# DAT 375 Module Four Assignment

# Daniel Gorelkin

### Introduction:

Data analysis could not be conducted without making observations and understanding the dataset. In most, if not all, cases, the scraped data must be filtered, altered, and eventually cleaned in order to conduct meaningful and accurate analysis. For example, a data analyst can retrieve a dataset that contains structural errors, missing values, duplicated records, outliers, and even values stored as the wrong data type. Therefore, before running the analysis, our dataset must undergo preprocessing, in the form of data cleaning and transformation to minimize the “garbage” that gets into our model, so that we can minimize the amount of inaccuracies and errors that our model outputs (Larose, 2015-04-13). For instance, the following dataset exploration reveals some errors in the column naming, multiple spaces between words in the fields, strings instead of numerical values, missing entries, and unexpected values.

*A screenshot of a calendar

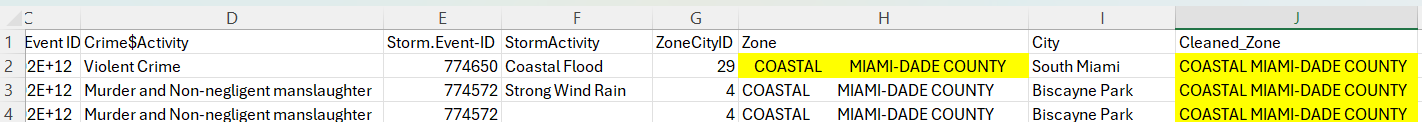
AI-generated content may be incorrect.*

Hence, such a dataset should be treated according to the CRISP-DM’s data preparation phase, and all the errors that can negatively impact our modeling should be removed.

To clean our data, we can use several available tools on the market, starting from the most well-known spreadsheet, Excel, or by using more advanced statistical tools such as SAS, Python, MySQL, or the R programming language. Obviously, there is no one “do it all tool,” and some tools can perform better than others, or require different operation skill levels from the user and the operator. In this article, I will focus on and compare two distinct methodologies by cleaning data manually and by using built-in formulas in Excel and the more advanced programmatic approach by using R.

### Clearing extra spaces in Excel VS R

One of the most common issues we can find in scraped data is whitespace. This can be found as leading and trailing spaces, or even as an inner single or multiple spaces in a single field, as can be seen in the H2 cell, “ COASTAL MIAMI-DADE COUNTY ” from the screenshot above. To address such issues, we can turn to Excel and modify the entries in a new column using the *TRIM()* function to rewrite a single field or an entire column. However, such a technique creates an additional column of data, or could be tedious when applied manually in a large dataset.



On the other hand, by cleaning the dataset using R, and specifying a single column or the entire dataset can be searched automatically, and the found whitespace can be mutated without adding any extra fields or columns by running several lines of code. For example, we can use the *trimws()* function from the *stringr* package to clear the leading and the trailing whitespace, or to clean all existing whitespaces in a field/s or the entire dataset by using the *str\_squish()* function, as can be seen in the following example:

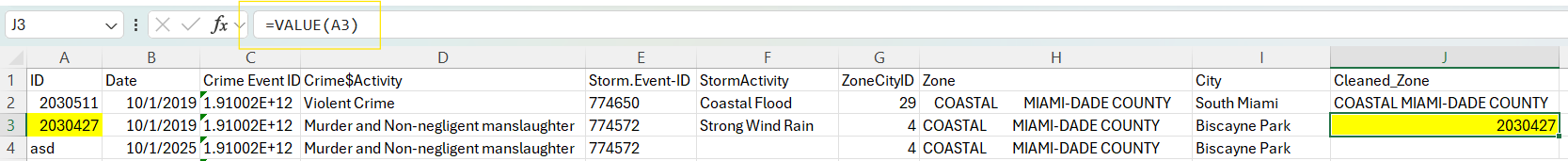
A screenshot of a computer

AI-generated content may be incorrect.

