Tutorial: Libraries, Plotting, and Data

BIOL 414/514

Quantitative Analysis of Vertebrate Populations

Learning Objectives:

1. Learn to reorganize data, summarize, and extract pieces of dataframes
2. Learn about plotting

Lab adapted from labs by Dr. Nicholas Nagel at the University of Tennessee

R libraries

There must be tens of thousands of different functions in R now, covering over a hundred different fields. More than you could possibly ever use. To keep all this straight, there is "base" R, and then over a thousand different add-on packages. Most people will only ever use a few.

The package we will use the most is called `ggplot2.` Another package that we will use in this lab is called `dplyr.`

To load packages, use the `library` function.

Below is an example of a code "chunk" in Rmarkdown. Try executing this chunk by either (1) clicking the *Run* button within the chunk (little green arrow on the right), (2) Selecting Run on the top menu, or (3) by placing your cursor inside the chunk and pressing *Cmd+Shift+Enter*.

```
library(ggplot2)
```

On a classroom comupter this might work if the packages are already installed. If you're on your personal computer or it hasn't been installed on the lab computer, chances are high that you get the message, `Error in library(ggplot2) : there is no package called 'ggplot2'`.

This means that the package `ggplot2` is not in your computer's library. You may have to install it first. This is how to install it and lme4.

```
install.packages("dplyr")
install.packages("ggplot2")
```

After they are installed, you can go back and load the libraries with the command:

```
library(lme4)
library(ggplot2)
library(dplyr)
```

You only need to install a package once. But everytime you fire up R, you will have to use the library command to load the package into R so that you can use it. Think of R as a desk in your bedroom where you do your studying. Before you can study, you will need to go to your `library` and get the books you need (you have your own library at home, right?). And if the library doesn't have the books, then you will have to ask the library to go out and get the books (with `install.packages`).

Please note, while you can run R without internet access, you will need to be online when you run install.packages. **Pro Tip:** If you are taking a plane trip and expect to get work done, then always make sure that you have installed the packages you will need before you take off.

Getting the data

Data are usually obtained from a file that someone gives you, or that you make yourself, or that you download from the web. Some data come installed with base R and others come as example datasets with particular packages. Download the `Sparrows.txt` file from Canvas and put it in a single folder containing your R Project and Rmd file (if can be in a subfolder within that folder such as `Data/Sparrows.txt`). Use the following code to read in the text file and look at the file structure. You can also click it in your "Environment" to the right in to open it with the RStudio Viewer.

```
sparrows <- read.table("Data/Sparrows.txt", sep = "",
    header = TRUE, stringsAsFactors = FALSE)
str(sparrows)
```

```
## 'data.frame':    1281 obs. of  12 variables:
##  $ Speciescode: int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Sex        : int  0 0 0 0 0 0 5 0 0 5 ...
##  $ SexNew     : int  1 1 1 1 1 1 6 1 1 6 ...
##  $ wingcrd    : num  59 55 53.5 55 52.5 57.5 53 55 55 55.5 ...
##  $ flatwing   : num  60 56 54.5 56 53.5 59 54 56 56 56.5 ...
##  $ tarsus     : num  22.3 19.7 20.8 20.3 20.8 21.5 20.6 21.5 20.8 20.5 ...
##  $ head       : num  31.2 30.4 30.6 30.3 30.3 30.8 32.5 31.2 31.6 31.4 ...
##  $ culmen     : num  12.3 12.1 12.8 11.9 12.6 12 13.5 12.3 13.2 13.2 ...
##  $ nalospi    : num  13 8.3 8.6 8.7 8.8 8.1 10.7 8.7 9.1 10.5 ...
##  $ wt         : num  9.5 13.8 14.8 15.2 15.5 15.6 15.6 15.7 15.7 15.7 ...
##  $ observer   : int  2 8 7 3 3 2 3 5 2 6 ...
```

```
##  $ Age         : int  2 2 2 1 1 2 1 1 1 1 ...
```

Let's also look at the first 10 rows of data.[1]

