Basic Data Management

Dr. Sarah Hunter

4/8/2020

Dealing with Real World Data

Working with real data can be tricky. Usually, the data you want either come as part of a dataset that is too big with which to work, or they come as several different data sources. Perhaps you will also need to transform a variable (e.g. take the log, divide by millions, etc.). There are even cases where you will need to change the value labels of a variable. This R Help file will show you how to do these things. We will focus here on simple Data Management using mostly R's base commands. Later, if you are interested, I can point you toward more advanced packages for data management called `dplyr` and `tidyr`.

Subsetting Data

Do you have too much data? Do you only want 1-2 variables from a dataset? Do you only want certain years of observations? The following section will show you how to *subset* data in several ways to tackle these problems.

Subsetting by Selecting Certain Variables

Some datasets such as the Quality of Governance and the VDem datasets are huge datasets that contain hundreds of variables. Sometimes, you only need a few of those variables. To choose only a certain number of variables, you can use the `subset` command. An example, using the Quality of Governance Data is below. The `subset` command is usually structured by first creating a new object (qog.subset.ciri in the example below), then then telling R which dataset you would like to split (qog in the example). From there, you then tell R how you would like to split the data. In our example, we are splitting the data by only keeping certain variables.

In the example, I take the Quality of Governance data, which, as you can see, is quite large, and only extract the variables that I need from it (in this case, variables pertaining to the CIRI human rights measure). The option to use here is the `select` option. This tells R to choose one or more columns. If you have more than one variable to select, you must use the `c()` designation. This tells R that you have more than one column to select.

```
#Setting Working Directory
setwd("/Users/sarahhunter/Documents")

#Loading the data

qog<-read.csv("qog_std_ts_jan20.csv")

#Making sure the data are loaded correctly
dim(qog)
```

```
## [1] 15614  2086
```

```
#Subsetting by Selecting Certain Variables

qog.subset.ciri<-subset(qog, select = c(ccode, cname, year, ciri_assn, ciri_dommov,
     ciri_formov, ciri_injud, ciri_physint, ciri_worker, ciri_speech))

summary(qog.subset.ciri)
```

```
##      ccode          cname                year        ciri_assn
##  Min.   :  4.0   Length:15614       Min.   :1946   Min.   :0.000
##  1st Qu.:218.0   Class :character   1st Qu.:1964   1st Qu.:0.000
##  Median :442.0   Mode  :character   Median :1982   Median :1.000
##  Mean   :458.1                      Mean   :1982   Mean   :1.113
##  3rd Qu.:694.0                      3rd Qu.:2001   3rd Qu.:2.000
##  Max.   :999.0                      Max.   :2019   Max.   :2.000
##                                                    NA's   :9498
##   ciri_dommov     ciri_formov      ciri_injud     ciri_physint
##  Min.   :0.000   Min.   :0.000   Min.   :0.000   Min.   :0.000
##  1st Qu.:1.000   1st Qu.:1.000   1st Qu.:0.000   1st Qu.:3.000
##  Median :2.000   Median :2.000   Median :1.000   Median :5.000
##  Mean   :1.518   Mean   :1.453   Mean   :1.122   Mean   :4.981
##  3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:7.000
##  Max.   :2.000   Max.   :2.000   Max.   :2.000   Max.   :8.000
##  NA's   :8953    NA's   :8952    NA's   :8987    NA's   :9520
##   ciri_worker     ciri_speech
##  Min.   :0.000   Min.   :0.000
##  1st Qu.:0.000   1st Qu.:0.000
##  Median :1.000   Median :1.000
##  Mean   :0.912   Mean   :0.974
##  3rd Qu.:1.000   3rd Qu.:2.000
##  Max.   :2.000   Max.   :2.000
##  NA's   :9497    NA's   :9495
```

