Ecological Analysis with R (Harvard Forest ***R***EU course)

[Why R?]

- Introduce yourselves (role at HF, experience with programming and analyses, fun fact)
- What to bring?
- Work with your mentors to develop good analyitcal plans

[Why R?]

R is obviously not the only option when it comes to programming languages. Ditto for statistical tools. In fact, everything you can do in R you can probably do in any number of other programs/languages/webapps/etc.

So why do we teach R?

[Why R?]

R is immediately useful. Once you know even a little bit it can save you time over programs like Excel and Google sheets. With only 2 lines of code we can create a dataset AND plot it. With three we can add custom text.

[Why R?]

```
fake_data <- rnorm(45,mean = 240, sd = 12)
hist(fake_data, col = "lightblue")
text(x = 242, y = 4, "How long would this take in Excel?" , font = 2,cex = 1.2,  col = "red")
```

[Why R?]

- R is growing in popularity - not just for statistics, it is being adopted for a variety of uses.
- This means that learning even just the basics can help you in many paths, not just if you end up teaching REUs at Harvard Forest ;)
- Pragmatically: Knowing R is an extremely marketable skill in research, finance, etc.
- This also means theres a growing community out there to help you - more on this in a bit.

[Why R?]

- R is open-source and free.
- This aligns with our philosophy of open access, documented science.
- Pragmatically: You and your collaborators will always have access to it regardless of funds and institutional affiliation.

[Why R?]

- http://shiny.rstudio.com/gallery/
- https://ecoapps.shinyapps.io/lvpredatorprey/
- http://dailynoise.blogspot.com/2012/10/fun-with-r-and-why-its-so-cool.html

Day 1

- [Getting setup]

- [R as a second language]
- [R Community = Packages + help?]
- [Coding basics]

[Getting setup]

- Install the newest version of **R** - from here
- Install the newest version of **RStudio** from here
  - You do not need RStudio to run R. We do suggest it however for the improvements it offers over the default R GUI (a more seamless integration of scripts, console and graphs) along with some other additional features.
  - While RStudio offers some point-and-click GUI alternatives, we will not be using or teaching them in this class.

[R as a second language]

Learning R (or any computer language) is similar to learning a foreign language.

- You have to learn more than just the words: you have to learn the grammar, punctuation and syntax of the language.
- Just like in a foreign language, R has many commands that are cognates of english words. This is helpful when learning, as you often can figure out what a function does from its name alone. *Though be wary, just like in human languages there are sometimes false cognates!!*

```
{r, eval = FALSE} #Can you guess what the following functions do:  mean(x) max(x) plot(x)
Map(x)
```

- Another benefit of learning R is that it can make learning other computer languages much easier. Once you know some of the concepts of programming picking up another language (especially some of the more similar ones) can be pretty quick!

[R Community = Packages + help?]

As we mentioned before one of the great things about R is the

```
community that comes with it. This is helpful in many ways, a few
of which we'll discuss a bit more in depth:
```

- Perhaps the most important resource out there to a new R user are packages. While base R comes with many handy functions you will quickly reach a point where you want to do something that the basic functions don't cover. Packages to the rescue! Many people have already poured hours into creating new functions that do many, many different things. They bundle to these functions into 'packages' that are then made freely available.

First we need to know the package name so we can install it:

```
install.packages("raster")
```

You only have to install a package once on your computer. Once it is installed we can then load it into any script in the future:

```
library("raster")
```

Once loaded, we can now use any of the functions from that package:

```
library("raster")
r <- raster(matrix(1:25,5,5))
plot(r)
```

This can save you from having to reinvent the wheel each time you need to do something new!

- Help literature: it can be daunting trying to remember what every function does and what input is needed to run that function? Your new best friend is this simple command: `?function.name` . Putting a `?` in front of any function and running that line will bring up the documentation for that function that should tell you everything you need to know to use it.
- Stack Overflow and various other online resources: There is already an abundance of information and tips about R online. Stack Overflow especially often has questions and answers that may help you figure out whats going on with your code. As R continues to grow in popularity, there are more and more specialized sites devoted to various disciplines that use R.

