element in m by element in n
t(n) #transpose
t(m)
m%*%t(n) #Matrix multiplication is %*%; most matrix math (later in the course)
 will ne more complicated than just multiplying one element of a matrix by a
 corresponding element in another matrix.

#*********************************************************************

#MORE MATH
x
exp(x) #exponential function, Euler's constant^x
log(x) #logarithm
sqrt(x) #square root
abs(x) #absolute value
abs(-x) #absolute value of -x

sum(x)
2+4+6+8+10+12+14+16+18+20
prod(x)
2*4*6*8*10*12*14*16*18*20

#Remember, R can be used as a calculator. This will help save time on exams.

# Exercise: Create a 5*3 matrix (5 rows, 3 columns) where the first column is
 the integers from 1 to 5, the second column is the square of these integers,
 and the third column is their square root.

x<-1:5
```

```
y<-x^2
z<-(sqrt(x))
x
y
z
m<-matrix(cbind(x,y,z),ncol=3,nrow=5)
m
#OR
m<-matrix(cbind((1:5),((1:5)^2),sqrt(1:5)), ncol=3, nrow=5)
m

#**********************************************************************

#LOGICAL OPERATIONS

z<-1:10
z
z==3 #Always use 2 equal signs to test equality! Does z equal 3?
z!=3 #Does z not equal 3?
z<3 #Is z less than 3?
z<=3 #Is z greater than or equal to 3?
z<3 | z>5 #| stands for OR -- is z less than 3 OR greater than 5?
z>3 & z<5 #& stands for AND -- is z greater than 3 AND less than 5?
z[z<3] #Which elements fulfill the condition that z is less than 3?
z[!z<3] #Which elements do NOT fulfill the condition that z is less than 3?
sum(z<3) #How many of the vector's elements fulfill the condition that z is
 less than 3?
z[z==5]

#**********************************************************************

#CLASSES

y<-1:4
y
m<-cbind(y,y+1)
m
class(y)
class(m)
is.vector(y) #Is y a vector?
is.matrix(m) #Is m a matrix?
is.numeric(m) #Is the matrix m a numeric matrix or a character matrix?
mypets<-c('dog','cat','bird','dog')
mypets
class(mypets)
yourpets<-c('moose','gorilla','cow','giraffe')
yourpets
ourpets<-cbind(mypets,yourpets)
```

```
ourpets
class(ourpets)
is.character(ourpets)
is.numeric(ourpets)
ourpets<-factor(ourpets) #How many distinct variables are there?
ourpets
class(ourpets)
unclass(ourpets) #Assigns integers to correspond to character data
ourpets

#We can turn objects into class data.frame, like a data set:

ourpets<-data.frame(ourpets)
ourpets
#Data frame is different from a matrix; specifically refers to a set of
 workable data and can contain more than one class (character or numeric,
 say), which matrices cannot.

#Let's create a data frame that tells us additional information about the
 pets' -- their current ages (which we will assume are y) and their ages a
 year from now (y+1). Remember we previously created m as a matrix of y and y
 +1.

petages<-data.frame(m)
petages
class(petsandages)
#Second column is just called V2 (default) because we didn't assign a name to
 the vector y+1

#Chnage the names of the columns to reflect the information they provide:
names(petages)<-c("Age.Now","Age.Next.Year")
petages

#You can create data frames by combining two matrices (character or numeric):
allpets<-data.frame(mypets,yourpets)
allpets
#This is the same as the previously-created dataframe ourpets


#There are additional ways to refer to columns in your data frame....

petages$Age.Now #Use $ to denote the name of a variable in your data set.
 Remember, all names in R are case sensitive!
attach(petages) #You're "attaching" the variable name in your data set to R in
 general; now you can refer to variables without specifying the data set the
 variable comes from. This can be very helpful when working with only one data
 set, but be careful when working with more than one data set at a time --
 it's easy to forget which variables you named what and which data sets each
```

```
 variable refers to.
Age.Now
detach(petages)
Age.Now #You'll see an error message

#EXERCISE: Create a data.frame out of the mypets data and the age data. Now
 assume that the numbers are the ages of these pets. Change the names of the
 columns to reflect this.

mypets<-data.frame(mypets)
mypets
mypetages<-cbind(mypets, petages)
mypetages
names(mypetages)<-c("mypets", "ages_mypets", "ages_mypetsnextyear")
mypetages

is.matrix(mypetages)
class(mypetages)
mypetages=="dog" #Which elements of this data frame are "dog"?
mypetages!="gorilla"
mypetages<2 #You'll see a message warning you that not all elements of the
 data frame respond to this command (characters, not numbers)
mypetages>2 | mypetages =="dog" #Say you want to pull out all the animals who
 are older than 2 AND all the dogs
mypetages

ls()