Subsetting by Dropping Certain Variables

We can also subset our datasets by dropping certain variables. We can do this with the same `select` option. The difference is that we put the `-` before each variable we want to drop. In the example below, I want to drop CIRI's independence of the judiciary from my dataset above. Then we could drop it by:

```
qog.subset.ciri2<-subset(qog.subset.ciri, select = -ciri_injud)
summary(qog.subset.ciri2)
```

```
##      ccode          cname                year        ciri_assn
##  Min.   :  4.0   Length:15614       Min.   :1946   Min.   :0.000
##  1st Qu.:218.0   Class :character   1st Qu.:1964   1st Qu.:0.000
##  Median :442.0   Mode  :character   Median :1982   Median :1.000
##  Mean   :458.1                      Mean   :1982   Mean   :1.113
##  3rd Qu.:694.0                      3rd Qu.:2001   3rd Qu.:2.000
##  Max.   :999.0                      Max.   :2019   Max.   :2.000
##                                                    NA's   :9498
##   ciri_dommov     ciri_formov     ciri_physint    ciri_worker
##  Min.   :0.000   Min.   :0.000   Min.   :0.000   Min.   :0.000
##  1st Qu.:1.000   1st Qu.:1.000   1st Qu.:3.000   1st Qu.:0.000
##  Median :2.000   Median :2.000   Median :5.000   Median :1.000
##  Mean   :1.518   Mean   :1.453   Mean   :4.981   Mean   :0.912
```

```
##  3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:7.000   3rd Qu.:1.000
##  Max.   :2.000   Max.   :2.000   Max.   :8.000   Max.   :2.000
##  NA's   :8953    NA's   :8952    NA's   :9520    NA's   :9497
##   ciri_speech
##  Min.   :0.000
##  1st Qu.:0.000
##  Median :1.000
##  Mean   :0.974
##  3rd Qu.:2.000
##  Max.   :2.000
##  NA's   :9495
```

Subsetting By Selecting Certain Values of Variables

Sometimes, you only want to keep data based on the values of a certain variable. For this, you would need to use the mathematical operators. For example, if you only wanted to keep observations for the years 2000 and later, you can use the following code:

```
#Remember when using >, it means greater than.
#So if you want 2000 and later, you need to say greater than 1999

qog.ciri.2000<-subset(qog.subset.ciri2, year>1999)
summary(qog.ciri.2000)
```

```
##      ccode          cname                year        ciri_assn
##  Min.   :  4.0   Length:4220        Min.   :2000   Min.   :0.000
##  1st Qu.:218.0   Class :character   1st Qu.:2005   1st Qu.:0.000
##  Median :442.0   Mode  :character   Median :2010   Median :1.000
##  Mean   :458.1                      Mean   :2010   Mean   :1.181
##  3rd Qu.:694.0                      3rd Qu.:2014   3rd Qu.:2.000
##  Max.   :999.0                      Max.   :2019   Max.   :2.000
##                                                    NA's   :807
##   ciri_dommov     ciri_formov     ciri_physint    ciri_worker
##  Min.   :0.000   Min.   :0.000   Min.   :0.000   Min.   :0.0000
##  1st Qu.:1.000   1st Qu.:1.000   1st Qu.:4.000   1st Qu.:0.0000
##  Median :2.000   Median :2.000   Median :5.000   Median :1.0000
##  Mean   :1.503   Mean   :1.504   Mean   :5.094   Mean   :0.8396
##  3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:7.000   3rd Qu.:1.0000
##  Max.   :2.000   Max.   :2.000   Max.   :8.000   Max.   :2.0000
##  NA's   :777     NA's   :778     NA's   :805     NA's   :809
##   ciri_speech
##  Min.   :0.0000
##  1st Qu.:0.0000
##  Median :1.0000
##  Mean   :0.9537
##  3rd Qu.:1.0000
##  Max.   :2.0000
##  NA's   :809
```

You can filter the data by any condition that you would like. You simply need to use R's mathematical operators to obtain the values you want. The following table provides a selection of mathematical operators to use as conditions:

Operator	Description
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	exactly equal to
!=	not equal to

The following R code shows how to use those operators in the context of subsetting:

```
####Keeping observations before 2000####

qog99.1<-subset(qog, year<2000)

#or

qog99.2<-subset(qog, year<=1999)


####Keeping observations without missing values for Ciri Physical Integrity (ciri_physint)

qog.ciri.all<-subset(qog.subset.ciri, ciri_physint!=NA)

#Only Keeping Observations for Afghanistan

qog.ciri.afg<-subset(qog.ciri.2000, ccode==4)
```