[1] You can use the `kable` package to make tables look pretty. Click here for tutorial

```
head(sparrows, 10)
```

Speciescode	Sex	SexNew	wingcrd	flatwing	tarsus	head	culmen	nalospi	wt	observer	Age
1	0	1	59.0	60.0	22.3	31.2	12.3	13.0	9.5	2	2
1	0	1	55.0	56.0	19.7	30.4	12.1	8.3	13.8	8	2
1	0	1	53.5	54.5	20.8	30.6	12.8	8.6	14.8	7	2
1	0	1	55.0	56.0	20.3	30.3	11.9	8.7	15.2	3	1
1	0	1	52.5	53.5	20.8	30.3	12.6	8.8	15.5	3	1
1	0	1	57.5	59.0	21.5	30.8	12.0	8.1	15.6	2	2
1	5	6	53.0	54.0	20.6	32.5	13.5	10.7	15.6	3	1
1	0	1	55.0	56.0	21.5	31.2	12.3	8.7	15.7	5	1
1	0	1	55.0	56.0	20.8	31.6	13.2	9.1	15.7	2	1
1	5	6	55.5	56.5	20.5	31.4	13.2	10.5	15.7	6	1

```
# sparrows[1:10, ] # also prints the first 10
# rows and each column
```

Or we can just look at the fist 12 rows of only the sex, wingcrd, and wt columns.

```
sparrows[1:12, c("Sex", "wingcrd", "wt")]
```

Sex	wingcrd	wt
0	59.0	9.5
0	55.0	13.8
0	53.5	14.8
0	55.0	15.2
0	52.5	15.5
0	57.5	15.6
5	53.0	15.6
0	55.0	15.7
0	55.0	15.7
5	55.5	15.7
0	57.0	15.8
0	55.0	15.8

```
sparrows_subset <- sparrows[, c("Sex", "wingcrd",
    "wt")]
str(sparrows_subset)
```

```
## 'data.frame':    1281 obs. of  3 variables:
##  $ Sex    : int  0 0 0 0 0 0 5 0 0 5 ...
##  $ wingcrd: num  59 55 53.5 55 52.5 57.5 53 55 55 55.5 ...
##  $ wt     : num  9.5 13.8 14.8 15.2 15.5 15.6 15.6 15.7 15.7 15.7 ...
```

```
mean(sparrows_subset$wt)
```

```
## [1] 19.79844
```

```
mean(sparrows_subset[, "wt"])
```

```
## [1] 19.79844
```

or print the first 10 rows of just the wingcrd using the `$` to call that column

```
head(sparrows$wingcrd, 10)
```

```
##  [1] 59.0 55.0 53.5 55.0 52.5 57.5 53.0 55.0
##  [9] 55.0 55.5
```

If you save a MS Excel file as a comma separated csv file, you will use a similar command but with a comma for the separator. You can try this with the `Salamander_Demographics.csv` file.

```
sally <- read.table("Data/Salamander_Demographics.csv",
    sep = ",", header = TRUE, stringsAsFactors = FALSE)
str(sally)
```

```
## 'data.frame':    3382 obs. of  20 variables:
##  $ line  : int  1861 1115 360 2897 1432 372 231 2739 2236 543 ...
##  $ page  : int  60 36 12 92 46 12 8 87 72 17 ...
##  $ dates : chr  "4/21/09" "9/9/08" "5/31/08" "5/7/11" ...
##  $ month : int  4 9 5 5 10 5 5 10 5 6 ...
##  $ day   : int  21 9 31 7 16 31 27 24 14 5 ...
##  $ year  : int  2009 2008 2008 2011 2008 2008 2008 2009 2009 2008 ...
##  $ time  : chr  "N" "N" "N" "N" ...
##  $ plot  : chr  "5" NA "3" "7" ...
##  $ mass  : num  0.427 0.633 0.639 0.921 0.943 ...
##  $ svl   : int  33 37 42 43 45 46 47 48 NA NA ...
##  $ tl    : int  63 68 63 79 74 NA 75 89 87 NA ...
##  $ sex   : chr  NA NA NA NA ...
##  $ gravid: chr  "N" "N" "N" "N" ...
##  $ group : chr  NA NA NA NA ...
##  $ clutch: int  NA NA NA NA NA NA NA NA NA NA ...
##  $ color : chr  "R" "R" "R" "R" ...
##  $ recap : chr  NA NA NA "N" ...
##  $ mark  : chr  NA NA NA NA ...
##  $ id    : int  1371 NA 187 2154 1042 198 74 2036 1564 351 ...
##  $ damage: chr  "N" "N" "Y" "N" ...
```

