Quantitative Data Analysis

Andy Grogan-Kaylor

2023-04-05

Contents

1 Introduction **1**

2 Some Tools for Analysis **2**

3 Our Data **2**

4 Cleaning Data **2**
4.1 Excel and Google Sheets . . . 3
4.2 R . . . 3
4.3 Stata . . . 5

5 Simple Analysis **6**
5.1 Excel and Google Sheets . . . 7
5.2 R . . . 7
5.3 Stata . . . 7

1 Introduction

A great deal of data analysis and visualization involves the same core set of steps.

$$\text{have a question} \rightarrow \text{get data} \rightarrow \text{process and clean data} \rightarrow \text{analyze data}$$

Below we describe some simple data cleaning, and simple analysis with 4 tools: Excel, Google Sheets, R, and Stata.

2 Some Tools for Analysis

Tool	Cost	Ease of Use	Analysis Capabilities	Suitability for Large Data	Keep Track of Complicated Workflows
Excel	Comes installed on many computers	Easy	Limited	Difficult when N > 100	Difficult to Impossible
Google Sheets	Free with a Google account	Easy	Limited	Difficult when N > 100	Difficult to Impossible
R	Free	Challenging	Extensive	Excellent with large datasets	Yes, with script
Stata	Some cost	Learning Curve but Intuitive	Extensive	Excellent with large datasets	Yes, with command file

3 Our Data

We take a look at our *simulated* data, which has an `id` number, `age`, and `happiness` (on a 5 point scale, with 5 being the happiest.)

id	group	age	happy	somethingelse
1	Group B	66.77	-99	0.5
2	Group A	200	5	0.6805
3	Group A	21.42	-99	-0.4052
4	Group A	45.29	4	-1.79
5	Group B	39.77	2	-0.9557
6	Group B	53.59	4	1.734

Notice that…

- There are variables in which we may not have interest (e.g. `somethingelse`).
- None of the variables have informative *variable labels*. We have to guess at what the variables mean.
- Variables do not seem to have informative *value labels*. While somewhat intuitive, we have to guess at what the values mean.
- Someone appears to 200 years old.
- There appear to be missing values in the variable happy that need to be recoded.

4 Cleaning Data

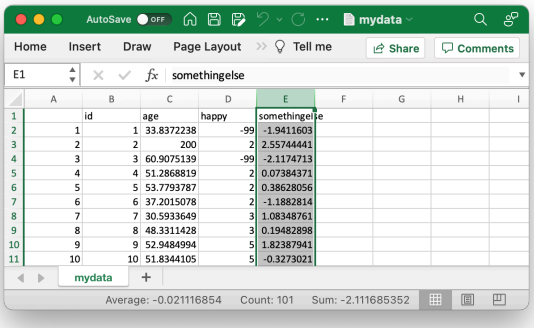
There are some basic data cleaning steps that are common to many projects.

1. Only keep the variables of interest.
2. Add variable labels (if we can).
3. Add value labels (if we can).
4. Recode outliers, values that are errors, or values that should be coded as missing

4.1 Excel and Google Sheets

4.1.1 1. Only keep the variables of interest.

Select the column, or columns, of data that you wish to remove, and right click, or control click, to delete them.



	A	B	C	D	E	F	G	H	I
1		id	age	happy	somethingelse				
2	1	1	33.8372238	-99	-1.9411603				
3	2	2	200	2	2.55744441				
4	3	3	60.9075139	-99	-2.1174713				
5	4	4	51.2868819	2	0.07384371				
6	5	5	53.7793787	2	0.38628056				
7	6	6	37.2015078	2	-1.1882814				
8	7	7	30.5933649	3	1.08348761				
9	8	8	48.3311428	3	0.19482898				
10	9	9	52.9484994	5	1.82387941				
11	10	10	51.8344105	5	-0.3273021				

4.1.2 2. Add variable labels (if we can).

We are unable to add informative labels to **variables** in Excel or Google Sheets.

4.1.3 3. Add value labels (if we can).

We are unable to add informative labels to **values** in Excel or Google Sheets.

4.1.4 4. Recode outliers, values that are errors, or values that should be coded as missing.

We are likely going to have to use **find and replace** to manually replace problematic values. For example, we will want to replace the `200` in the `age` column with a `.` or `NA` for missing. Similarly, we will want to replace the values of `-99` in the `happy` column with a `.` or `NA` for missing.

For small data sets, this will not be difficult, but for larger data sets–especially data with many different kinds of values that need to be recoded–this process will become more difficult and cumbersome.

4.2 R

Much of R's functionality is accomplished through writing *code*, that is saved in a *script*. Notice how–as our tasks get more and more complicated–the saved script provides documentation for the decisions that we have made with the data.

4.2.1 1. Only keep the variables of interest.