You can even combine conditions in one line of code:

```
qog.ciri.2000.2<-subset(qog, year>1999, select =  c(ccode, cname, year, ciri_assn,
      ciri_dommov, ciri_formov, ciri_injud, ciri_physint,
      ciri_worker, ciri_speech))

head(qog.ciri.2000.2)
```

```
##    ccode       cname year ciri_assn ciri_dommov ciri_formov ciri_injud
## 55     4 Afghanistan 2000         0           0           1          0
## 56     4 Afghanistan 2001         0           0           0          0
## 57     4 Afghanistan 2002         0           0           0          0
## 58     4 Afghanistan 2003        NA          NA          NA         NA
## 59     4 Afghanistan 2004        NA          NA          NA         NA
## 60     4 Afghanistan 2005         1           0           0          0
##    ciri_physint ciri_worker ciri_speech
## 55            0           0           0
## 56            0           0           0
## 57            3           0           1
## 58           NA          NA          NA
## 59           NA          NA          NA
## 60            4           0           0
```

Merging Data

Sometimes the data you want come in two separate datasets. In this case, you would have to *merge* the two or more files together. The key thing to remember with merging is that the observations in both datasets must be defined or identified the same way. For example, if you are merging together a dataset about US states, you can merge state names with state names or state abbreviates with state abbreviations, but not state names with state abbreviates. When merging datasets with countries, you need should use country codes rather than the names of countries. Some country names are different depending on which language was used to write the name, the use of long form or short form names, etc. For example, some datasets might list "South Korea", where others would list "Republic of Korea". R does not know that these two are the same observation. Therefore, we usually use either country numeric codes from the Correlates of War (called COW) Dataset, or alphabetic codes. In this case, we are using the COW codes. So we know to match country code (ccode) 410 with all the 410 in the other datasets. I demonstrate the how to merge two data sets in the following code.

First, I make another dataset from the larger Quality of Governance data using the `subset` command. Then I use the `merge` command to combine the two datasets. For this command, you must specify which datasets are to be merged. The second part is to indicate by which variables R should match the two datasets. You can use up to two criteria. For this time series cross sectional data, we have both cross sectional units (countries) and time series units (years) to match. We do this with they `by.x` and `by.y` options. The order here is very important. The syntax should be followed exactly. If you put `ccode` first for the first dataset, it must also be first in the second dataset. Remember to keep your x's and y's straight.

```
#Creating a new dataset just for Polity scores

polity<-subset(qog, select=c(ccode, year, p_polity2))

head(polity)
```

```
##   ccode year p_polity2
## 1     4 1946       -10
## 2     4 1947       -10
## 3     4 1948       -10
## 4     4 1949       -10
## 5     4 1950       -10
## 6     4 1951       -10
```

```
merged_data<-merge(x=qog.subset.ciri, y=polity, by.x = c("ccode", "year"),
      by.y = c("ccode", "year"))

head(merged_data)
```

```
##   ccode year    cname ciri_assn ciri_dommov ciri_formov ciri_injud ciri_physint
## 1   100 1946 Bulgaria        NA          NA          NA         NA           NA
## 2   100 1947 Bulgaria        NA          NA          NA         NA           NA
## 3   100 1948 Bulgaria        NA          NA          NA         NA           NA
## 4   100 1949 Bulgaria        NA          NA          NA         NA           NA
## 5   100 1950 Bulgaria        NA          NA          NA         NA           NA
## 6   100 1951 Bulgaria        NA          NA          NA         NA           NA
##   ciri_worker ciri_speech p_polity2
## 1          NA          NA        -6
## 2          NA          NA        -7
## 3          NA          NA        -7
## 4          NA          NA        -7
## 5          NA          NA        -7
## 6          NA          NA        -7
```

Transforming Variables

Occasionally, you might need to transform a variable, that is, change the values. For example, if I were to use GDP in a model, does a one dollar increase in a country's GDP have much of an inpact on anything? You would get a very small regression coefficient if you were to include the unchanged value of GDP. However, if you were to make the values instead GDP in millions of dollars, then your coefficients would not be as small, and the increases would make more sense. Another example is including income in a model. Income is a highly skewed variable most of the time. Therefore, many scholars would include the log of income instead of income in dollars.

