Transition to R Class 3: Basic functions for descriptive statistics and summarizing data
Use simple functions and Xapply functions for summarizing and describing data
Goals:
(1) Summarizing continuous data in vectors
(2) Creating your own customized complex functions
(3) Summarizing continuous data in groups (by, aggregate, tapply)
(4) Applying functions across rows or columns (apply, sapply, lapply)

Sample data sets:
#**ferpcom-** data from the UCSC Forest Ecology Research Plot

```
ferpcom<-
read.table("http://people.ucsc.edu/~ggilbert/Rclass_docs/FERP07data.csv",sep=
",",header=TRUE)
head(ferpcom)
```

```
  tag   code     date  dbh east north stems status07          BA near_dist near_east near_north utm_east utm_north
1   2 QUERPA 8-Dec-06   31  1.9   6.3     1          0.000754768       6.6         0          0 417698.1   4096655
2   3 PSEUME 8-Dec-06  378  0.6   6.2     1          0.112220831       6.2         0          0 417696.8   4096656
3   4 QUERPA 8-Dec-06   20  0.5   6.8     1          0.000314159       6.9         0          0 417696.9   4096656
4   5 SEQUSE 8-Dec-06 1420  3.1  13.7     1          1.583676857       7.0         0         20 417701.2   4096662
5   6 QUERPA 8-Dec-06   74  4.7  19.2     1          0.004300840       4.7         0         20 417704.1   4096667
6   8 LITHDE 8-Dec-06   31  5.6  14.8     1          0.000754768       7.7         0         20 417703.9   4096663
```

#**allom-** height x dbh relationships for tree species on the UCSC FERP

```
allom<-
read.table("http://people.ucsc.edu/~ggilbert/Rclass_docs/sometreeallometries.
csv",sep=",",header=TRUE)
head(allom)
```

```
    code  tag height dbh
1 ACERMA  465   2.55  15
2 ACERMA 6468  17.00 532
3 ACERMA 6505  15.80 438
4 ARBUME  192  13.10 177
5 ARBUME  339   3.62  39
6 ARBUME  499   9.83 126
```

#**a-** a vector of 100 random number from a normal distribution

```
a<-rnorm(100, mean= 32, sd=6)
```

General structure of functions in R is: `function(object,modifiers)`

```
mean(a)  #returns the mean value directly
[1] 31.37834
MeanOfA<-mean(a)  #assigns the mean value to a variable called MeanOfA
MeanOfA  #show the value of that variable
[1] 31.37834
quantile(a) #gives estimated quartiles
0%      25%      50%      75%     100%
15.46291 28.04012 31.24346 35.30932 50.00404
?quantile   #check out help to see the structure of the function
# quantile(x, probs = seq(0, 1, 0.25), na.rm = FALSE,
         names = TRUE, type = 7, ...)
#define which quartiles are wanted, and the type of underlying distribution
quantile(a,probs=c(0.1,0.2,0.5,0.8,0.9),type=4)
10%      20%      50%      80%      90%
23.07496 25.03319 31.20045 36.32380 39.43275
```

Brief cheat sheet of major functions covered here

Mean of a vector	mean(data)
Number of observations	length(data)
Number of missing data (NA)	length(which(is.na(data)))
Standard Deviation of a vector	sd(data)
Variance of a vector	var(data)
Standard error of a vector	sd(data)/sqrt(length(data))
Median of a vector	median(data)
Min and Max of a vector	range(data) #or min(data) and max(data)
Quartiles, individually defined	quantile(data,probs=c(0,.25,.50,.75,1))
Quantiles, defined by sequence	quantile(data,probs=seq(from=0,to=1,by=.2))
Quartiles and mean	summary(data)
Shapiro-Wilk normality test	shapiro.test(data)
histogram of data	hist(data)
Normal Q-Q plot of data	qqnorm(data)
Unique levels of a vector	unique(data)
Number of unique levels of a vector	length(unique(data))
by – to do calculations by group	by(data=a$dbh,INDICES=a$code,FUN=mean)
aggregate – to do calculations by group	aggregate(x=a$dbh,by=list(a$code),FUN=mean)
tapply – to do calculations by group	tapply(X=a$dbh,INDEX=a$code,FUN=mean)
sapply – apply same function to all columns	sapply(X=com, FUN=sum)
lapply – apply same function to all columns	sapply(X=com, FUN=sum)
apply – apply same function to all columns	apply(X=com,MARGIN=2,FUN=sum)
apply – apply same function to all rows	apply(X=com,MARGIN=1,FUN=sum)
functions – write your own	myFunction<-function(arguments) {what to do}