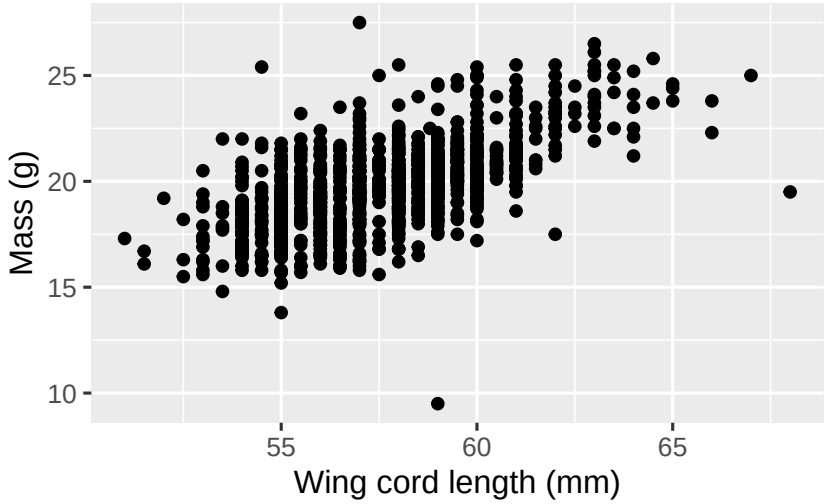
Plotting

Let's use the sparrow data because we are familiar with it. Let's pick a length measurement, wingcrd, and examine the relationship with mass (unfortunately named `wt` for weight).

It's always a good idea to plot your data to examine patterns. The x-axis should be `wingcrd`, and the y-axis should be `wt`. We usually use a scatter plot for plotting two continuous variables, and so let's use a point geometry in the ggplot2 package to show each observation.

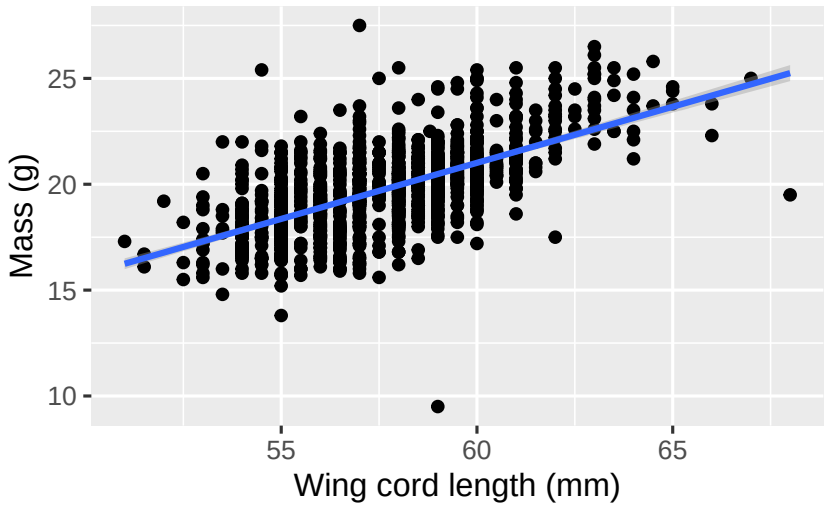
ggplot2 (see cheatsheet)

```
ggplot(sparrows, aes(x = wingcrd, y = wt)) + geom_point() +
    labs(x = "Wing cord length (mm)", y = "Mass (g)")
```



It looks like a line fits that fairly well. So first, let's go ahead and add a line to the plot using `geom_smooth`.

```
ggplot(sparrows, aes(x = wingcrd, y = wt)) + geom_point() +
    labs(x = "Wing cord length (mm)", y = "Mass (g)") +
    geom_smooth(method = "lm")
```



You see the regression line in blue, but a grey band around the regression that represents our confidence in the regression line. We'll get that soon enough, but for now just recognize that we estimated this based on a sample, and so there is some uncertainty in our data, and thus, there is some uncertainty in our line. The grey band is an estimate of where the regression could be given this uncertainty. Visually, this looks like a decent regression.

