```r
#PO 841
#9/17/14
#Lab 2: Classes, tables, computing mean and variance

#CLASSES

x<-rep(18,4)
m<-cbind(x,x+1)
x
m
class(x)
class(m)
is.vector(x)
is.matrix(m)
is.numeric(m)
freshmen<-c('Claire','Joe','Jane','Sam')
freshmen
class(freshmen)
sophomores<-c('Jim','Sarah','Julie','John')
sophomores
students<-cbind(freshmen, sophomores)
students
class(students)
is.character(students)
is.numeric(students)

#We can turn objects into the class data.frame, like a data set:
#Let's recall our numeric matrix m, suppose that these are the ages of the
 students, then turn the matrix into a data frame:
ages<-data.frame(m)
ages
class(ages)

#Now let's change the column names to reflect what we want the numbers to
 mean:
names(ages)<-c("freshmen.ages","sophomores.ages")
ages

#Let's turn our matrix of student names into a data set:
students<-data.frame(students)
students

#Now let's combine our two data frames:

studentsandages<-data.frame(cbind(students,ages))
studentsandages

#There are additional way to refer to data.frame columns:
```

```
studentsandages$freshmen.ages
attach(studentsandages)
freshmen.ages
freshmen
detach(studentsandages)
freshmen.ages #Error message will tell you that referring just to the variable
 "age.as.freshman" without specifying the accompanying data set won't work

#We can also easily rearrange columns to present our data howevever we wish:

attach(studentsandages)
studentsandages<-data.frame(cbind(freshmen, freshmen.ages, sophomores,
 sophomores.ages))
studentsandages

#Exercise: Let's refer back to the pet example from the previous lab. Create a
 data frame that presents the type of pet and the age of that pet for the
 following animals: a 2-year-old dog, a 5-year-old cat, a 1-year-old lizard
 and a 7-year-old turtle.

pets<-data.frame((c('dog','cat','lizard','turtle')),c(2,5,1,7))
names(pets)<-c("pets","ages")
pets

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

#EXPECTATIONS

#Let's create a table with values of k and Px(k) in order to compute an
 expectation

x<-c(rep(5,5),rep(4,4),rep(3,3),rep(2,2),rep(1,1))
x
draws<-sample(x,size=1000,replace=T) #The "replace" command just means you
 throw the number "back in the pot" after you choose it, so that for every
 draw, each number still has an equal chance of being picked (even if it's
 been chosen already)
draws

tab1<-table(draws)
tab1
class(tab1)
names(tab1)
tab2<-prop.table(tab1) #Gives proportions, not frequencies
tab2

#Now, we want to create a table where one column is all possible values of X
```

```
(i.e., the k's), and the other column is the probability of each value of X,
or Px(k).

draws #Shows all draws. This won't work for our first column; there are
duplicate values

unique(draws) #Gives us all unique possible values for k.

#Now sort them in ascending order
sort(unique(draws))

#Create a data frame that displays the probability distribution of your draws:
the unique possible draws in ascending order along with how frequently each
value was drawn.

tab3<-data.frame(cbind(sort(unique(draws)),tab2))
tab3
names(tab3)<-c('k','Px.k') #(Naming the second column Px(k) won't work)
tab3
class(tab3)

#Let's manually compute the expected value:
tab3$k * tab3$Px.k
mu<-sum(tab3$k * tab3$Px.k)
mu

#Compare this manual computation to R's built-in formula:
mean(draws)

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

#VARIANCE AND STANDARD DEVIATION

#Manually compute the variance:
(tab3$k - mu)^2
((tab3$k - mu)^2) * tab3$Px.k
variance<-sum(((tab3$k - mu)^2) * tab3$Px.k)
variance

#Compute the variance using R's built-in formula:
var(draws)

#Why the slight difference?

#We manually calculated sigma-squared from our probability-distribution table,
but the actual size of our sample (# of draws, or n) wasn't accounted for
(remember, we use statistics to infer, from a sample, info about the
population at large). But R's var(x) function does account for the size of
```

```
our sample. R's function calculates s-squared, the SAMPLE VARIANCE, which is
an unbiased estimator of sigma-squared. This will be covered more in class,
but take brief note of the difference now.

#If we manually compute the sample variance and compare to var(draws), we get
 the same answer:

((length(draws))*(sum(((tab3$k - mu)^2) * tab3$Px.k))/(length(draws)-1))
var(draws)

#Now let's compute the standard deviation. We can compute it manually from our
 probability-distribution table:
sd<-sqrt(sum(((tab3$k - mu)^2) * tab3$Px.k))
sd
#OR
sd<-sqrt(variance)
sd

#Or we can use R's built-in formula:
sd(draws)
#OR
sqrt(var(draws))

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

#HOMEWORK 1: SOLUTION REVIEW

#a.) Create a vector of exam grades
grades1<-c(75,90,65,83,78,87)
grades1

#b.) Replace the grade of 65 with a 70
grades1[grades1==65]<-70
grades1

#c.) Inflate each grade by 10%
grades1<-grades1 + (grades1*.10)
grades1

#d.) Randomly assign grades between 90-100
grades2<-sample(c(90:100),6,replace=TRUE)
grades2

#e.) How many grades are above 97?
length(grades2[grades2>=97])

#f.) Create a matrix of exam grades from both tests
allgrades<-matrix(cbind(grades1,grades2),ncol=2,nrow=6)
```

```
allgrades

#g.) Calculate Student #3's average grade
student3avg<-(allgrades[3,1]+allgrades[3,2])/2
student3avg
#OR
student3avg<-mean(allgrades[3,])
student3avg
```