EEB 177 Lecture 16

Thursday March 2nd, 2017

Topics

- lightning talk updates
- ggplot example

Hacking Sessions

Tuesday from 6-8:30 in LS 3209

Homework #7

Due Wednesday by 5:00 PM

Projects Deadline #3

Due Friday by 5:00 PM. To complete this assignment:

- Commit with data I/O and 2+ functions due Friday, Week 7: Feb 24 (5%) AND
- Markdown outline of the complete project
    - include pseudocode for portions you need to write
    - placeholders for figures
    - in ipython or Rmarkdown
    - any code that you do include should be well-commented
    - include references in the intro about evolutionary history or your research topic

```
find all parrotfish species
for each species
   extract diet
   extract range
```

Lightning talks

Lightning talks are meant to be short presentations about your final projects. Presentations must adhere to the following guidelines:

- 3 minutes max length
- should provide a short overview of the programming challenge
- must include active demonstration of code
- must show a user-generated plot of data

I will post a grading rubric to the course site tonight.

Grad students will give lightning talks on Thursday the 9th.

Undergrads will give lightning talks on the 14th and 16th.

R

R is a computing language for statistical analyses. R has grown in popularity over the last decade for several reasons

- it is fully programmable, leading to reproducible science
- it has powerful data visualization libraries
- it is free and open source

RStudio

RStudio is an integrated development environment (IDE) for R. RStudio makes it easier to write and run scripts, save graphics files, and create R-Markdown documents. RStudio runs **on top of R**--you need to have R installed to use RStudio.

RMarkdown

RMarkdown is an extension of markdown that includes syntax highlighting and code execution for R. We will use RMarkdown throughout our work in this class to create reproducible science.

Best Practices

We will follow some recommendations from a paper on best practices for projects.

- Put each project in its own directory, which is named after the project.
- Put text documents associated with the project in the doc directory.
- Put raw data and metadata in the data directory, and files generated during cleanup and analysis in a results directory.
- Put source for the projectÃ¢â‚¬â„¢s scripts and programs in the src directory, and programs brought in from elsewhere or compiled locally in the bin directory.
- Name all files to reflect their content or function.

Can you arrange your final projects folder on github to reflect these suggestions?

Data Structures in R

We are going to learn about basic data types and structures in R and we will take advantage of RMarkdown to make our investigation repeatable. First open up a text editor and write the following lines:

```
coat,weight,likes_string
calico,2.1,1
black,5.0,0
tabby,3.2,1
```

Save the file as `feline-data.csv` to a directory called `data`

Now open RStudio and select create a New Markdown document from the File, New File menu. RStudio will populate this document with example code to show how RMarkdown works. There are two main things to keep in mind.

1. Markdown syntax works within RMarkdown. You can create lists, headers, etc with it.
2. Codeblocks within RMarkdown can be specified to run R code when your document is compiles (*knit*).

Try knitting your RMarkdown document with the example code right now. You can select knit from the program menu or use a keyboard shortcut (**command-shift-k** on a mac). Notice that the R code blocks are run each time you knit the document.

R and markdown

You can include R chunks with three ticks like so

```{r name_of_chunk}
some r code

other r code

```

```
x <- rnorm(100)
y <- 2*x + rnorm(100)
y[1:10]
```

Notice that the code in your markdown is executed each time you knit. This is one way of creating **reproducible science!**

Chunk options

You can include commands between the curly braces to control R behavior. echo=FALSE hides the code but shows the output of the code.

```{r hidden_chunk echo=False}
x <- rnorm(100)
y <- 2*x + rnorm(100)
cor(x, y)
```

```
x <- rnorm(100)
y <- 2*x + rnorm(100)
cor(x, y)
```

Hide

Hide evaluates and shows the code but does not show results

```{r hidden_res, results="hide"}
x <- rnorm(100)
y <- 2*x + rnorm(100)
cor(x, y)
```

```
x <- rnorm(100)
y <- 2*x + rnorm(100)
cor(x, y)
```

Evaluate without showing

You can use *include = False* to evaluate code without showing it

```{r chunk_name3, include=FALSE}
x <- rnorm(100)
y <- 2*x + rnorm(100)
cor(x, y)
```

```
x <- rnorm(100)
y <- 2*x + rnorm(100)
cor(x, y)
```

Useful if you have code that needs to be executed to make a figure later in the document....

Figure options

You can control the size of figures with **fig.width** and **fig.height**

{r scatterplot, fig.width=8, fig.height=6}

```
plot(x,y)
```

to hide figures: **fig.show="hide"**

to hide results but show figure: **include=TRUE and results="hide"**

Global chunk options

knitr::opts_chunk$set lets you set the default for your code chunks

```
knitr::opts_chunk$set(fig.width=12, fig.height=8, fig.path
        echo=FALSE, warning=FALSE, message=FALSE)
```

This sets the default figure size and creates a path to save all figures.

inline code

One advantage of writing your reports in R is that values can be evaluated by your collaborators in the document. Consider:

```
samp = rnorm(100) + 0.2
```

Markdown allows you to write documents that are directly tied to your data. You might write The mean of my sample was `\ r mean(samp)` and I sampled a total of `\ r length(samp)` individuals.

Type this in to your RMarkdown document and knit it see the output.

tables

tables are kind of a pain in markdown but the kable package gives some options

```
n <- 100
x <- rnorm(n)
y <- 2*x + rnorm(n)
out <- lm(y ~ x)
library(knitr)
kable(summary(out)$coef, digits=2)
```

table with cars data

```
library(knitr)
kable(summary(cars), digits=2)
```

tables with pander

another table option (note you need to install pander first: **install.packages("pander"))**

```
n <- 100
x <- rnorm(n)
y <- 2*x + rnorm(n)
out <- lm(y ~ x)
library(pander)
panderOptions("digits", 2)
pander(out)
```