We can easily accomplish this with the `subset` function

```
mynewdata <- subset(mydata,
                    select = c(id, group, age, happy))
```

id	group	age	happy
1	Group B	66.77	-99
2	Group A	200	5

id	group	age	happy
3	Group A	21.42	-99
4	Group A	45.29	4
5	Group B	39.77	2
6	Group B	53.59	4

4.2.2 2. Add variable labels (if we can).

Adding *variable labels* is not well established in R. There are libraries that can add variable labels for some purposes, but not every library in R recognizes *variable labels*.

4.2.3 3. Add value labels (if we can).

In contrast, *value labels* are straightforward in R, and can be accomplished by creating a *factor variable*. Below we demonstrate how to do this with the happy variable.

```
mynewdata$happyFACTOR <- factor(mynewdata$happy,
                                levels = c(1, 2, 3, 4, 5),
                                labels = c("Very Unhappy",
                                           "Somewhat Unhappy",
                                           "Neutral",
                                           "Somewhat Happy",
                                           "Very Happy"))
```

id	group	age	happy	happyFACTOR
1	Group B	66.77	-99	NA
2	Group A	200	5	Very Happy
3	Group A	21.42	-99	NA
4	Group A	45.29	4	Somewhat Happy
5	Group B	39.77	2	Somewhat Unhappy
6	Group B	53.59	4	Somewhat Happy

4.2.4 4. Recode outliers, values that are errors, or values that should be coded as missing.

We can easily accomplish this using Base R's syntax for recoding: `data$variable[rule] <- newvalue`.

```
mynewdata$age[mynewdata$age >= 100] <- NA # recode > 100 to NA

mynewdata$happy[mynewdata$happy == -99] <- NA # recode -99 to NA
```

id	group	age	happy	happyFACTOR
1	Group B	66.77	NA	NA
2	Group A	NA	5	Very Happy
3	Group A	21.42	NA	NA
4	Group A	45.29	4	Somewhat Happy
5	Group B	39.77	2	Somewhat Unhappy
6	Group B	53.59	4	Somewhat Happy

4.3 Stata

Much of Stata's functionality is accomplished through writing *code*, that is saved in a *script*, which Stata calls a *do file*. Notice how–as our tasks get more and more complicated–the saved script provides documentation for the decisions that we have made with the data.

4.3.1 1. Only keep the variables of interest.

This is easily accomplished with Stata's `drop` command. We could also choose to `keep` our variables of interest.

```
drop somethingelse // drop extraneous variable(s)
```

4.3.2 2. Add variable labels (if we can).

Variable labels can easily be added in Stata.

```
label variable age "Respondent's Age" // variable label for age

label variable happy "Happiness Score" // variable label for happy

describe // describe the data

Contains data from mydata.dta
 Observations:           100
    Variables:             4
-------------------------------------------------------------------------------
Variable      Storage   Display    Value
    name         type    format    label      Variable label
-------------------------------------------------------------------------------
id                long    %9.0g                 id
group             long    %9.0g      group      group
age               double  %9.0g                 Respondent's Age
happy             double  %9.0g                 Happiness Score
-------------------------------------------------------------------------------
```

4.3.3 Add value labels (if we can)

Value labels are a natural part of Stata.

```
label define happy /// create value label for happy
5 "Very Unhappy" ///
4 "Somewhat Unhappy" ///
3 "Neutral" ///
2 "Somewhat Happy" ///
1 "Very Happy"

label values happy happy // assign value label happy to variable happy
```

Note that the code above is actually a *mistake*. `5` should be assigned to *Very Happy*, not to *Very Unhappy*. `1` should be assigned to *Very Unhappy*. Having our thinking written down in the form of code or syntax allows us to even see that this mistake has been made–and to ultimately correct it–when we are double checking our workflow.

```
list in 1/10 // list first 10 lines of data
```

```
     +-----------------------------------------+
     | id     group        age             happy |
     |-----------------------------------------|
  1. |  1   Group A   47.99102              -99 |
  2. |  2   Group A        200   Somewhat Unhappy |
  3. |  3   Group B   49.13296              -99 |
  4. |  4   Group A   36.66538   Somewhat Unhappy |
  5. |  5   Group B   57.67317           Neutral |
     |-----------------------------------------|
  6. |  6   Group A   54.10485        Very Happy |
  7. |  7   Group A   39.23244      Very Unhappy |
  8. |  8   Group B   44.18384   Somewhat Unhappy |
  9. |  9   Group A   35.08472           Neutral |
 10. | 10   Group A   52.86285   Somewhat Unhappy |
     +-----------------------------------------+
```