There are various ways to transforming a variable. Most of these transformations use R's arithmetic functions. To transform a variable, just use one of these operators:

Operator	Description
+	Addition
-	Subtraction
	Multiplication
/	Division
^	Exponentiation
log()	Natural log

Recoding Variables

Creating Dummy Variables

Sometimes, you need to create a dummy variable to capture a certain concept. For example, if we only car about democracies compare to the rest of countries, we can use the data from out above dataset to create a new variable that is just "democracy". the `ifelse` command is the easiest way to create a dummy variable. With the `ifelse` command, you first give a True/False condition. The next number is the value R should put if the condition is true. The last number is the value R should put if the condition is false. Below is an example of creating a "democracy" variable, where a country is considered a democracy if it is a 6 or above on the polity scale.

```
merged_data$democracy<-ifelse(merged_data$p_polity2>=6, 1, 0)

head(merged_data)
```

```
##   ccode year    cname ciri_assn ciri_dommov ciri_formov ciri_injud ciri_physint
## 1   100 1946 Bulgaria        NA          NA          NA         NA           NA
## 2   100 1947 Bulgaria        NA          NA          NA         NA           NA
## 3   100 1948 Bulgaria        NA          NA          NA         NA           NA
## 4   100 1949 Bulgaria        NA          NA          NA         NA           NA
## 5   100 1950 Bulgaria        NA          NA          NA         NA           NA
## 6   100 1951 Bulgaria        NA          NA          NA         NA           NA
##   ciri_worker ciri_speech p_polity2 democracy
## 1          NA          NA        -6         0
## 2          NA          NA        -7         0
## 3          NA          NA        -7         0
## 4          NA          NA        -7         0
## 5          NA          NA        -7         0
## 6          NA          NA        -7         0
```

Recoding Categorical Variables

Sometimes, you need to change the labels of a categorical variable. Perhaps these variables are coded with numbers and you would like value labels instead so R knows that this is a categorical, not continuous variable. For example, the CIRI scale for the freedom of foreign movement index (`ciri_formov`) is a three point scale where 0 means foreign travel is severely restricted, 1 means foreign movement is somewhat restricted, and 2 means foreign movement is unrestricted. We can get summary statistics from our freedom of foreign movement variable using the following code:

```
table(merged_data$ciri_formov)
```

```
##
##    0    1    2
##  912 1823 3927
```

We check this first to make sure that our recoding has been successful. To recode using R's `car` library, we need to use the following code that introduces the new command, `recode`. With this command, we first create a new object, then assign it values based on the value of the previous incarnation of the variable you are recoding.

```
library(car)
```

```
## Loading required package: carData
```

```
merged_data$new_formov<-recode(merged_data$ciri_formov, "0='severely_restricted';
                                1='somewhat_restricted'; 2='unrestricted'")

table(merged_data$new_formov)
```

```
##
## severely_restricted somewhat_restricted        unrestricted
##                 912                1823                3927
```

We can even create a new categorical variable from a range of values in a continuous variable. For example, you might care about the distinctions between democracies (Polity of 6 and above), autocracies (Polity of -6 and below), and anocracies (Polity of -5 to 5). We can use the same `recode` function to assign those labels to the appropriate range of Polity scores.

```
merged_data$regime<-recode(merged_data$p_polity2, "-10:-6='autocracy';
                           -5:5='anocracy'; 6:10='democracy'")

table(merged_data$regime)
```

```
##
##  anocracy autocracy democracy
##      2332      3401      3841
```

In the previous two examples, you can ignore the error message 'NAs introduced by coercion'. That simply means that you have NA values in the data, and therefore you have NAs in the resulting recoded variables.

Conclusion

You are now ready to tackle many data management problems! However, I this is the most basic code for data management and manipulation. There are many more ways to everything I have shown you here. I have simply shown the simplest way to accomplish simple tasks. As you encounter more and more complex data problems, more advanced data management will be needed. While these commands could handle most of

these problems, they sometimes become clunky and inefficient. This is why we continue to push forward and learn more about various R packages, especially the `dplyr` and `tidyr` packages.

As you work on your own projects, try these commands first. If they do not work or do not complete the task you need, then explore the other libraries. But until you are comfortable with R, these commands are perfectly sufficient.