Previously, I wrote a [blog post showing a number of R packages and functions which you could use to quickly explore your data set.](https://www.littlemissdata.com/blog/simple-eda) Since posting that, I’ve become aware of another exciting EDA package: [inspectdf](https://github.com/alastairrushworth/inspectdf) by [Alastair Rushworth!](https://twitter.com/rushworth_a) As is very often the case, I became aware of this package in a [twitter post](https://twitter.com/dataandme/status/1124980819586617344) by none other than [Mara Averick](https://twitter.com/dataandme).

Preview of some of the inspectdf output graphs

I like this package because it’s got a lot of functionality and it’s incredibly straightforward to use. In short, it allows you to understand and visualize column types, sizes, values, value imbalance & distributions as well as correlations. Better yet, you can run each of these features for an individual data frame, or compare the differences between two data frames.

I liked the inspectdf package so much that in this blog, I’m going to extend [my previous EDA tutorial](https://www.littlemissdata.com/blog/simple-eda) with an overview of the package.

**Set Up**

**Working Environment**

For this tutorial, we are going to be using R as our programming language. The entire code is hosted in my [github repo,](https://github.com/lgellis/MiscTutorial/tree/master/inspectdf) and you can also copy and paste to follow along below. If you are looking to understand your options for an R working environment, I recommend that you can [check out IBM Watson Studio to run hosted R notebooks](https://www.littlemissdata.com/blog/watsonstudio), or [RStudio](https://www.rstudio.com/products/rstudio/#Desktop).

**Install and Load Packages**

Before we get rolling with the tutorial, we need to get our environment ready. Please remember that if you do not have any of the packages already installed, uncomment the installation line by removing the #.

#First install devtools to allow you to install inspectdf from github

#install.packages("devtools")

library(devtools)

#install and load the package - https://github.com/alastairrushworth/inspectdf

#devtools::install\_github("alastairrushworth/inspectdf")

library(inspectdf)

#install.packages("tidyverse")

library(tidyverse)

#install.packages("readr")

library(readr)

**Download the Data**

We are going to be using the survey data from my previous [data + art STEAM project.](https://www.littlemissdata.com/blog/steam-data-art2) Note that there were some issues with survey gathering and therefore you will see some odd values in the data.

#Download the data set

df= read\_csv('https://raw.githubusercontent.com/lgellis/STEM/master/DATA-ART-1/Data/FinalData.csv', col\_names = TRUE)

**Transform the Data Set**

We will create three data frames for our tutorial.

* **allGrades** is the full data frame with the complete set of survey results
* **oldGrades** includes a subset of the survey results for all grades greater than 5. This includes grades 6-8.
* **youngGrades** includes a subset of the survey results for all grades less than 6. This includes grades 3-5.

We will use allGrades for the single data frame analysis and oldGrades and youngGrades for the data frame comparisons.

allGrades <- df

oldGrades <- allGrades %>%

filter(Grade > 5)

youngGrades <- allGrades %>%

filter(Grade < 6)

#View the distribution of grade to ensure it was split properly

ggplot(oldGrades, aes(x=Grade)) + geom\_histogram()

ggplot(youngGrades, aes(x=Grade)) + geom\_histogram()

**Run the Package Functions**

For each of the functions, we are going to run it first against the full data frame (allGrades) to view the basic functionality. We will then pass two data frames into the function (oldGrades, youngGrades) to see how the data frame comparison works.

**inspect\_types()**

We can use the inspect\_types() command to very easily see a breakdown of character vs numeric variables.

inspect\_types(allGrades, show\_plot = TRUE)

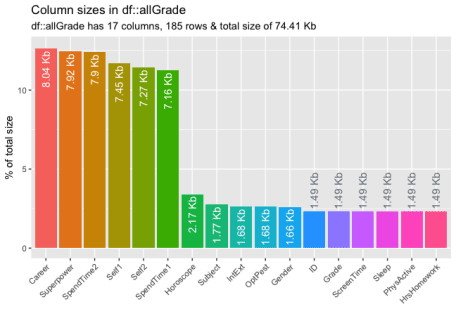
inspect\_types(youngGrades, oldGrades, show\_plot = TRUE)

