Session 2: Data Manipulation

Andrew McCormack

2018-11-07

Review

Set your working directory and import data:

Set the working directory as the folder where your data file is stored:

```
setwd("/Users/andrewmccormack/Documents/DSC/")
civilwar <- read.csv("fearon03.csv")

# Use head() to get acquainted with the first few rows of data
head(civilwar)
```

```
##   X.1 X country year war onset    pop polity2 gdppc
## 1   1 1     USA 1945   0     0 140969      10 7.626
## 2   2 2     USA 1946   0     0 141936      10 7.654
## 3   3 3     USA 1947   0     0 142713      10 8.025
## 4   4 4     USA 1948   0     0 145326      10 8.270
## 5   5 5     USA 1949   0     0 147987      10 8.040
## 6   6 6     USA 1950   0     0 152273      10 8.772
##                          region   mtn oil   ethfrac cow
## 1 western democracies and japan -99.0   0 0.3569501 USA
## 2 western democracies and japan  23.9   0 0.3569501 USA
## 3 western democracies and japan  23.9   0 0.3569501 USA
## 4 western democracies and japan  23.9   0 0.3569501 USA
## 5 western democracies and japan  23.9   0 0.3569501 USA
## 6 western democracies and japan  23.9   0 0.3569501 USA
```

Because I set my working directory as `/Users/andrewmccormack/Documents/DSC/`, R will look for the file `fearon03.csv` in this folder.

Selecting variables in a dataframe

We can restrict the `civilwar` dataframe to include only those variables that are relevant to us. For instance, if I only want to look at the country, population, and GDP per capita variables, I can run the following command:

```
civilwar[ , c("country", "pop", "gdppc")]
```

We type in the variable names on the right-hand side of the comma inside the square brackets. We leave the left hand side of the comma blank to indicate that we want to keep all the rows (or observations) in the dataframe.

Note that this only produces output, it does not create a new dataframe. To save these three variables in their own dataframe, we need to assign it to a new object:

```
civilwar_b <- civilwar[ , c("country", "pop", "gdppc")]
```

Filtering observations in a dataframe

Perhaps I only want to look at data from wealthy countries. If this is the case, I need to filter the observations. Note that GDP per capita is measured in thousands of dollars. I will filter the observations in the `civilwar_b` dataframe I just created to include countries where GDP per capita (in thousands of US dollars) exceeds 10:

```
civilwar_b[civilwar_b$gdppc > 10, ]
```

Instead of GDP per capita, I can also filter the observations by country. We can use the `%in%` operator to identify certain elements of the country variable

```
civilwar_b[civilwar_b$country %in% c("MEXICO", "LATVIA", "BELARUS", "YEMEN"), ]
```

Creating new variables

We use the assignment operator to create new variables in R. Let's create a new variable that is the log of population. First, note that we can take the natural log of a numeric variable by using the `log()` function:

```
log(civilwar$pop)
```

This gives us a vector that is the same length as the number of observations in the dataframe. We can incorporate it into the dataframe itself using the assignment operator. Let's call the new variable `pop_log`:

```
civilwar$pop_log <- log(civilwar$pop)
```

Let's create an entirely new variable using polity scores. Polity scores range from -10 (very undemocratic) to 10 (very democratic). We will count scores above 5 as democratic and scores below 5 as undemocratic. To do this, we use the `ifelse()` function:

```
civilwar$democracy <- ifelse(test = civilwar$polity2 > 5, yes = 1, no = 0)
```

Note that `test` returns a `TRUE` or `FALSE` value for each of the observations. We specify `yes` as the value we want the variable to take when the condition is `TRUE`, and `no` as the value when the condition is `FALSE`. The condition in this case is whether the `polity2` value is greater than 5.

We can also use `ifelse()` to create a character variable (as opposed to numeric):

```
civilwar$democracy_chr <- ifelse(test = civilwar$polity2 > 5, yes = "democracy", no = "not democracy")
```

Recoding variables

In most datasets you will work with, there will be issues in the dataset that requires fixing. This stage is called data cleaning. For instance, in the `mtn` variable, there are values of `-99`. This variable is a measure of mountainous terrain, and -99 is not a valid measure of this—it signifies that there is missing data. We don't want R to confuse this as a real value, so we need to recode it to `NA` (missing):

```
civilwar$mtn[civilwar$mtn == -99] <- NA
```

Alternatively, we can recode using `ifelse()`. In this case, we tell R to recode the variable to `NA` if it is `-99`, otherwise keep the original value:

```
civilwar$mtn2 <- ifelse(civilwar$mtn == -99, NA, civilwar$mtn)
```

Packages in R

R comes pre-installed with a wide variety of functions, all these functions are part of "base R". However, because R is open-source, there are many additional packages that have been developed to improve these function and add new functions. There are thousands of packages to choose from, making R a very powerful tool for data science. We will focus on one of the most popular and useful packages: `dplyr`. `dplyr` allows us to execute much of the data manipulation we have done already, but in more intuitive and elegant way. `dplyr` has a wide variety of functions, making all sorts of data manipulation possible. We will focus on the `select()`, `filter()`, `mutate()`, `summarise()` functions.