Summary of functions used to get summary data grouped by some grouping variable

	Input	Function	Output
by	Data frame	Any	List
aggregate	Data frame	Scalar returns	Data frame
tapply	List, Data frame	Any	Array, List
table	Factors	Counts only	Table

Summary of apply, sapply, and lapply

	Margin	Applied To	Simplify	Output
apply	MARGIN=1	rows	-	Array
apply	MARGIN=2	columns	-	Array
sapply	-	columns	TRUE	Array
sapply	-	columns	FALSE	List
lapply	-	columns	-	List

Basic summary statistics on a vector or a single column of a data frame

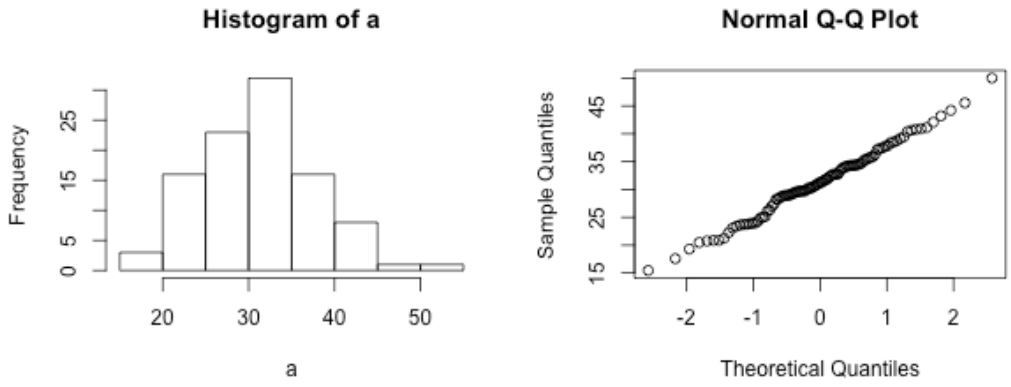
```
#a - create a vector of 100 random number from a normal distribution
a<-rnorm(100, mean= 32, sd=6)

#try these functions for basic descriptive stats:
#you can run them as a block to see what they give you
str(a) # see the structure of the object a
length(a) # get the number of elements in vector a
mean(a) # mean of vector a
median(a) # median of vector a
sd(a) #standard deviation of vector a
sd(a)/sqrt(length(a)) #standard error of vector a
range(a) #gives min and max of vector a - see also min(a) max(a)
quantile(a) #gives 0, 25%, 50%, 75%, 100% estimated quartiles
quantile(a, prob=c(0.1,0.5, 0.9)) #gives estimated quantiles at given
probabilities
summary(a) #gives quartiles and mean
shapiro.test(a) #test of normality for data

#you can also assign the output to an object to call up later
aq<-quantile(a); str(aq); aq[3]; aq["50%"] # dissect out results of quantile

summary_a<-summary(a) #assign the output of summary to object summary_a
summary_a
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  15.46   28.04   31.24   31.38   35.31   50.00
summary_a[2] #call up just the 1st quartile
1st Qu.
  28.04
summary_a[c("Min.","Max.")]  #call up just the min and max from summary
 Min.  Max.
15.46 50.00

hist(a)  #basic histogram of data in vector a in quartz window
qqnorm(a) #normal quantile-quantile plot of data in vector a
```



Create your own customized complex summary statistics function

You can create complex functions that contain various other functions, and then apply your function to a data set. Here we create a complex function MyStats that calculates a variety of different summary statistics on a vector, then combines them into a single object.

To create a function, use this structure:

```
NameOfMyFunction<-function(dataobject)
{
a<-function1(dataobject) #do something to your dataobject
b<-function2(dataobject) #do something else
a*b #the result of the last line is what is returned by the function
}
```

#note that dataobject is a placeholder name – it takes on the value of whatever you put in the ()

```
#create your own summary stats output table
myStats<-function(mydata)
{
mean<-mean(mydata)
sd<-sd(mydata)
se<-sd(mydata)/sqrt(length(mydata))
n<-length(mydata)
med<-median(mydata)
q<-quantile(mydata,prob=c(0.025,0.975))
sw<-shapiro.test(mydata)
aa<-as.table(c(mean,sd,se,n,med,q[1],q[2],sw$statistic,sw$p.value))
names(aa)<-
c("mean","sd","se","n","median","l95CI","u95CI","W","probW")
aa
      }

#then to use it to get summary stats of vector a from above
myStats(a)

#or to apply it to a column from a data frame (e.g., height in allom)
myStats(allom$height)

#if you want to control the number of digits printed try
round(myStats(allom$height),4)
```

```
    mean       sd       se        n   median    l95CI    u95CI        W    probW
 10.0101   8.7216   0.7398 139.0000   6.1600   2.2790  29.9650   0.8017   0.0000
```

