Data Cleaning in R

INFO 640 | Pratt Institute | McSweeney

Introduction

This lab will introduce you to basic data cleaning techniques. We will work with a toy dataset, though the principles apply to larger datasets. Data cleaning is the most time consuming and often most challenging part of any data analysis project. Especially early in your programming career, you may want to alternate between a program like Excel to "see" your data and R to manipulate it, since seeing your data will help you get a better sense of what you will need to do to clean it.

I *strongly* encourage you not to clean in Excel. Excel does not allow you to keep track of your steps, and you will need to report what you did to clean your data when you report your findings.

Getting Started

Lubridate helps us work with dates, and dplyr helps us clean data.

```
#install.packages("lubridate")
#install.packages("dplyr")
library(tidyverse)
library(dplyr)
library(lubridate)
```

Load the mealplan dataset. Also open it in Excel to get a sense of what is going on with this dataset. Do not save it in Excel.

```
meals <- read.csv("Desktop/INFO640-Labs/Datasets/mealplan.csv", header=TRUE)
```

make sure it read as a dataframe

```
class(meals)
```

Let's ask a few questions about this dataset just to get a handle on it.

how big is this dataset?

```
dim(meals)
```

what are those variable names?

```
names(meals)
```

overview of the dataset

```
str(meals)
```

get a better overview

```
glimpse(meals)
```

statistical summary of the data

```
summary(meals)
```

check out the beginning and the end of the data set

```
head(meals)
tail(meals)
```

Articulate the Problems

A key step in datacleaning is to make a plan for what needs to be cleaned and what transformations need to be made. Identify as many problems as you can and make a list. Writing this down in your language will help you stay organized and discover if any of the steps need to be ordered.

- all of these variables were read as factors, change Meal & Location to characters.
- price is connected to the entree, break those apart
- the data in the columns beginning with X are actually dates, transform those
- there's data in the headers - this will have to be changed *first*

pivot the columns with data in the headers

```
meals1 <- gather(meals, date, value, -Meal, -Location)
head(meals1)
```

break apart the entree/price column

```
meals1 <- separate(meals1, value, c("entree", "price"), sep=',')
str(meals1)
```

fix the dates

```
meals1$date <- mdy(meals1$date)
head(meals1)
```

remove the dollar sign

```
meals1$price <- gsub("\\$","",meals1$price)
head(meals1)
```

force the prices as numeric

```
meals1$price <- as.numeric(meals1$price)
```

now that we have a numeric, we can check the values with a historgram

```
hist(meals1$price)
```

or a boxplot

```
boxplot(meals1)
```

Looks like we found another anomoly, let's look at the distribution of these values

```
unique(meals1$price)
meals1$price[meals1$price==400] <- 4.00
hist(meals1$price)
summary(meals1)
```

We have 2 NAs for different reasons - will have to reconcile that later

remove white space from our character columns

```
meals1$Meal <- str_trim(meals1$Meal)
meals1$entree <- str_trim(meals1$entree)
meals1$Location <- str_trim(meals1$Location)
```

an empty space appeared in one of our columns, let's replace that with NA too

```
meals1$entree[meals1$entree==""] <- "NA"
```

Still need to make our character columns

```
meals1$Meal <- as.character(meals1$Meal)
meals1$Location <- as.character(meals1$Location)
```

We have this in a good position. let's make a copy of our dataframe and call it meals2. This serves as a kind of checkpoint. I like to do this after a particularly long step (like something that takes a few minutes to run) so if I have to go back, it has already been saved in memory and I won't have to re-do that step.

```
meals2 <- meals1[,]
```

Looking good, we could work with this, but let's do a little more standardization

```
names(meals2) <- tolower(names(meals2))
head(meals2)
```

now let's lower the entree desriptions - you would often do this on 1 or 2 cols for text analysis

```
meals2$entree <- tolower(meals2$entree)
```

what if you wanted to know the mean & median prices of a meal in the cafeteria? Should we factor in the NA's or not? If you want to exclude the NA's, write na.rm=TRUE. If you want NA to be treated as 0, na.rm=FALSE. Here, if we want a more accurate mean, we definitely want to remove the NA's.

```
mean(meals2$price, na.rm = TRUE)
```

what if you wanted to know what it would cost to eat in the cafeteria for a full week? Let's say you are setting your budget for the week and want to make sure you have enough money. What do we do about those NA's? Well, we don't want to remove them, because then we wouldn't have enough money to eat on those days. The best we can do is find the maximum price for an entree and use that in place of the NA's. If we really wanted to be safe, we could add 10%, but we'll skip that step here (you'll learn about mutate() in a future lab).

First find the max price

```
maxprice <- max(meals2$price, na.rm=TRUE)
print(maxprice)
```

Then make a new dataframe where you've filled in the missing values with that maxprice.

```
meals2$price_imputed <- meals2$price
meals2$price_imputed[is.na(meals2$price_imputed)]<- maxprice
```

Add up the price column

```
sum(meals2$price_imputed)
```

You can apply this same technique in different ways: you can fill in the missing values with the mean or median, or fill them all with zeros. The important thing is to justify why you are filling the values the way you are and report what you did.

Missing data handling is probably the most important part of data cleaning because it can greatly affect your results. And sometimes, the most important story is in the missing values.

Congratulations! You've finished the data cleaning lab. Please submit your code as an R file.