`select()`

Selecting variables in a dataframe is more straightforward with `dplyr`'s `select()` function than with base R. To select the variables `country`, `pop` and `gdppc` (the same variables as above), we run the following command:

```
select(civilwar, country, pop, gdppc)
```

In `select()`, we start with the name of the dataframe and then type in the variables we want to extract.

To select all the columns *except* a specific column, use the "-" (subtraction operator):

```
select(civilwar, -country, -pop, -gdppc)
```

Now we have a dataframe with all variables except `country`, `pop` and `gdppc`.

`filter()`

To select only certain rows of a dataframe based on some condition, we use `dplyr`'s `filter()` function. Let's use the same example from above and filter to observations to include to include only those countries where GDP per capita (in thousands of US dollars) exceeds 10:

```
filter(civilwar, gdppc > 10)
```

We can also add the further restriction that the countries must also be non-democratic (which we will define as below 5 on the polity scale):

```
filter(civilwar, gdppc > 10, polity2 < 5)
```

`mutate()`

To add new columns to the data frame, we use `dplyr`'s `mutate()` function. Let's create a new variable that is the log of population, like we did above.

```
mutate(civilwar, pop_log = log(pop))
```

This function returns the original dataframe with the addition of our new `log_pop` variable.

We can use mutate to create multiple new variables at once. Let's save these new variables in a new dataframe called `civilwar_c`:

```
mutate(civilwar,
       pop_log = log(pop),
       gdppcln = log(gdppc),
       democracy = ifelse(polity2 > 5, 1, 0))
```

We have created three new variables here: logged population, logged GDP per capita, and a binary indicator for democracy.

The pipe operator: %>%

The pipe operator allows you to pipe the output from one function to the input of another function. Let's look at an example. Perhaps we want to (1) select the `country`, `pop` and `gdppc` variables, (2) filter the observations to only include Mexico, Latvia, Belarus, and Yemen and (3) create a new logged population variable. One way to do this is by creating new objects at each step:

```
cw <- filter(civilwar, country %in% c("MEXICO", "LATVIA", "BELARUS", "YEMEN"))
cw2 <- select(cw, country, pop, gdppc)
cw3 <- mutate(cw2, pop_log = log(pop))
```

Alternatively, we could nest the functions inside one another

```
mutate(select(filter(civilwar, country %in% c("MEXICO", "LATVIA", "BELARUS", "YEMEN")),
              country, pop, gdppc), pop_log = log(pop))
```

Neither of these methods are terribly efficient. Let's try using the pipe operator with the first step.

```
civilwar %>%
  select(country, pop, gdppc)
```

You can see that the pipe carries the `civilwar` dataset through to the `select()` function. We can go further than this and pipe these results into other functions:

```
civilwar %>%
  select(country, pop, gdppc) %>%
  filter(country %in% c("MEXICO", "LATVIA", "BELARUS", "YEMEN")) %>%
  mutate(pop_log = log(pop))
```

Take note of what we did here: we took the civilwar dataset, selected the `country`, `pop` and `gdppc` variables, then we piped this restricted dataset into the filter function, where we selected Mexico, Latvia, Belarus, and Yemen. Next, we piped the selected and filtered dataset into the `mutate()` function to add a logged population variable.

Summarize function

`dplyr`'s `summarise()` function is also a powerful tool for creating descriptive statistics. Let's get the mean value for the `polity2` variable:

```
civilwar %>%
  summarise(democracy_mean = mean(polity2, na.rm = T))
```

Because this dataset has observations for many years, let's get the population mean for the year 1999. First, we filter the data to include only observations from 1999, then we run the `summarise()` function:

```
civilwar %>%
  filter(year == 1999) %>%
  summarise(democracy_mean = mean(polity2, na.rm = T))
```

Let's compare this to an earlier year and see if the world got more democratic:

```
civilwar %>%
  filter(year == 1980) %>%
  summarise(democracy_mean = mean(polity2, na.rm = T))
```

It looks like the world got more democratic. We are happy about this. Yet, we suspect that these mean values are masking variation across the world's region. Luckily, we can use `group_by()` in combination with summary `summarise()` to get region specific means:

```
civilwar %>%
  filter(year == 1999) %>%
  group_by(region) %>%
  summarise(democracy_mean = mean(polity2, na.rm = T))
```

Our suspicions are confirmed, there is significant variation in democracy scores across different regions of the world. Perhaps we think that GDP per capita has some relationship with democracy. We can look at the mean values of democracy and GDP per capita at the same time:

```
civilwar %>%
  filter(year == 1999) %>%
  group_by(region) %>%
  summarise(democracy_mean = mean(polity2, na.rm = T),
            gdppc_mean = mean(gdppc, na.rm = T))
```