#**********************************************************************

#BIRTHDAY PROBLEM SIMULATION

#Create a vector with the numbers of the days of the year
bdays<-1:365

#Create another vector that randomly samples 20 of these birthdays, with
 replacement (meaning that you throw the possible birthday "back in the pot"
 -- because we're assuming birthdays are independent of one another)

our.bdays<-sample(bdays,20,replace=T)
our.bdays

#Any duplicates?
#Now let's assume there are 150 people in a room:

our.bdays<-sample(bdays,150,replace=T)
our.bdays
```

```
#There is an almost virtual certainty that there will be duplicate birthdays
 in a sample of 150 people

#But how can we get R to tell us for sure?

unique(our.bdays)
length(unique(our.bdays))

#This tells us how many unique birthdays were randomly selected from a sample
 of 20

#Is the number of unique birthdays less than 20? In other words, do two or
 more people share a birthday (i.e., there would only be 19 or 18 or however
 many unique birthdays selected)? True or false?

length(unique(our.bdays))<20

#Try again with 150 people:

our.bdays<-sample(bdays,150,replace=T)
our.bdays
length(unique(our.bdays))<150

#So how do we know the probability of two or more person's sharing the same
 birthday in a class of 20?
#We could run the simulation a hundred times and record our answers....

our.bdays<-sample(bdays,20,replace=T)
length(unique(our.bdays))<20

#Or we could have R organize all our attempts into a matrix for easy
 comprehension. Let's assume we went into 100 classrooms of 20 students each
 and asked for everyone's birthdays. What is the probability that, in a
 randomly-selected class of these 100 classes, two or more people will share a
 birthday?

#We already calculated the theoretical probability that two or more people
 will share a a birthday in a room of 20 people:
1-
 ((365/365)*(364/365)*(363/365)*(362/365)*(361/365)*(360/365)*(359/365)*(358/3
 65)*(357/365)*(356/365)*(355/365)*(354/365)*(352/365)*(351/365)*(350/365)*(34
 9/365)*(348/365)*(347/365)*(346/365)*(345/365))

#Let's see how our simulation compares. The following command will create a
 matrix of the results of all 100 trials:
many.bdays<-matrix(sample(bdays,2000,replace=T),nrow=20,ncol=100)
many.bdays
```

```
#Now let's see a matrix of the results that shows only the number of UNIQUE
 birthdays that occurred in each of the 100 different trials:
unique.bdays<-apply(many.bdays,2,unique)
unique.bdays

#In how many of those 100 trials did we find a class wherein the number of
 unique birthdays was less than 20? In other words, in how many of those 100
 trials did 2 or more people in a class of 20 share a birthday?
sapply(unique.bdays,length)<20

#Count up the number of "trues," i.e., the number of times that 2 or more
 people shared a birthday in a class of 20:
same.bday<-sum(sapply(unique.bdays,length)<20)

#Probability?
same.bday/100

#*****************************************************************************

#MONTY HALL PROBLEM SIMULATION

#Let's assign the doors numbers (1,2,3) rather than letters. The prize is
 randomly behind one of these doors, with equal probability. We randomly
 choose a door.

doors<-1:3
prize<-sample(doors,1)
prize
choice1<-sample(doors,1)
choice1

#If the prize is behind our chosen door, Monty opens one of the other 2,
 randomly choosing between them. Otherwise, Monty opens the door that we did
 not choose and that does not contain the prize.

opendoor<-ifelse(prize==choice1,sample(doors[-choice1],1),doors[-
 c(choice1,prize)])

#Let's examine the result in a nice format:

result<-c(choice1,opendoor,prize)
names(result)<-c("Choice","Open Door","Prize")
result

#If we stick with our chosen door, do we win?

prize==choice1
```

```
#If we switch, do we win?

choice2<-doors[-c(choice1,opendoor)]
prize==choice2

# If we randomly choose between switching and sticking, do we win? (You can
 run these lines multiple times and get different answers.)

choice2<-sample(doors[-opendoor],1)
prize==choice2

#If we stick with our chosen door, what's our estimated probability of
 winning?

prize<-sample(doors,10000,replace=T)
choice1<-sample(doors,10000,replace=T)
wins<-sum(choice1==prize)
wins/10000

#How about if we always switch?

opendoor<-NA
choice2<-NA
for(i in 1:10000) {
    opendoor[i]<-ifelse(prize[i]==choice1[i],
    sample(doors[-choice1[i]],1),
    doors[-c(choice1[i],prize[i])])

    choice2[i]<-doors[-c(choice1[i],opendoor[i])]
}
wins<-sum(choice2==prize)
wins/10000

# How about if we randomly choose between switching and sticking?

for(i in 1:10000) {
    opendoor[i]<-ifelse(prize[i]==choice1[i],
    sample(doors[-choice1[i]],1),
    doors[-c(choice1[i],prize[i])])

    choice2[i]<-sample(doors[-opendoor[i]],1)
}
wins<-sum(choice2==prize)
wins/10000
```