In the next tutorial, we will learn about running more formal linear regressions and summarizing and interpreting the results.

dplyr (see cheatsheet)

The `dplyr` package is extremely powerful for reoganizing and manipulating data. The important basic functions within dplyr are `filter`, `select`, `mutate` and the pipe `%>%`. These allow us do the same things with dataframes that we did above but more efficiently and with more clear commands.

dplyr (see cheatsheet): `filter()`, `select()`, `mutate()` and the pipe `%>%`

`filter` is used for getting just the rows that mean particular conditions

`select` is for getting just the columns your want

`mutate` is used for creating new columns, often using calculations from other columns

`%>%` is called a pipe and allows us to string together multiple function calls

For example we could use the following functions to create a new dataframe with the features we want with the following code:

```
sparrows_mini <- sparrows %>%
  select(Speciescode, wingcrd, wt, Age) %>% # just select 4 columns
  mutate(wingcrd_cm = wingcrd / 10) # add column converting wingcord from mm to cm

dim(sparrows_mini)

## [1] 1281    5
```

```
summary(sparrows_mini)
```

```
##   Speciescode       wingcrd
##  Min.   :1.000   Min.   :51.00
##  1st Qu.:1.000   1st Qu.:56.00
##  Median :1.000   Median :58.00
##  Mean   :1.116   Mean   :57.71
##  3rd Qu.:1.000   3rd Qu.:59.00
##  Max.   :2.000   Max.   :68.00
##        wt            Age
##  Min.   : 9.5   Min.   :1.000
##  1st Qu.:18.5   1st Qu.:1.000
##  Median :19.9   Median :1.000
##  Mean   :19.8   Mean   :1.157
##  3rd Qu.:20.9   3rd Qu.:1.000
##  Max.   :27.5   Max.   :2.000
##    wingcrd_cm
##  Min.   :5.100
##  1st Qu.:5.600
##  Median :5.800
##  Mean   :5.771
##  3rd Qu.:5.900
##  Max.   :6.800
```

Booleans

Boolean - denoting a system of algebraic notation used to represent logical propositions, especially in computing and electronics.

Booleans are extremely useful for filtering and later you could use them for `ifelse` statements. For example if we want to just select Species 1 and individuals not equal to Age 2 with wing cord lengths greater than 6 cm and weight (mass) greater than or equal to 21.5 g.

```
sparrows_filtered <- sparrows_mini %>% filter(Speciescode ==
    1, Age != 2, wingcrd_cm > 6, wt >= 21.5)
```

```
sparrows_filtered
```

Speciescode	wingcrd	wt	Age	wingcrd_cm
1	60.5	21.5	1	6.05
1	61.0	21.5	1	6.10
1	61.0	21.5	1	6.10
1	61.0	21.5	1	6.10
1	60.5	21.6	1	6.05
1	62.0	21.7	1	6.20
1	61.0	21.7	1	6.10

Speciescode	wingcrd	wt	Age	wingcrd_cm
1	61.0	21.7	1	6.10
1	61.0	21.8	1	6.10
1	62.0	22.2	1	6.20

Commands learned in this lab

Base R

- `$`
- `mean( , na.rm=TRUE)`
- `save()`
- `write.csv()`
- `==, > >=, !=`

tidyverse (dplyr)

- `%>%`
- `filter()`
- `select()`
- `mutate()`

ggplot

- `ggplot()`
- `geom_point()`
- `geom_smooth(, method='lm')`