This module will discuss character strings, pattern matching, and for loops.

**Lines 7 to 24** provide a reminder about how to set your working directory and read in data. Variable and value labels for the items that are used in the demonstration follow on **lines 29 to 96**. We generate a dataframe that only includes the variables of interest at **lines 100-101.**

**Lines 105 to 165** introduce two functions that are useful for manipulating character strings : str\_split\_fixed and paste.str\_split\_fixed is used to split a string by a given character into a given number of sections. For example, **line 122** supplies a string, **line 123** supplies the character to split on, and **line 124** supplies the number of groups to split into.

[SCROLL TO 141]

str\_split\_fixed can also be used on a series of strings, as demonstrated on **lines 146 to 148.**

**Lines 157 to 162** demonstrate how to use paste to combine strings. Here, a series of strings is combined with the column names of the subset dataframe to add an additional label to all of the specified columns.

**Lines 168 to 224** provide an overview of two functions for pattern matching and pattern replacement, grep and gsub. **Lines 180 to 182** demonstrate how to use grep to search the columns of df\_subset for any entries that include “underscore module 4”. The value argument is set to TRUE so that grep returns the matching column names, rather than the number associated with them. Pattern matching can be extended to include multiple patterns, as shown on **line 190**.

[SCROLL TO 194]

grep output can be used directly inside the brackets operator to extract the columns that match the given pattern from a dataframe; for example, **line 199** extracts the columns that include underscore module 4 in their name from df\_subset and saves those columns as a new dataframe.

[SCROLL TO 206]

gsub can be used to replace a given string with a new series of characters. **Lines 214 to 216** replace the string “Module 5” with the number “5” in the column names of df\_subset, and save the output as the new column names of df\_subset.

**Lines 226 to 326** provide an overview of how to construct for loops in R. For loops are a flexible way to repeat a given procedure over a specified interval. An example to illustrate how for loops work follows. **Line 240** generates a random sequence of 5 digits. **Lines 244 to 248** demonstrate how each of these 5 digits can be printed to the console individually.

[SCROLL TO 250]

**Lines 252 to 254** accomplish this procedure with a for loop. The variable “i” is replaced with values 1 through 5. At **line 253**, we specify that the loop should print element number “i” from the random sequence of digits. **Line 254** closes the loop**. Lines 258 to 267** provide a depiction of what is happening at each iteration of the loop. The command inside the loop remains the same between loop iterations, but the value of “i” changes.

[SCROLL TO 269]

It is often useful to define the range of a loop based on the dimensions of existing data objects. For example, if we want to create a table for every column in df\_subset, we can define “i” to take values 1 to the number of columns in df\_subset.

[SCROLL TO 279]

When loops include more than one dataframe and use a single index variable, it is important to check that the columns of the dataframe align as expected. **Line 293** lines up the columns of two dataframes to provide a visualization of which column “i” refers to in each dataframe.

[SCROLL TO 304]

Lines 323 and 326 provide links to some useful information about writing efficient loops and alternatives to using loops.