A Diamond in the Rough

Lab 1F

Directions: Follow along with the slides and answer the questions in **red** font in your journal.

## Messy data? Get used to it

* Since lab 1, the data we’ve been using has been pretty *clean*.
* Why do we call it *clean*?
  + Variables were named so we could understand what they were about.
  + There didn’t seem to be any *typos* in the values.
  + Numerical variables were considered numbers.
  + Categorical variables were composed of categories.
* Unfortunately, more often than not, data is *messy* until YOU clean it.
* In this lab, we’ll learn a few essentials for cleaning *dirty* data.

## Messy data?

* What do we mean by messy data?
* Variables might have *non-descriptive names*
  + *Var01*, *V2*, *a*, …
* Categorical variables might have *misspelled categories*
  + *“blue”*, *“Blue”*, *“blu”*, …
* Numerical variables might have been *input incorrectly*. For example, if we’re talk about people’s height in inches:
  + *64.7*, *6.86*, *676*, …
* Numerical variables might be *incorrectly coded* as categorical variables (Or vice-versa)
  + “64.7”, “68.6”, “67.6”

## The American Time Use Survey

* To show you what *dirty* data looks like, we’ll check out the *American Time Use Survey*, or *ATU* survey.
* What is ATU survey?
  + It’s a survey conducted by the US government (Specifically the Bureau of Labor Statistics).
  + They survey thousands of people to find out exactly what activities they do throughout a single day.
  + These thousands of people combined together give an idea about how much time the typical person living in the US spends doing various activities.

## Load and go:

* Type the following commands into your console:

data(atu\_dirty)

View(atu\_dirty)

* **Just by viewing the data, what parts of our ATU data do you think need cleaning?**

## Description of ATU Variables

* The description of the actual variables:
  + caseid: Anonymous ID of survey taker.
  + V1: The age of the respondent.
  + V2: The gender of the respondent.
  + V3: Whether the person is employed full-time or part-time.
  + V4: Whether the person has a physical difficulty.
  + V5: How long the person sleeps, in minutes.
  + V6: How long the survey taker spent on homework, in minutes.
  + V7: How long the respondent spent socializing, in minutes.

## New name, same old data

* To fix the variable names, we need to *assign* a new set of names in place of the old ones.
  + Below is an example of the rename function:

atu\_cleaner <- rename(atu\_dirty, age = V1,  
 gender = V2)

* **Use the example code and the variable information on the previous slide to rename the rest of the variables in atu\_dirty.**
  + Names should be short, contain no spaces and describe what the variable is related to. So use abbreviations to your heart’s content.

## Next up: Strings

* In programming, a *string* is sort of like a *word*.
  + It’s a value made up of *characters* (i.e. letters)
* The following are example of strings. Notice that each **string** has quotes before and after.

"string"

"A1B2c3"

"Hot Cocoa"

"0015"

## Numbers are words? (Sometimes)

* In some cases, R will treat values that look like *numbers* as if they were *strings*.
* Sometimes we do this on purpose.
  + For example, we can code Yes/No variables as "1"/"0".
* Sometimes we don’t mean for this to happen.
  + The *number of siblings* a person has should not be a string.
* Look at the structure of your data and the variable descriptions from a few slides back:
  + **Write down the variables that should be *numeric* but are improperly coded as *strings* or *characters*.**

## Changing strings into numbers

* To fix this problem, we need to tell R to think of our *“numeric”* variables as numeric variables.
* We can do this with the as.numeric function.
  + An example using this function is below:

as.numeric("3.14")

## [1] 3.14

* Notice: We started with a string, "3.14", but as.numeric was able to turn it back into a number.

## Mutating in action

* Look at the variables you thought should be *numeric* and select one. Then fill in the blanks below to see how we can correctly code it as a number:

atu\_cleaner <- mutate(atu\_cleaner,   
 age = as.numeric(age),  
 \_\_\_ = as.numeric(\_\_\_))

* **Once you have this code working, use a similar line of code to correctly code the other *numeric* variables as numbers.**

## Deciphering Categorical Variables

* We mentioned earlier that we sometimes code categorical variables as numbers.
  + For example, our gender variable uses "01" and "02" for "Male" and "Female", respectively.
* It’s often much easier to analyze and interpret when we use more descriptive categories, such as "Male" and "Female".

## Factors and Levels

* R has a special name for *categorical* variables, called *factors*.
* R also has a special name for the different *categories* of a *categorical* variable.
  + The individual categories are called *levels*.
* To see the levels of gender and their counts type:

tally(~gender, data = atu\_cleaner)

* **Use similar code as we used above to write down the levels for the three factors in our data.**

## A level by any other name…

* If we know that ‘01’ means ‘Male’ and ‘02’ means ‘Female’ then we can use the following code to recode the *levels* of *gender*.
* Type the following command into your console:

atu\_cleaner <- mutate(atu\_cleaner, gender =   
 recode(gender,   
 "01"="Male",   
 "02" = "Female"))

* This code is definitely a bit of a mouthful. Let’s break it down.

## Allow me to explain

atu\_cleaner <- mutate(atu\_cleaner, gender =   
 recode(gender, "01"="Male",   
 "02" = "Female"))

* This code is saying:
  + Replace my current version of atu\_cleaner…
  + with a mutated one where …
  + the gender variable’s levels …
  + have been recoded…"
  + where "01" will now be "Male"…
  + and "02" will now be "Female".

## Finish it off!

* **Recode the categorical variable about whether the person surveyed had a physical challenge or not. The coding is currently:**
  + "01": Person surveyed *did not* have a physical challenge.
  + "02": Person surveyed *did* have a physical challenge.
* **Write a script that:**
  1. Loads the atu\_dirty data set
  2. Cleans the the data as we have in this lab
  3. Saves a copy of the cleaned data (see next slide).

## The final lines

* The last few lines of your script are extremely important because they will save all of your work.
* Be sure to View your data and check its structure to make sure it looks clean and tidy before saving.

Run the code below:

atu\_clean <- atu\_cleaner

* This code will create a new data frame in your Environment called atu\_clean which is a final copy of atu\_cleaner
  + If atu\_clean is swept from your Environment all of the changes you made will NOT be saved
  + You would need to re-run the script to clean the data again
* To permanently save your changes you need to save the file as an R data file or .Rda

Run the code below:

save(atu\_clean, file = "atu\_clean.Rda")

* Look in your Files pane for the atu\_clean.Rda file
  + This is as permanent copy of your clean atu data
  + To load the data onto your Environment click on the file
  + A pop-up window confirming the upload will appear

## Flex your skills

* Now that you have learned some cleaning data basics, it’s time to revisit the food data.

Run the code below:

histogram(~calories | healthy\_level, data = food)

* **Use the as.factor() function to convert healthy\_level into a categorical variable and re-run the histogram function.**

Notice that the healthy\_level categories are now numbers as opposed to tick-marks. This is an improvement but an even better solution would be to recode the categories.

* **Recode the healthy\_level categories and re-run the histogram function.**
  + “1” = “Very Unhealthy”
  + “2” = “Unhealthy”
  + “3” = “Neutral”
  + “4” = “Healthy”
  + “5” = “Very Healthy”
* If your food data is cleared from your Environment, the changes that you made to the healhty\_level variable will not be saved.
* To save your changes permanently save your food file as an R data file.