Module 7: R Markdown

BIOSTAT 213: Introduction to Computing in the R Software Environment

Yea-Hung Chen, PhD, MS

Wednesday, February 27, 2019

Introduction to R Markdown

Brief Exercise

R Markdown

The header

Formatting text

R code

Beyond this course

Introduction to R Markdown

Introduction to R Markdown

Brief Exercise

R Markdown

The header

Formatting text

R code

Beyond this course

R Markdown

- R Markdown is a "syntax" for generating reports and presentations that incorporate R code and output

R Markdown

- resources:
  - short introduction
  - cheat sheet
  - book

R Markdown

- R Markdown is based on Markdown, a syntax for formatting text
- the language allows for quick text-based indication of headers, bullets, bold face, italicization, etc
- for example, bold face is indicated by asterisks and bullet points are indicated by dashes:

```
Things I learned in **this class**:
- object types
- importing data
```

- the text file can then be converted to a more readable (by humans) file (an HTML file)

Module 7: R Markdown
Yea-Hung Chen, PhD, MS
Introduction to R Markdown
Brief Exercise
R Markdown
The header
Formatting text
R code
Beyond this course

R Markdown

- R Markdown adds the ability to add in R code and output
- it also supports various output files, including HTML, PDF, and Word documents
- it also can be used to create presentations (such as this one)

Brief Exercise

Brief Exercise

You may work in groups of 2–3.

1. In RStudio, generate a new R Markdown file: File > New File > R Markdown....
2. In the new window, along the left, select Document (this should already be selected).
3. On the right, type some title for this new document (for example, `My first R Markdown document`).
4. On the right, select HTML (this should already be selected).

More on the next slide...

Introduction to R Markdown

Brief Exercise

R Markdown

The header

Formatting text

R code

Beyond this course

Brief Exercise

5. Your new R Markdown document should open. Save the document by selecting the icon in the pane or using the menu (File > Save), giving the file a `.Rmd` file extension.
6. Convert ("knit") the document to an HTML file by selecting the icon in the pane. A window should pop up displaying the HTML file.
7. If you enounter errors, try installing the knitr package:

```
> install.packages('knitr')
```

R Markdown

Introduction to R Markdown

Brief Exercise

R Markdown

The header

Formatting text

R code

Beyond this course

The header

Introduction to R Markdown

Brief Exercise

R Markdown

The header

Formatting text

R code

Beyond this course

The header

- at the top of your R Markdown file, you'll notice a section marked off with two sets of 3 dashes (---)
- for example:

```
---
title: "Untitled"
author: "Yea-Hung Chen"
date: "February 25, 2019"
output: html_document
---
```

The header

- the header defines basic metadata, including the title, author, and date
- RStudio automatically generated the header when you selected File » New, but you can also manually edit it

Formatting text

Headings

- in your R Markdown file, you'll also notice two hashes (##)
- this indicates a heading
- use varying numbers of # to indicate different heading levels

Headings

- for example:

```
# This is the first-level heading

This is some introductory text.

## This is a second-level heading

This is some more text.
```

Bold face

- to add bold face, use two sets of 2 asterisks (**):

```
I have something **very important** to tell you.
```

Italicization

- to add italicization, use two sets of 1 asterisk (*):

```
Have you ever looked at *Wikipedia*'s article on R?
```

Links

- to add links, use brackets and parentheses:

```
Have you ever looked at *Wikipedia*'s [article on
R](https://en.wikipedia.org/wiki/R_(programming_language))?
```

Lists

- to create unnumbered lists, use dashes (−):

```
Biostat 213 topics:
- object types
- importing data
```

- asterisks (∗) and plus signs (+) also work

Lists

- to create nested lists, indent by 4 spaces per level:

```
Biostat 213 topics:
- object types
    - numeric
    - character
- importing data
```

Lists

- to create numbered lists, use numbers:

```
Biostat 213 modules:
1. Fundamentals
2. Univariate analysis
```

Block quotes

- to add block quotes, use the greater-than sign (>):

```
*Wikipedia* states that:

> R is a programming language and free software environment
for statistical computing and graphics supported by the R
Foundation for Statistical Computing. The R language is
widely used among statisticians and data miners for
developing statistical software and data analysis.
```

R code

Code chunks

- to add R code, use two sets of 3 back quotes (```), curly brackets, and `r`:

````
```{r}
animals<-c('zebra','lion','elephant','giraffe')
animals
sort(animals)
```
````

- this is known as a code chunk

Code chunks

- there are various options for displaying code chunks
- to use these options, add them within the curly brackets:

````
```{r,echo=FALSE}
plot(bmxbmi~ridageyr,data=ii)
```
````

Code chunks

- useful options:
  - `echo`: controls whether or not the code is displayed (default is `TRUE`)
  - `collapse`: controls whether or not the code and output are displayed in the same block (default is `FALSE`)

Code chunks

- useful options for stylizing the chunk:
  - `prompt`: controls whether or not the R prompt and those pesky plus signs are shown (default is `FALSE` to facilitate copying-and-pasting)
  - `comment`: controls whether or not results are commented out (default is `##` to facilitate copying-and-pasting)

Code chunks

- for more options see the cheat sheet or documentation

kable()

- to output formatted tables, use `kable()` (you must have the knitr package installed and loaded)

````
```{r}
kable(ii[1:3,1:5],align='c')
```
````

kable()

- kable()'s align argument controls how columns should be aligned
- use l for left-aligned or c for centered
- for example, align='c' makes all columns centered
- repitition is allowed, with no spaces or commas, for multiple columns
- for example align='lcc' makes the first column left-aligned and the second and third columns centered

kable()

- `kable()`'s `digit` argument controls rounding
- for example, `digits=1` rounds all columns to 1 digit
- to specify the rounding for multiple columns, use a numeric vector
- for example, `digits=c(1,1,2)` rounds the first two column to 1 digit and the last column to 2 digits

kable()

- for other kable() arguments see help(kable)

Inline code

- to put code within text, use use two sets of back quotes (`) and `r`:
- for example:

```
The NHANES data file contains `r nrow(ii)` individuals.
```

- this is very useful for including results in text

Beyond this course

Other object types

- lists
- matrices and arrays

Regular expressions

- regular expressions for advanced character manipulation: `grep()` and `gsub()`
- you may consider *Handling Strings with R*

`for()` loops

- `for()` loops for iteratively running code

dplyr and tidyr

- the dplyr package for data manipulation
- the tidyr package for reshaping data

Statistical/epidemiological analysis

- the epitools package for measures of association
- `glm()` for generalized linear models (including logistic regression)
- the survival package for survival analysis (including Kaplan-Meier estimation and Cox proportional hazards models)
- the lmer package for mixed models
- the geepack package for generalized estimating equations
- the survey package for analysis of surveys
- the rmeta package for meta-analysis

Probability distributions

- various built-in functions exist for probability distributions
- for a list, see `help(Distributions)`
- for example, for normal distributions:
  - `rnorm()` for generating random numbers from normal distributions (useful for simulations or generating fake data)
  - `qnorm()` for finding z scores corresponding to areas under the curve
  - `pnorm()` for finding areas under the curve

Probability distributions

- for example, for the z score corresponding to a lower-tail area under the curve of 0.975:

```
> qnorm(0.975)
[1] 1.959964
```

Simulations

- for simulations, be sure to use `set.seed()`
- you may also consider using packages that take advantage of parallel computing, such as doParallel and doRNG