A Diamond in the Rough (Part b)

Unit 1 - Lab 7b

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

Last time . . .

- We loaded our **American Time Use Survey** data and found that it had lots of problems.
  - The variable **names** weren't very descriptive.
  - Our *numerical variables* were miss-specified as **strings** or **characters**
- **Explain the difference between the *string* 118 and the *number* 118.**

How did we fix these problems?

- We Loaded our data:

```
data(atus_dirty)
```

- **Run this line of code**

How did we fix these problems?

- We renamed our variables

```
names(atus_dirty) <- c("caseid",
                       "age",
                       "gender",
                       "fulltime_emp",
                       "phys_challenge",
                       "sleep",
                       "homework",
                       "socializing")
```

- **Run this line of code**

How did we fix these problems?

- We changed our **string** numbers back into **numerical** numbers:

```
atus_dirty <-
  transform(atus_dirty,
            age = as.numeric(age),
            sleep = as.numeric(sleep),
            homework = as.numeric(homework),
            socializing =
              as.numeric(socializing))
```

- **Run this line of code**

So what's next?

- Let's take a look at our data to find our next problem

```r
View(atus_dirty)
```

- **What do you notice about the `gender` and `phys_challenge` variables?**
- Recall that the variables tell us:
  - `gender`: The gender of the respondent.
  - `phys_challenge`: Whether the person has a physical difficulty.

Deciphering Categorical Variables

- Clearly, **gender** is a categorical variable but it's categories are represented by numbers.
  - This isn't necessarily a huge problem, but our data would be much clearer if we could replace the numbers `"01"` and `"02"` with `"Male"` and `"Female"`.
- The sames is true of the **phys_challenge** variables.

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** type:

```r
with(atus_dirty, levels(gender))
```

What's with with()?

```r
with(atus_dirty, levels(gender))
```

- This line of code says:
  - "With our atus_dirty data. . . "
  - ". . . print out the levels of. . . "
  - ". . . the variable `gender`."

What's with() phys_challenge?

- **Using the method from the last slide, write down the `levels` of the `phys_challenge` variable.**

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 revalue the **`levels`** of *gender*.
- Type the following command into your console:

```r
require(plyr)
```

```
## Loading required package: plyr
##
## Attaching package: 'plyr'
##
## The following object is masked from 'package:mosaic':
##
##     count
##
## The following objects are masked from 'package:dplyr':
##
##     arrange, desc, failwith, id, mutate, summarise, summarize
```

```r
atus_dirty <-
  transform(atus_dirty,
            gender =
              revalue(gender,
                      c("01" = "Male",
                        "02" = "Female")))
```

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

Allow me to explain

```r
atus_dirty <-
  transform(atus_dirty,
            gender =
              revalue(gender,
                      c("01" = "Male",
                        "02" = "Female")))
```

```
## The following `from` values were not present in `x`: 01, 02
```

- This code is saying:
  - “Replace my current version of `atus_dirty`…”
  - “… with a transformed one where …”
  - “… the `gender` variables levels …”
  - “… have been revalued…”
  - “… where ‘01’ will now be ‘Male’…”
  - “… and ‘02’ will now be ‘Female’.”

Factors and Levels

- `View` your data again and look at the values for **gender**
- **Rename the values of the variable `phys_challenge` where**
  - **‘01:** No difficulty
  - **‘02:** Has difficulty

Ta-da!

- It took some work, but you should have a data set you can be proud of.
- Let’s rename our data now that it’s clean:

```
atus_clean <- atus_dirty
```

- And let’s also take a moment to admire it:

```
View(atus_clean)
```