codecademy

Learn R: Data Frames

dplyr package

The dplyr package provides functions that perform data manipulation operations oriented to explore and manipulate datasets. At the most basic level, the package functions refers to data manipulation "verbs" such as select, filter, mutate, arrange, summarize among others that allow to chain multiple steps in a few lines of code. The dplyr package is suitable to work with a single dataset as well as to achieve complex results in large datasets.

data frame object

A data frame is an R object that store data in two dimensions represented by columns and rows. The columns are the different variables of the dataframe and the rows are the observations of each variable. Each row of the dataframe represent a unique set of observations. This object is a useful data structure to store data with different types in columns and perform analysis around them.

Excluding Columns with select() in Dplyr

The select() function of dplyr allows users to select all columns of the data frame except for the specified columns. To exclude columns, add the `-` operator before the name of the column or columns when passing them as an arguments to select(). This will return a new data frame with all columns except ones preceded by a `-` operator. For example: `select(-genre, -spotify_monthly_listeners, -year_founded)`.

Loading and Saving CSVs with R

The `read_csv()` and `write_csv()` functions belong to the tidyverse package and perform smart reading and writing operations of files in R. The `read_csv()` function reads a file and converts it to a better format of a data frame called a tibble. The first argument of the `read_csv()` is the file to be read. Tibbles in R can be exported to csv files using the `write_csv()` function. The first argument of `write_csv()` is the tibble to be exported.

filter with logical operators

The filter() function can subset rows of a data frame based on logical operations of certain columns. The condition of the filter should be explicity passed as a parameter of the function with the following syntax: name of the column, operator(<,==,>,!=) and value. On the other hand is possible to chain conditions within a column or on different columns using logical operators such as boolean operators(&,|,!).

Dplyr's filter()

The `filter()` function of the dplyr package allows users to select a subset of rows in a data frame that match with certain conditions that are passed as arguments. The first argument of the function is the data frame and the following arguments are the conditional expressions that serve as the `filter()` criteria. For example: `filter(artists, genre == 'Rock', spotify_monthly_listeners > 20000000)`.

data frames primary information

Data frames in R can be inspected using `head()` and `summary()`. The `head()` function accepts an integer argument which determines the number of rows of the data frame that you can see. The default value of the `head()` function is 6. The `summary()` returns summary statistics such as min, max, mean, and three quartiles.

Dplyr's select()

The `select()` function of dplyr package is used to choose which columns of a data frame you would like to work with. It takes column names as arguments and creates a new data frame using the selected columns. `select()` can be combined with others functions such as `filter()`.

dplyr arrange()

The `arrange()` function of the dplyr package orders the rows of a data frame based on the values of a column or a set of columns that are passed as parameters. The resulting order of the data frame can be in ascending or descending order. By default `arrange()` orders the dataframe in ascending order, but it is possible to change this and order the data frame in descending order using the `desc()` parameter over the column.

Comma Separated Values (CSV)

CSV (Comma-separated values) files represent plain text in the form of a spreadsheet that use comma to separate individual values. This type of file is easy to manage and compatible with many different platforms. This file can be imported to a database or to an Integrated Development Environment (IDE) to work with its content.

pipes

The pipe `%>%` can be used to input a value or an object into the first argument of a function. Instead of passing the argument into the function seperately, it is possible to write the value or object and then use the pipe to convert it as the function argument in the same line. This can be used with the functions *select()* and *filter()* that contain a data frame as the first argument.
In the example, the weather data frame is piped into the select function that would select the first two columns of the weather data frame.

```
weather %>% select(1:2)
```

rename-dplyr

The `rename()` function of *dplyr* package can be used to change the column names of a data frame. It has a simple syntax where it is necessary to pass the new name followed by the `=` operator and the old name of the column. On the other hand to rename multiple columns based on logical criteria, the `rename()` function has variants such as `rename_if()`, `rename_at()` and `rename_all()`.

transmute() dplyr

The `transmute()` function, from dpylr, creates new columns from a data frame by transforming existing ones. The result of the function is the new column while all the original columns are removed in the new data frame. The function receives the data frame as the first argument, and the new variable name with the function to transform it as the second parameter. It is possible to perform multiple transformations in the same line, by specifying each individual transformation.

```
transmute(population, increase
= total_population/lag(total_population))
```

mutate() dplyr

The `mutate()` function from dplyr package adds new columns to an existing data frame based on a transformation of an existing column, while maintaining all the other columns. The function receives the data frame as the first parameter, and subsequently specify the new column name followed by the `=` operator and a transformation function. After the first variable parameter, further parameters can be added to mutate more variables at the same time.

```
mutate(heights, cm = inches * 0.39)
```