Contents

[3.1 Video: 1](#_Toc59527993)

[Markdown ideas 12:46 1](#_Toc59527994)

[Markdown syntax and Chunk options 19:18 2](#_Toc59527995)

[3.2 Video: 3](#_Toc59527996)

[Logical Statements 11:13 3](#_Toc59527997)

[Basics of dplyr 21:51 5](#_Toc59527998)

[3.3 Video 6](#_Toc59527999)

[Summarizing dplyr 9:26 6](#_Toc59528000)

[If vs ifelse 24:47 6](#_Toc59528001)

[3.4 Video 7](#_Toc59528002)

[Homework Notes 7](#_Toc59528003)

# 3.1 Video:

## Markdown ideas 12:46

What is a ‘markdown’ or ‘markup’ language?

Feedback

A markdown/markup language is one that uses plain text that is interpreted by software (such as a web browser) in order to render formatted text and other output.

What is the YAML header for?

* Meta-data about your document
* Indicating individual R code chunk options
* Setting global R chunk options
* Defining which markdown syntax to use

Feedback

* Right! In the YAML header you can set the author, output file type, and more.
* Individual R code chunk options are set at the top of each code chunk ```{r options\_here}
* You’ll set global R code chunk options in a ‘setup’ code chunk.
* There is only one R Markdown syntax, which we’ll look at in more detail shortly.

Check all that apply. Besides the YAML header, what are the other two main elements of an R Markdown document?

* Code chunks
* Plain text with simple text formatting
* Plots
* Numeric Summaries
* User input sections

Feedback

None

## Markdown syntax and Chunk options 19:18

What is the plain text that you should write so that "length(iris)" shows up as formatted "code" text in the final document?

* `length(iris)`
* \*length(iris)\*
* r length(iris)
* \_length(iris)\_
* \*\*length(iris)\*\*

Feedback

* Right!
* A single star on each side would do italic text.  The ‘backtick’ (often in the top left of the keyboard) begins and ends a section of text that you want to have code formatting.
* The ‘backtick’ (often in the top left of the keyboard) begins and ends a section of text that you want to have code formatting.  If this were inside backticks, it would execute the R code.
* An underscore on each side would do italic text.  The ‘backtick’ (often in the top left of the keyboard) begins and ends a section of text that you want to have code formatting.
* Double stars would indicate bold text.  The ‘backtick’ (often in the top left of the keyboard) begins and ends a section of text that you want to have code formatting.

TRUE of FALSE: You can create sub-sub lists (3-level lists) in R markdown (hint - give it a try to see!)

* TRUE
* FALSE

Feedback

* Right!  Just be mindful of the spacing. It is definitely recommended to turn on “show whitespace characters” in your RStudio options.
* If you indent the next line four more spaces and you should be able to get as many levels down as you want. It is definitely recommended to turn on “show whitespace characters” in your RStudio options.

TRUE or FALSE: Local chunk options override global chunk options.

* TRUE
* FALSE

Feedback

* Right!
* Global chunk options set the default code chunk settings, but local options (options on that particular chunk) override global ones.

What is the chunk code option to hide or show R code in the final document?

* echo
* eval
* message
* display

Feedback

* Right!
* eval determines whether or not to evaluate the code chunk.
* message determines whether or not to display any R messages that usually appear in the console when code is run.
* display is not an R code chunk option.

# 3.2 Video:

## Logical Statements 11:13

What is the standard comparison operator in R?

* ==
* !=
* =
* <->
* <=>

Feedback

* Right!
* This is the ‘not equal’ operator in R.
* This is an assignment operator in R (similar to <-) and also used in defining function arguments.
* This is not an R operator.
* This is not an R operator.

What "family" of functions can be used to check the type of R object you have?

* is.
* type.
* this.
* object.

Feedback

* Right!
* We used the function typeof(), but this won’t return TRUE/FALSE values.
* This isn’t a common R family of functions.
* This isn’t a common R family of function.

