codecademy

Learn R: Data Cleaning

gsub() R Function

The base R `gsub()` function searches for a regular expression in a string and replaces it. The function recieve a string or character to replace, a replacement value, and the object that contains the regular expression. We can use it to replace substrings within a single string or in each string in a vector.
When combined with dplyr's `mutate()` function, a column of a data frame can be cleaned to enable analysis.

```
# Replace the element "1" with the empty
string in the teams vector in order to
get the teams_clean vector with the
correct names.
teams <- c("Fal1cons", "Cardinals",
"Seah1awks", "Vikings", "Bro1nco",
"Patrio1ts")

teams_clean <- gsub("1", "", teams)

print(teams_clean)

# Output:
# "Falcons" "Cardinals" "Seahawks"
"Vikings" "Bronco" "Patriots"
```

gather() tidyr

The `gather()` function from tidyr package is useful to gather columns over a data frame into key-value pairs, changing the shape of a data frame from wide to long. The original data frame has multiple columns that can be gathered, in a unique structure of key-value pair with all values in one column and the column names in another column.

distinct() dplyr

The `distinct()` function from dplyr package is used to keep only unique rows on a data frame. If there are duplicate rows, the function will preserve only the first row. The function can be used to remove equal rows of a dataframe, and to remove rows in a data frame based on unique column values or unique combination of columns values.

```
# Keep unique rows in the
match_statistics  data frame
distinct(match_statistics)

# Keep only rows with different values in
the prices column of trips
# dataframe
distinct(trips,prices)
```

str() Function

The `str()` function display the internal *structure* of an R object that is passed as parameter of the function. The function outputs the data structure of the object as well as the elements of the object. When the object is a dataframe, the function returns the data type of each column in the data frame, the number of observations and the number and variables.

Combing Data with R

Data from multiple files can be combined into one data frame using the base R functions `list.files()` and `lappy()`, with readr's `read_csv()` and dplyr's `bind_rows()` functions. Consider the following steps:

1. *Get the list of files*. The following code will get a list of all files in the current directory that match the pattern "file_.*csv"

```
files <- list.files(pattern
= "file_.*csv")
```

2. *Read in the files*. The following code applies `read_csv()`, a function from readr, to each file, and adds the resulting data frames to the list `df_list`.

```
df_list <- lapply(files,
read_csv)
```

3. *Combine the file data*. Below `bind_rows()`, a dplyr function, is used to combine the data from each data frame in the list into one data frame.

```
df <- bind_rows(df_list)
```

R as.numeric() Function

The base R `as.numeric()` function can coerce character string objects into numeric types.
This function is useful because often numbers are stored as characters which do not allow operations or analysis. The function receives the object to be transformed as a parameter and transforms it to numeric.
When this function is combined with the `mutate()` function from dplyr, new columns of a dataframe can be created with the numeric data type.

str_sub() function

The `str_sub()` function from the stringr package can split a string by index position separating combined data values into their individual components. The function uses the `start=` and `end=` arguments to perform the split operation. This function can be used with `mutate()` from dplyr in order to generate multiple new columns on a data frame based on split string values of a particular column.

```
# This command would take the first index
to the five index of the string.
str_sub('Marya1984', start=1,end=5)
```

The dplyr and tidyr packages

The `dplyr` and `tidyr` packages provide functions that solve common data cleaning challenges in R.
Data cleaning and preparation should be performed on a "messy" dataset before any analysis can occur. This process can include:

- diagnosing the "tidiness" of the data
- reshaping the data
- combining multiple files of data
- changing the data types of values
- manipulating strings to better represent the data

Tidy Dataset

In a tidy dataset each variable is represented by a column, and each row is a separate observation. Tidy datasets are the best way to conduct data analysis on specific data. By adhering to the standard of a tidy dataset, it is easier for an analyst to extract from.
Datasets that are not tidy present some issues in their structure such as one column storing multiple variables, the same information of a variable is spread out in multiple columns, or the variables can be stored in both rows and columns.

separate() Function

The `separate()` function from the tidyr package, is used to separate a single character column of a data frame into multiple columns. Arguments of this function are, in order, a dataframe, the column used to create the new columns(column name or column position in the data frame), the new column names that will be used, and the separator argument. The default seperator will match any non-alphanumeric sequence, such as a space or semicolon.

```
# This function would separate the complete_name column into new columns called names and surnames on the individuals data frame.
separate(individuals, complete_name, c("names","surnames"))
```