4.3.4 4. Recode outliers, values that are errors, or values that should be coded as missing

```
recode age (100 / max = .) // recode ages > 100
```

```
recode happy (-99 = .) // recode -99 to missing
```

```
list in 1/10 // list first 10 lines of data
```

```
     +-----------------------------------------+
     | id     group        age             happy |
     |-----------------------------------------|
  1. |  1   Group A   47.99102                 . |
  2. |  2   Group A         .   Somewhat Unhappy |
  3. |  3   Group B   49.13296                 . |
  4. |  4   Group A   36.66538   Somewhat Unhappy |
  5. |  5   Group B   57.67317           Neutral |
     |-----------------------------------------|
  6. |  6   Group A   54.10485        Very Happy |
  7. |  7   Group A   39.23244      Very Unhappy |
  8. |  8   Group B   44.18384   Somewhat Unhappy |
  9. |  9   Group A   35.08472           Neutral |
 10. | 10   Group A   52.86285   Somewhat Unhappy |
     +-----------------------------------------+
```

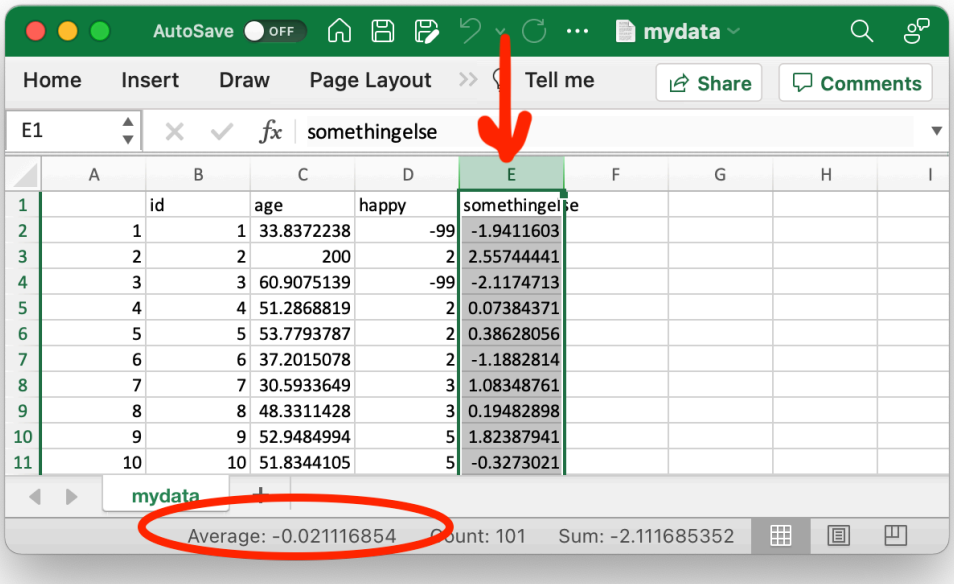
5 Simple Analysis

Our first step in analysis is to discover what kind of variables we have. We need to make a distinction between *continuous variables* that measure things like mental health or neighborhood safety, or age, and *categorical variables* that measure non-ordered categories like religious identity or gender identity.

- For continuous variables, it is most appropriate to take the *average* or *mean*.
- For categorical variables, it is most appropriate to generate a *frequency table*.

5.1 Excel and Google Sheets

In Excel and Google Sheets, our ability to do data *analysis* is very limited. In general, we are only able to easily calculate the *average* of *continuous variables*. There are various *add-ins* that can calculate other quantities, but their availability, usability, and ongoing stable development, tends to be inconsistent.



	A	B	C	D	E	F	G	H	I
1		id	age	happy	somethingelse				
2	1	1	33.8372238	-99	-1.9411603				
3	2	2	200	2	2.55744441				
4	3	3	60.9075139	-99	-2.1174713				
5	4	4	51.2868819	2	0.07384371				
6	5	5	53.7793787	2	0.38628056				
7	6	6	37.2015078	2	-1.1882814				
8	7	7	30.5933649	3	1.08348761				
9	8	8	48.3311428	3	0.19482898				
10	9	9	52.9484994	5	1.82387941				
11	10	10	51.8344105	5	-0.3273021				

5.2 R

As a mostly command based language, R relies on the idea of `do_something(dataset$variable)`.

```
summary(mynewdata$age) # descriptive statistics for age
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   21.42   44.31   50.21   49.81   56.22   71.57       1
```

```
table(mynewdata$group) # frequency table of group
```

```
##
## Group A Group B
##      49      51
```

5.3 Stata

As a mostly command based language, Stata relies on the idea of `do_something variable(s), options`.

```
summarize age // descriptive statistics for age
```

```
    Variable |        Obs        Mean    Std. dev.       Min        Max
-------------+---------------------------------------------------------
         age |         99    47.30858    9.615174   29.53682   71.51345
```

```
tabulate group // frequency table of group

      group |      Freq.     Percent        Cum.
------------+-----------------------------------
    Group A |         46       46.00       46.00
    Group B |         54       54.00      100.00
------------+-----------------------------------
      Total |        100      100.00
```