**Lab 2: Exploratory Data Analysis**

**2.1. Contingency table review**

In this lesson you’ll work with the comics dataset introduced previously. This is a collection of characteristics on all of the superheroes created by Marvel and DC comics in the last 80 years.

***Please upload the \*.csv file from your computer in R, using read\_csv() function.***

Let’s start by creating a contingency table, which is a useful way to represent the total counts of observations that fall into each combination of the levels of categorical variables. Make sure the contingency table has the different levels of gender in the rows, and the different levels of align in the columns.

A common way to represent the number of cases that fall into each combination of levels of two categorical variables, such as these, is with what’s called a “contingency table.” Creating a contingency table requires three steps:

1. use the count() function to count the number of observations
2. specify the variables you are interested in **inside** the count() function
3. pivot the table from its current “long” format to a “wide” format using the pivot\_wider() function

Example:

# to get a wider table

comics |>

count(align, id) |>

pivot\_wider(names\_from = id, values\_from = n)

|  |
| --- |
| ***Exercise 1***   1. Print the first rows of the dataset 2. Check the levels of ***align*** column 3. Check the levels of ***gender*** column 4. Create a 2-way contingency table.   *Note: assign the* ***tab*** *name to the resulting data frame* |

**2.2. Dropping levels**

The contingency table from the last exercise revealed that there are some levels that have very low counts. To simplify the analysis, it often helps to drop such levels from the dataframe.

In R, this requires two steps:

1. filter out any rows with the levels that have very low counts
2. remove these levels from the variable with droplevels()

We are using the droplevels() function to eliminate any levels that have zero counts from a variable.

The contingency table from the last exercise is available in your workspace as tab. We’ll use this table and the **dplyr** package to filter()levels with few observations from our dataframe, using the following steps:

|  |
| --- |
| ***Exercise 2***   1. print tab to explore which level of align has the fewest total entries 2. use filter() to filter out all rows of comics with that level 3. use droplevels() to drop the unused levels from the dataframe 4. save the simplified dataset as comics\_filtered 5. check if the unused level of align is dropped |

**2.3. Side-by-side barcharts**

Here you’ll construct two side-by-side barcharts of the comics data. This exercise will show you that often there can be two or more options for presenting the same data. Passing the argument position = "dodge" to geom\_bar() tells the function that you want a side-by-side (i.e. not stacked) barchart.

To create these plots, let’s carry out the following steps:

|  |
| --- |
| ***Exercise 3***   1. load the **ggplot2** package. 2. create a side-by-side barchart with align on the x-axis, and fill the bars with the gender of the character 3. create another side-by-side barchart with sex on the x-axis, and fill the bars with the alignment of the character. |

In many visualizations you make, you may be interested in changing the axis labels on the plot. Here, the axis labels of “gender” and “align” could use some spicing up, to be more descriptive. To change the labels of a ggplot(), you add (+) a labs() layer to the plot, where you can specify the x, y, and fill labels, as well as the plot’s title.

**Finally, make the interpretation of the bar charts.**