Day 2

- [Data entry]
- [Data manipulation]
- [Data Visualization]
- [Analysis overview]

[Coding basics]

```
4 + 8
```

Starting with simple math:

```
## [1] 12
```

```
10 / 2
```

```
## [1] 5
```

```
15 %% 4
```

```
## [1] 3
```

```
x <- 4
y <- 6
x + y
```

What are objects?

```
## [1] 10
```

```
z <- "Hello Harvard Forest"
print(z)
```

```
## [1] "Hello Harvard Forest"
```

```
v <- c(1,2,3,4,10)
sum(v)
```

What if we want to store more than one value at a time?

```
## [1] 20
```

```
v2 <- c(10,15,20,30,40)
v + v2
```

```
## [1] 11 17 23 34 50
```

[Analysis overview]

[Data entry]

The first thing we need to do for most analyses is get our data into R so that we can do things with it. One way to do this is to enter it by hand. We can do this in a few ways: * Create multiple objects with our data:

```
temperature <- c(70,80,50,53,60,45) # in Fahrenheit
precipitation <- c(.01,1,0,2,.8,1.5) # in inches
```

- Maybe more useful is to create a dataframe:

```r
weather <- data.frame(date = c("3/18","3/19","4/10","4/11","4/18","5/10"),temperature = c(70,80,50,53,6
weather
```

```
##   date temperature precipitation
## 1 3/18          70          0.01
## 2 3/19          80          1.00
## 3 4/10          50          0.00
## 4 4/11          53          2.00
## 5 4/18          60          0.80
## 6 5/10          45          1.50
```

From this point on we will be using the data we collected from you all.

[Data manipulation]

- Vectors - indexing
- Matrices - rows and column
- Data Frames - mixing data types
- Lists - applys

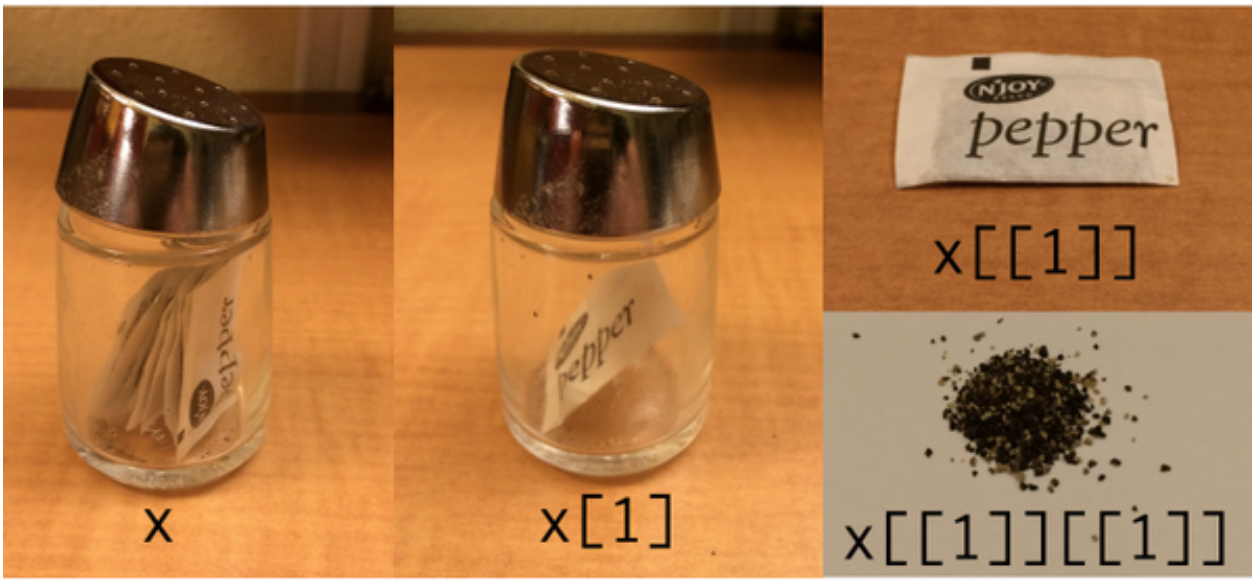

Figure 1: list example

[Data Visualization]

Day 3

- [Stats and science]
- [Results: organization and reporting]
- [Life-long lea**R**ning]

[**Stats and science**]

[**Results: organization and reporting**]

[**Life-long leaRning**]

- TBD: Optional Hacker Session 1 and 2 (e.g. GIS and mapping, iterative functions, accessing the Harvard Forest Archives, linear algebra, making Shiny or Leaflet apps)