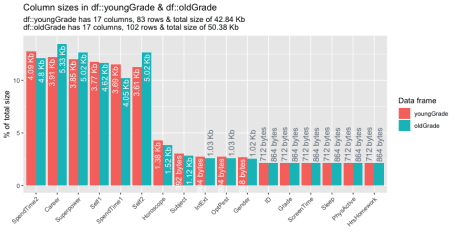
**inspect\_mem()**

The inspect\_mem() function will tell us some basic sizing information, including data frame columns, rows, total size and the sizes of each variable.

inspect\_mem(allGrades, show\_plot = TRUE)

inspect\_mem(youngGrades, oldGrades, show\_plot = TRUE)



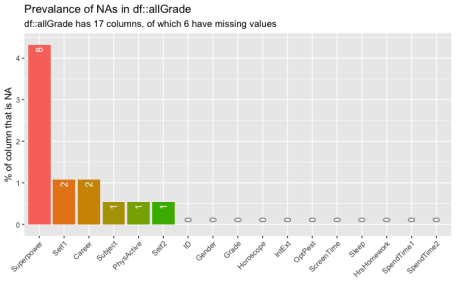


**inspect\_na()**

The inspect\_na() function shows us the percentage of na values for each variable. The comparison view is quite neat as it highlights variables with unequal na percentages.

inspect\_na(allGrades, show\_plot = TRUE)

inspect\_na(youngGrades, oldGrades, show\_plot = TRUE)



**inspect\_num()**

The inspect\_num() function shows us the distribution of the numeric variables. The heat plots used for the data frame comparison are pretty cool. Though, I think I might’ve liked [histograms with two bars](https://stackoverflow.com/questions/50648388/histogram-with-two-variables-in-ggplot) (one for each data frame) a little better.

inspect\_num(allGrades, show\_plot = TRUE)

inspect\_num(youngGrades, oldGrades, show\_plot = TRUE)

**inspect\_imb()**

Similar to the inspect\_num() function, the inspect\_imb() function allows us to understand the a bit about the value distribution for our categorical values. It shows the most prevalent values for each variable and displays how prevalent they are.

inspect\_imb(allGrades, show\_plot = TRUE)

inspect\_imb(youngGrades, oldGrades, show\_plot = TRUE)

**inspect\_cat()**

A step further from inspect\_imb(), inspect\_cat() allows us to visualize the full distribution of our categorical values. Note that if there are a lot of unique values in a particular category, it’s not expected that you should see every value. However, it quite nicely surfaces common values.

inspect\_cat(allGrades, show\_plot = TRUE)

inspect\_cat(youngGrades, oldGrades, show\_plot = TRUE)

**inspect\_cor()**

We finish off our review with the inspect\_cor() function. This allows us to see the Pearson correlation coefficient to see how the variables may relate to one another. [onlinestatbook.com](http://onlinestatbook.com/2/describing_bivariate_data/pearson.html) has a great definition of the Pearson correlation coefficient below.

The **Pearson correlation coefficient**, r, can take a range of values from +1 to -1. A value of 0 indicates that there is no association between the two variables. A value greater than 0 indicates a positive association; that is, as the value of one variable increases, so **does** the value of the other variable.

inspect\_cor(allGrades, show\_plot = TRUE)

inspect\_cor(youngGrades, oldGrades, show\_plot = TRUE)

**Thank you**

Thank you for exploring the inspectdf package with me.  Please comment below if you enjoyed this blog, have questions, or would like to see something different in the future.  Note that the full code is available on my [github repo.](https://github.com/lgellis/MiscTutorial/tree/master/inspectdf)

If you have trouble downloading the files or cloning the repo from github, please go to the [main page of the repo](https://github.com/lgellis/MiscTutorial) and select “Clone or Download” and then “Download Zip”. Alternatively or you can execute the following R commands to download the whole repo through R

install.packages("usethis")

library(usethis)

use\_course("https://github.com/lgellis/MiscTutorial/archive/master.zip")