This module will discuss a few approaches to manipulating, merging, and exporting data in R, focusing on the brackets operator, ifelse statements, the merge function, and the write table function.

**Lines 9 to 35** provide a reminder about how to set your working directory and read in data. We also add an ID column to the dataframe for use later in the module. Variable and value labels for the items that are used in the demonstration follow **on lines 39 to 75**.

**Lines 79 to 159** review the brackets operator and provide a few examples of ways to access the row sand columns of a dataframe, recode values, and create subset dataframes. In brackets, rows are indexed to the right of the comma and columns are indexed to the left of the comma. Columns and rows can be indexed by supplying either text labels or the number corresponding to the row or column (for example, row 1 or column 1). When interacting with objects with only 1 dimension, the brackets should not include a comma.

[SCROLL to 108]

Recoding can be accomplished with brackets by identifying the rows and columns to recode and passing the new value to the object with an arrow. Note that at **line 112** we use a double equals sign to indicate that we want to recode the rows of the variable that equal 5. A single equals sign is equivalent to the arrow in R.

[SCROLL to 130]

Here, we set any values greater than 7 to missing in the three variables. NA is the code for missing in R.

Notably, when we want to index the missing values that are already in a variable (rather than assign a code to missing) we have to use the is.na() function. Math operations that include NA will always return NA, including the double equals sign.

[SCROLL to 148]

The brackets operator can be used to create subsets of data that meet certain conditions. At **line 152** we create a new dataframe with only the rows of df2 where the variable is equal to 1. **Line 154** creates a new dataframe with only the rows of df2 where the variable is equal to 2.

**Lines 163 to 198** demonstrate how to use ifelse statements for recoding. Ifelse statements have three arguments: the condition to check in the data, the value to pass if the condition is TRUE, and the value to pass if the condition is FALSE. Ifelse is a vectorized function, meaning that is automatically applies a procedure to all values in a column. At **line 183**, we pass a 0 to the new object if the depression frequency variable is 0 and, if not, pass the current value of depression level. **Line 193** demonstrates how to add an additional condition to the ifelse statement.

**Lines 201 to 250** provides an example of how to merge two dataframes the share a common index variable. In this case, we merge two dataframes using the ID variable that we generated at the beginning of the module.

[SCROLL to 246]

**Lines 246 to 250** provide a quick overview of the different merging options that are available in the merge() function to control which rows are retained in the merged file.

**Lines 254 to 266** demonstrate how to export a dataframe from R with the write table function.