Analyzing and Plotting Data

## Learning Objectives

* understand how to deal with missing data
* being able to generate summary statistics from the data
* calculate basic statistics across levels of a factor (here species)
* generate plots from the data using the average weights of the animals as an example

*If you have not already downloaded the* [*surveys*](http://emudrak.github.io/2015-01-15-cornell/data/biology/surveys.csv) *data and put it in directory "data" within your working directory, please do so now.*

# Calculating statistics

Let's get a closer look at our data. For instance, we might want to know how many animals we trapped in each plot, or how many of each species were caught.

The function table(), tells us how many of each species we have:

table(surveys$species)

R has a lot of built in statistical functions, like mean(), median(), max(), min(), var(), sd(). Let's start by calculating the average weight of all the animals using the function mean():

mean(surveys$wgt)

[1] NA

Hmm, we just get NA. That's because we don't have the weight for every animal and missing data is recorded as NA. By default, all R functions operating on a vector that contains missing data will return NA. It's a way to make sure that users know they have missing data, and make a conscious decision on how to deal with it.

When dealing with simple statistics like the mean, the easiest way to ignore NA (the missing data) is to use na.rm=TRUE (rm stands for remove):

mean(surveys$wgt, na.rm=TRUE)

[1] 42.67243

In some cases, it might be useful to remove the missing data from the vector. For this purpose, R comes with the function na.omit:

wgt\_noNA <- na.omit(surveys$wgt)

For some applications, it's useful to keep all observations, for others, it might be best to remove all observations that contain missing data. The function complete.cases() removes any rows that contain at least one missing observation:

surveys\_complete <- surveys[complete.cases(surveys), ]

**Note**: What this is actually doing is rather interesting. The simple line

complete.cases(surveys)

returns a list of TRUE/FALSE values for each row of surveys, showing whether or not it is a complete case. When you put such a list into the brackets [,] to index a matrix, it returns the rows that are listed as TRUE.

The new data.frame surveys\_complete now has only complete data. Lets see how many observations of each species was retained:

table(surveys\_complete$species)

Many species have 0 values in this table, meaning there are no records of this species left (i.e. none had the weight recorded). Yet they still show up in this table. This is because R "remembers" all species that were found in the original dataset, even though they aren't included in this data set. This could get annoying later on (with plotting, etc...).

To remove the NA and make things clearer, we can redefine the levels for the factor "species" by re-establishing this as a factor.

surveys\_complete$species <- factor(surveys\_complete$species)

An equivalent way to do this is to use the droplevels() function. This will work if you have many columns from the same data frame that need excess factors removed.

surveys\_complete <- droplevels(surveys\_complete)

Now see how things are looking:

table(surveys\_complete$species)

## PAUSE

Make sure everyone has a clean copy of surveys\_complete.

If you have been lost/confused/in the bathroom, you can catch up now with:

##CLEAR WORKPACE????  
  
#surveys <- read.csv('data/surveys.csv')  
surveys$logwgt <- log(surveys$wgt)  
surveys$plot <- as.factor(surveys$plot)  
surveys\_complete <- surveys[complete.cases(surveys), ]  
surveys\_complete <- droplevels(surveys\_complete)

**Now save the script!**

### Challenge

1. To determine the number of elements found in a vector, we can use use the function length() (e.g., length(surveys$wgt)). Using length(), how many animals have not had their weights recorded?
2. What is the range (minimum and maximum) weight?
3. What is the median weight for the males?

# Statistics across factor levels

What if we want the maximum weight for each species, or the mean for each plot?

R comes with convenient functions to do this kind of operations, functions in the apply family.

For instance, tapply() allows us to repeat a function across each level of a factor. The format is:

tapply(columns\_to\_do\_the\_calculations\_on, factor\_to\_sort\_on, function)

If we want to calculate the maximum for each species (using the complete dataset):

tapply(surveys\_complete$wgt, surveys\_complete$species, max)

If we want to calculate the mean for each plot:

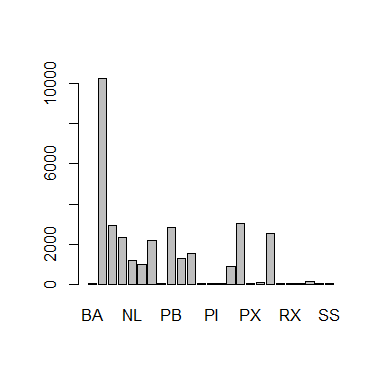
tapply(surveys\_complete$wgt, surveys\_complete$plot, mean)

# Creating a barplot

The best way to develop insight is often to visualize data. Visualization deserves an entire lecture (or course) of its own, but we can explore a few features of R's base plotting package.