Summarizing your data by groups
tapply, aggregate, by

There are several ways to calculated summary data grouped by some indicator variable.

```
#read in allom, which includes height and dbh data for tree species on the UCSC FERP
allom<-
read.table("http://people.ucsc.edu/~ggilbert/Rclass_docs/sometreeallometries.
csv",sep=",",header=TRUE)
```

1. First, check what the structure and contents of your file

```
str(allom) #check the dimensions, and names and types of variables
head(allom,3) # look at the first three lines of data
```

```
> str(allom)
'data.frame':     139 obs. of  4 variables:
 $ code  : Factor w/ 4 levels "ACERMA","ARBUME",..: 1 1 1 2 2 2 2 2 2 2 ...
 $ tag   : int  465 6468 6505 192 339 499 743 832 910 1134 ...
 $ height: num  2.55 17 15.8 13.1 3.62 9.83 40.6 4.81 30.1 6.21 ...
 $ dbh   : int  15 532 438 177 39 126 835 36 926 80 ...
> head(allom,3)
    code  tag height dbh
1 ACERMA  465   2.55  15
2 ACERMA 6468  17.00 532
3 ACERMA 6505  15.80 438
```

2. How many and which species (code) are in the file?

```
unique(allom$code) #get all the distinct values of code
length(unique(allom$code)) #count how many distinct values
> unique(allom$code)
[1] ACERMA ARBUME ARCTCR LITHDE
Levels: ACERMA ARBUME ARCTCR LITHDE
> length(unique(allom$code))
[1] 4

#or, more simply, use the table function
table(allom$code)
ACERMA ARBUME ARCTCR LITHDE
     3     49     38     49
```

3. What is the median dbh of each species code separately?

```
median(allom$dbh) #gives the median dbh across all four species.
> median(allom$dbh)
[1] 78
```

But we want something like this:

```
  code median
ACERMA  15.80
ARBUME  12.22
ARCTCR   4.25
LITHDE   7.21
```

There are three main ways to do this (by, aggregate, tapply), each with a different kind of output.

4. Use "by" to calculate medians by group (output object is a List)
General structure: by(data,INDICES,FUN) where
data – the column of numerical data
INDICES – the column of group names
FUN – what to calculate

```
by(data=allom$dbh,INDICES=allom$code,FUN=median)
```

OR

```
by(allom$dbh,allom$code,median)  #must keep arguments in the expected order

allom$code: ACERMA
[1] 438
------------------------------------------------------------
allom$code: ARBUME
[1] 222
------------------------------------------------------------
allom$code: ARCTCR
[1] 63
------------------------------------------------------------
allom$code: LITHDE
[1] 74

byout<-by(allom$dbh,allom$code,median)
byout[2]   #or byout["ARBUME"]
ARBUME
   222
```

#note: if you use a function that has options, simply tag them on after a comma

```
by(a$dbh, a$code, quantile, probs=c(0, .2, .4, .6))
```

5. Use "aggregate" to calculate means by group (output object is data frame)
General structure: aggregate(data,by,FUN)
data – the column of numerical data
by – a list of grouping elements; must specify it is a list
FUN – what to calculate

```
aggregate(x=allom$dbh,by=list(allom$code),FUN=median)
  Group.1   x
1  ACERMA 438
2  ARBUME 222
3  ARCTCR  63
4  LITHDE  74

agout<-aggregate(x=allom$dbh,by=list(allom$code),FUN=median)
names(agout)<-c("species","median")  #add pretty names
agout
  species median
1  ACERMA    438
2  ARBUME    222
3  ARCTCR     63
4  LITHDE     74
#or, to add the names to aggregate output in one step, use the setNames function
agout<-
setNames(aggregate(x=allom$dbh,by=list(allom$code),FUN=median),c("species","median"))
```

6. Use "tapply" to calculate means by group (output as array or list)
General structure: tapply(X,INDEX,FUN,simplify)
X – the vector or column of numerical data
INDEX – a list of grouping elements; do not need to specify it is a list
FUN – what to calculate
simplify – TRUE (output as array) or FALSE (output as list)

```
tapply(X=allom$dbh,INDEX=allom$code,FUN=median)
ACERMA ARBUME ARCTCR LITHDE
   438    222     63     74

taout<-tapply(allom$dbh,allom$code,median);  taout["ARCTCR"]
ARCTCR
    63
```