What does R do if you run the following code?

iris[c(1,0,0,1), ]

* Returns the first and fourth rows of iris
* Returns the second and third rows of iris
* Throws an error since the index vector is only of length four
* Returns the first row twice.

Feedback

* If you feed in a vector TRUE/FALSE, the data that correspond to indices TRUE are returned. However, if you directly feed in 1/0 values (without coercing them to be TRUE/FALSE values) R will return the 1st element each time it sees a 1 and won’t return anything when it sees a 0.
* If you feed in a vector TRUE/FALSE, the data that correspond to indices TRUE are returned. However, if you directly feed in 1/0 values (without coercing them to be TRUE/FALSE values) R will return the 1st element each time it sees a 1 and won’t return anything when it sees a 0.
* If you feed in a vector TRUE/FALSE, the data that correspond to indices TRUE are returned. However, if you directly feed in 1/0 values (without coercing them to be TRUE/FALSE values) R will return the 1st element each time it sees a 1 and won’t return anything when it sees a 0.
* Right!

TRUE of FALSE: subset and dplyr::filter can both be used to select rows from a data set (matrix or data frame).

* TRUE
* FALSE

Feedback

* Right!  Subset is a "base R" function and "filter" is a tidyverse function.
* Subset is a "base R" function and "filter" is a tidyverse function. Both return rows from a data set.

## Basics of dplyr 21:51

Why does the following code return different types of R objects?

iris$Species #Returns a vector

select(tbl\_df(iris), Species) #Returns a tibble

Feedback

Tbbles do not simplify when you subset them, even when you are returning only one column. This is programmatically easier to deal with because you’ll always know what type of object you’ll have returned to you when you subset your columns. Remember that you can use the pull() function to grab a single column and return it as a vector.

TRUE or FALSE: The select function from dplyr can be used to reorder the columns of a data frame or tibble.

* TRUE
* FALSE

Feedback

* Right! It can be used to reorder. The order of columns selected is how they are returned. The everything() function is useful here!
* It can be used to reorder the columns. The order of columns selected is how they are returned. The everything() function is useful here!

What will the following code return?

library(dplyr)

iris %>% select(ends\_with("Length"))

* Two columns of the iris data frame: Sepal.Length and Petal.Length
* An error since ends\_with can only be used with starts\_with is also used
* All rows where the Length variable is present
* An error since iris is used before the select function

Feedback

* Right!
* You can use either ends\_with(), starts\_with(), or both when using select().
* Functions like ends\_with() and starts\_with() are great for character matching columns.
* As long as dplyr (or tidyverse) has been read in, you have access to the %>% or pipe operator. This allows for writing of some code in a left-to-right fashion rather than a nested fashion.

# 3.3 Video

## Summarizing dplyr 9:26

The function ------------------ from dplyr will create new variable(s) and append it (them) to the data frame used to create it.

mutate, mutate()

Feedback

Remember that mutate will add the new variable(s) and transmute will just return the new variable(s).

What function from the dplry package can be used to create summaries of a variable at each level (or setting) of another variable (or variables)?

* group\_by()
* by()
* across()
* summarize\_on()

Feedback

* Right!
* When group\_by() is used, it creates a grouping attribute. summarize() then creates the requested summary for each group. (If no group exists then summarize() simply creates the overall summary.)
* When group\_by() is used, it creates a grouping attribute. summarize() then creates the requested summary for each group. (If no group exists then summarize() simply creates the overall summary.)
* When group\_by() is used, it creates a grouping attribute. summarize() then creates the requested summary for each group. (If no group exists then summarize() simply creates the overall summary.)

## If vs ifelse 24:47

What is the difference between using

if (condition) {

    execute code

} else {

    execute code

}

and

ifelse(condtion, if\_true\_execute, if\_false\_execute)

?

Feedback

If() can only take in a condition of length one whereas ifelse() (or if\_else()) can take a vector of conditions. Vectorized functions are generally very fast in R.

# 3.4 Video

None

# Homework Notes