Let's use the surveys\_summary data that we generated and plot it. R has built in plotting functions.

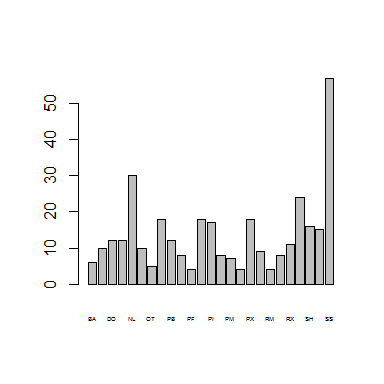
barplot(table(surveys\_complete$species))



The axis labels are too big though, so you can't see them all. Let's change that.

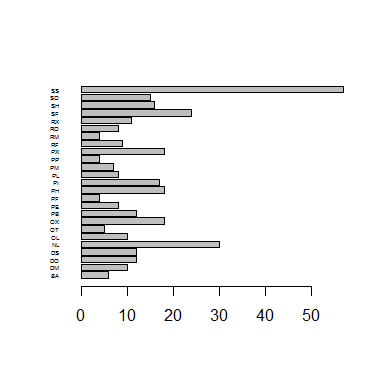
**Tip:** To learn about a function in R, e.g. barplot, we can read its help documention by running help(barplot) or ?barplot.

barplot(surveys\_summary$mean\_wgt, cex.names=0.4)



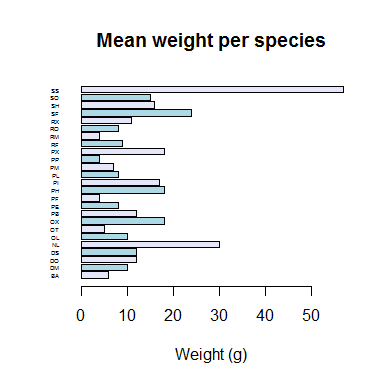
Alternatively, we may want to flip the axes to have more room for the species names:

barplot(surveys\_summary$mean\_wgt, horiz=TRUE, las=1, cex.names=0.4)



Let's also add some colors, and add a main title, label the axis:

barplot(surveys\_summary$mean\_wgt, horiz=TRUE, las=1, cex.names=0.4,  
 col=c("lavender", "lightblue"), xlab="Weight (g)",  
 main="Mean weight per species")



### Challenge

1. Create a new plot showing the standard deviation for each species. Choose one or more colors from [here](http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf). (If you prefer, you can also specify colors using their hexadecimal values #RRGGBB.)

## More about plotting

There are lots of different ways to plot things. You can do plot(object) for most classes included in R base. To explore some of the possibilities:

?barplot  
?boxplot  
?plot.default  
example(barplot)

If you wanted to output this plot to a pdf file rather than to the screen, you can specify where you want the plot to go with the pdf() function. If you wanted it to be a JPG, you would use the function jpeg() (other formats available: svg, png, ps).

Be sure to add dev.off() at the end to finalize the file. For pdf(), you can create multiple pages to your file, by generating multiple plots before calling dev.off().

pdf("mean\_per\_species.pdf")  
barplot(surveys\_summary$mean\_wgt, horiz=TRUE, las=1,  
 col=c("lavender", "lightblue"), xlab="Weight (g)",  
 main="Mean weight per species")  
dev.off()

#### Key Points

* Use variable <- value to assign a value to a variable in order to record it in memory.
* Objects are created on demand whenever a value is assigned to them.
* The function dim gives the dimensions of a data frame.
* Use object[x, y] to select a single element from a data frame.
* Use from:to to specify a sequence that includes the indices from from to to.
* All the indexing and slicing that works on data frames also works on vectors.
* Use # to add comments to programs.
* Use mean, max, min and sd to calculate simple statistics.
* Use apply to calculate statistics across the rows or columns of a data frame.
* Use plot to create simple visualizations.