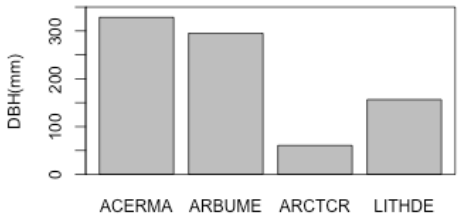
Note: if you have two grouping variables, you can get a 2-way summary table like this

```
#add a random grouping column (site) to allom
allom$plot<-round(runif(139,0,1),0); allom$site<-"B";
allom$site[allom$plot==0]<-"A"; allom<-allom[,-5];
head(allom)
    code  tag height dbh site
1 ACERMA  465   2.55  15    B
2 ACERMA 6468  17.00 532    B
3 ACERMA 6505  15.80 438    A
4 ARBUME  192  13.10 177    A  etc.

tall<-tapply(allom$dbh,list(allom$site,allom$code),median); tall
  ACERMA ARBUME ARCTCR LITHDE
A  273.5  343.0   67.5     79
B  438.0  121.5   58.0     72
Tall["A","ARCTCR"]  # this shows just the ARCTCR in site A
```

7. What if you want mean, standard deviation, and sample size for each group?

```
#You can calculate each statistic separately, putting each into a vector to hold it
#then bind them together into a single array.
avg<-tapply(allom$dbh,allom$code,mean)
stdev<-tapply(allom$dbh,allom$code,sd)
n<-tapply(allom$dbh,allom$code,length)
s_allom<-cbind(avg,stdev,n)
barplot(s_allom[,1],ylab="DBH(mm)");box()
s_allom
            avg     stdev  n
ACERMA 328.3333 275.39487  3
ARBUME 295.3265 257.60575 49
ARCTCR  60.5000  26.54191 38
LITHDE 156.5714 195.13884 49
#you can access array elements using s_allom[,"median]
s_allom<-as.data.frame(cbind(avg,stdev,n)) #or create a data frame like this
write.table(s_allom,"AllometrySummary.txt",sep="\t",col.names=NA,quote=FALSE)
#Saves the data frame to a .txt file to open in Excel or Word
#or check out the doBy library to automate a lot of this ☺
```



Applying Functions Across Multiple Rows Or Multiple Columns (margins) of an Array apply, sapply, lapply

#we will use an abbreviated community matrix (fc) from the ferpcom data from the UCSC FERP

```
ferpcom<-
read.table("http://people.ucsc.edu/~ggilbert/Rclass_docs/FERP07data.csv",sep=
",",header=TRUE);
ferpcom$quadrat<-
as.factor(paste(ferpcom$near_east,ferpcom$near_north,sep="_"))
ferpsm<-
ferpcom[ferpcom$code%in%c("ARBUME","LITHDE","QUERAG","QUERPA","PSEUME"),]
fc<-table(ferpsm$quadrat,ferpsm$code)
fc<-as.data.frame(fc[,c("ARBUME","LITHDE","QUERAG","QUERPA","PSEUME")])

head(fc)
        ARBUME LITHDE PSEUME QUERAG QUERPA
  0_0        0      0      2      0      3
  0_100      4      1      1      3     14
  0_120      3     35      4      2      2
  0_140      1      4      7      5      3
  0_160      4      2      3      7      0
  0_180      3      1     12      6      2
```

1. Use apply to summarize data either across rows or columns of a data frame

The apply function has an extra argument called margin.
If margin = 1 you apply the function to each row
If margin = 2 you apply the function to each columns.

```
apply(X=fc,MARGIN=2,FUN=sum)  #find the sum of each column in "fc"
ARBUME LITHDE PSEUME QUERAG QUERPA
   687   1258   2162    919   1188

apply(X=fc,MARGIN=1,FUN=sum) #find the sum of each row in "fc"
  0_0   0_100   0_120   0_140   0_160   0_180    0_20   0_200
    5      23      46      20      16      24      18      15
0_220   0_240   0_260   0_280   0_300    0_40    0_60    0_80
    6      22      37      19      11      31      32      19
```

etc.

2. Use sapply to use the same function across multiple columns (output is an array)

```
sapply is a wrapper for apply(X, MARGIN=2,FUN)
sapf<-sapply(X=fc,FUN=sum)
sapf
ARBUME LITHDE PSEUME QUERAG QUERPA
   687   1258   2162    919   1188
```

3. Use lapply to use the same function across multiple columns (output is a list)

```
lapply(X=fc,FUN=sum)
$ARBUME
[1] 687

$LITHDE
[1] 1258   etc.
```