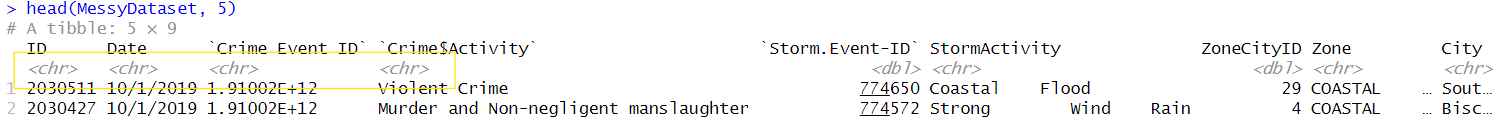
From this example, we can see how easy, fast, and effective it is to clear extra spaces from values in R compared to the manual and ineffective process in Excel. Hence, R can be considered the preferred tool for clearing unwanted whitespace from a messy dataset.

### Converting numbers stored as text into numerals

Another less evident but no less common issue with data cleaning is converting the type of the stored data from one data type to another. For example, the data stored in our A3 cell can be stored as a string type; however, to manipulate this value and produce a value out of it, we might need to use it as an integer or double type, though our data must be converted to a different data type. Again, this operation is not a deal breaker in Excel and could be done by using the *=VALUE(A3)* function, which will create a copy of the A3 cell and will store the data as a numerical type.



Similar to Excel’s *TRIM()* function, the *VALUE()* function can be applied to a single field or an entire row or column by increasing the dataset and consuming more storage space.  
R, on the other hand, offers us the capabilities of rewriting the values in several columns all at once from one type to another by writing only several lines of code. The following example demonstrates the mutation of several columns from type *chr* to *double*.



A screenshot of a computer

AI-generated content may be incorrect.

Once again, despite the more complex task of writing the code in R, we are gaining more control and taking the lead over our data cleaning process compared to the same task in Excel.

### Removing irrelevant or duplicate data

Once again, cleaning data duplication seems to be an easy task with R, although the task of removing redundant records in Excel could be done by using the existing tool from the ribbon menu and selecting the desired columns to be taken care of:

A screenshot of a computer

AI-generated content may be incorrect.

Running the same operation in R seems to be even more intuitive for a coding knowledgeable user.

By pushing our dataset through the pipe operator to the *distinct()* function from *dplyr* package, we can easily get rid of all the record duplication in our dataset. For instance, in our dirty dataset *MessyDataset*, we can find that the second and the fourth observation rows are identical and redundant. Therefore, all of the rows from the dataset that repeat themselves could be safely removed if their presence adds no value to the analysis.

A screenshot of a computer

AI-generated content may be incorrect.

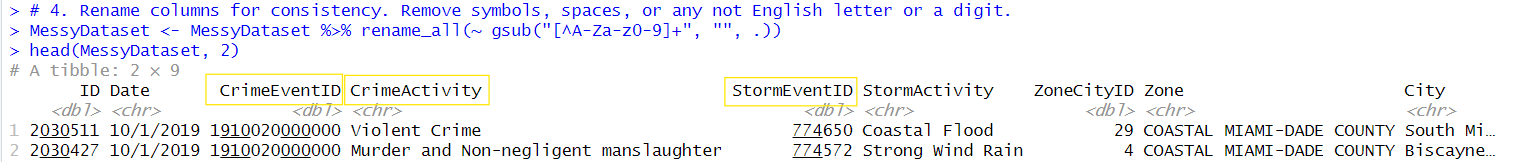
Once again, the benefit of using one line of code in R gives us great flexibility and functionality compared to the selective method of columns and data to be manipulated in Excel.

### Fixing structural errors and altering formatting as needed

Structural errors, such as renaming column names and value cleaning, such as removing commas in some places and semicolons in others, could be a significant task if they were to be performed while considering each character individually. Even here, R excels in accomplishing such a task by importing the *dplyr* package. By writing only one line of R code, the complete dataset can be scanned, modified, and corrected. For example, the following line of code rewrites all the values in the dataset and removes unwanted characters like $, -, ‘, ;, and allows only alphanumeric characters to be present.



As a result, it renames the dirty column names from `*Crime Event ID*` to *CrimeEventID*, `*Crime$Activity*` to *CrimeActivity*, `*Storm.Event-ID*` to *Storm.Event-ID*, and cleans all of the records in the dataset in the same breath:



Once again, the benefit of using the *rename\_all()* function from the *dplyr* package in R gives us great flexibility and functionality compared to the selective method of columns, individual characters, and data in Excel, as the latter must be performed manually while iterating through each character repeatedly by considering every possible unwanted character to be altered.

A screenshot of a computer

AI-generated content may be incorrect.

### Filtering unnecessary outliers

It is just as important to identify and clean outliers from our dataset, as they may represent errors in data entry, and even if an outlier is a value and not a mistake, running a model while the outlier is included may deliver unreliable results. This could be done both in Excel and by using R, both in a graphical and programmatic way.

While finding the outliers in Excel could be done by sorting column values in an ascending or descending order and applying some logic to it, it could also be done by creating box plots with a whisker chart that can assist us in spotting outliers visually. After an outlier was found, it should be removed manually from the dataset, which keeps our model prone to human errors.

The same visualization method could also be used in R, but it will make more sense to filter outliers in R programmatically by using Z-score and calculating the standard deviations for the inspected columns, or by using the tools from R’s outliers package such as Grubbs test (see: *Outliers Package - RDocumentation*, n.d.).

With that said, finding and filtering outliers using R will provide much better results when applied to a large-scale dataset, as it can provide more detailed, scaled, and reliable filtering, while the operation in Excel might be suitable for quick visual checkups, users who are unfamiliar with R, or small-scale projects.

### Handling missing data

Lastly, specific importance in data cleaning may have the missing data replacement or deletion feature. Many “dirty” datasets may contain empty, null, or unique values held in their fields, which can significantly impact our model quality. Hence, such values must be modified, rewritten, or removed from the cleaned dataset. Here as well, it could be done both in Excel and R. And just like the example mentioned previously, the capabilities of using the R language outweigh Excel by far. For example, to mark “empty” cells in Excel or to modify our cell value, we can use the conditional formatting formulas by creating a new column that will hold the new records as shown in column *J* such as *“=IF(D3="", "InvalidValue", ""*)” and drag the formula down for the rest of the rows in the dataset. Or, alternatively, we can use the built-in conditional formatting function and create different use cases for each column as shown below:

A screenshot of a computer

AI-generated content may be incorrect.

Similarly, we can implement this logic in R by importing the *tidyr* package from CRAN and using the replace\_na(.x, "Missing\_Data") function that will search all values from the dataset and format all NA values to "Missing\_Data" by running a single line of code. That technique could be used to find empty, null, or any other values and modify them in a flash. For instance, in the following example from a messy dataset,

A close-up of a white background

AI-generated content may be incorrect.

the function rewrites all the numerical (double) empty fields in the dataset into 0s, and modifies the empty string (char) fields to “Missing\_Data”:

A computer code with text

AI-generated content may be incorrect.

As we can see from the examples above, handling missing data could be done in Excel, while no programming knowledge is required to complete the operations, but, for handling large and complex datasets with multiple columns and thousands of rows, this manual operation might take hours if not days, while getting the same result in R will take several minutes or hours in the worst case. Therefore, for professional usage with reliable output, there is no doubt that using R compared to a Microsoft Excel spreadsheet is a much better choice.

### References

*Formulas and functions - Microsoft Support. (n.d.).* [*https://support.microsoft.com/en-us/office/Formulas-and-functions-294d9486-b332-48ed-b489-abe7d0f9eda9#ID0EAABAAA=More\_functions*](https://support.microsoft.com/en-us/office/Formulas-and-functions-294d9486-b332-48ed-b489-abe7d0f9eda9#ID0EAABAAA=More_functions)

*Function reference. (n.d.).* [*https://stringr.tidyverse.org/reference/*](https://stringr.tidyverse.org/reference/)

*Larose, D. T. (2015-04-13). Data Mining and Predictive Analytics, 2nd Edition. [[VitalSource Bookshelf version]]. Retrieved from vbk://9781118991121*

*Outliers package - RDocumentation. (n.d.).* [*https://www.rdocumentation.org/packages/outliers/versions/0.15*](https://www.rdocumentation.org/packages/outliers/versions/0.15)

*Replace NAs with specified values — replace\_na. (n.d.).* [*https://tidyr.tidyverse.org/reference/replace\_na.html*](https://tidyr.tidyverse.org/reference/replace_na.html)