POL 345 Precept Week 9: Some Useful Commands

Jing Qian

2020/04/08

Welcome! In this note we will discuss how to do *linear regression* in R. Please note that we will be focusing on the programming side, and you are encouraged to check the lecture slides and the textbook for the statistical and substantive interpretation.

In this note, we will use the `resources.csv` dataset from precept exercise 7 for demonstration. First, let's read the dataset into R and remind ourselves what the dataset looks like:

1. Introduction

```
resources = read.csv("data/resources.csv")
head(resources)
```

```
##      cty_name year logGDPcp regime       oil       metal illit life
## 1 Afghanistan 1966       NA     -7        NA 0.004242876    NA   NA
## 2 Afghanistan 1967       NA     -7 0.2251016 0.003824710    NA 36.0
## 3 Afghanistan 1968       NA     -7 0.6578155          NA    NA   NA
## 4 Afghanistan 1969       NA     -7 0.8730071          NA    NA   NA
## 5 Afghanistan 1970       NA     -7 0.8196415          NA    88 37.2
## 6 Afghanistan 1971       NA     -7 0.7939780          NA    NA   NA
```

The most basic syntax to do linear regression in R is as the following:

```
lm(formula, data)
```

where `formula` specifies the regression model that you want to estimate. It is characterized by a set of variables names, connected by the tilde sign ~. The variable on the left of ~ is your dependent/outcome variable, while anything on the right of ~ is your independent/explaning variable(s).

In the case of a bivariate regression, say if we want to estiamte the following model:

$$Y_i = \alpha + \beta X_i + \varepsilon_i$$

The formula to estimate the model above would be: `Y ~ X`.

And the `data` argument of `lm()` specifies the name of dataset, which contains the variables that you would use to estimate a linear model.

2. Estimate a Bivariate Linear Regression Model

Let's say we want to use linear regression to examine the relationshi between GDP per capita (logged) and life expectancy from the `resources.csv` dataset. Specifically, we want to see whether we can predict life expectancy with the country's logged GDP per capita, with the following model:

$$\text{Life Expectancy}_i = \alpha + \beta \text{log GDP pc}_i + \varepsilon_i$$

We can use the following code to estimate this model:

```
my.lm = lm(life ~ logGDPcp, data = resources)
```

In the code above, we have asked R to do the following things:

1. Estimate the linear regression, with `life` as the dependent variable, and `logGDPcp` as the independent variable.
2. Use the `resources` data (therefore the `life` and `logGDPcp` columns in that) to estimate the model.
3. Store the results of the linear regression to a new object called `my.lm`

3. Access Results from `lm()`

As you will recall from the lecture, usually we are interested in various results of a linear regression, for different reasons. Specifically, we want to access at least two sets of results:

1. The estimated coefficients: α and β
2. The R^2

If you simply print the result from `lm()`, which in our case is stored in the object `my.lm`, you will see a summary of the regression:

```
my.lm
```

```
##
## Call:
## lm(formula = life ~ logGDPcp, data = resources)
##
## Coefficients:
## (Intercept)     logGDPcp
##       2.656        7.760
```

3.1 Get Coefficients

Although this is convenient as a quick check, we might want to directly access the coefficients, or store the coefficients into a separate object, or only get part of the coefficients. To do so, we can first use the `coef()` function to only get the coefficients of our estimated linear model. For example:

```
coef(my.lm)
```

```
## (Intercept)    logGDPcp
##    2.655596    7.760333
```

The `coef()` function takes the result of `lm()` as the argument (which in our case is the object `my.lm`), and returns a vector of numerical values. We can store this vector to a new object, say called `my.coef`:

```
my.coef = coef(my.lm)
#
class(my.coef)
```

```
## [1] "numeric"
```

```
names(my.coef)
```

```
## [1] "(Intercept)" "logGDPcp"
```

We can access the specific coefficient that we are interested in by indexing this vector, either by name or by location. Say if we want to get the coefficient β, which is the coefficient for the `logGDPcp` variable:

```
#The folllowing two ways are equivalent
my.coef[2]
```

```
## logGDPcp
## 7.760333
```

```
my.coef["logGDPcp"]
```

```
## logGDPcp
## 7.760333
```

3.2 Get R^2

What if you want to get the value of R^2? The following code will do the work:

```
summary(my.lm)$r.squared
```

```
## [1] 0.7169483
```

Specifically, we first ue the `summary()` function to get the results of our regression, which contains various values. Then we use `$r.squared` to get the element which contains the value of R^2.

Again, we can store this value into a new object, say called `my.r2`:

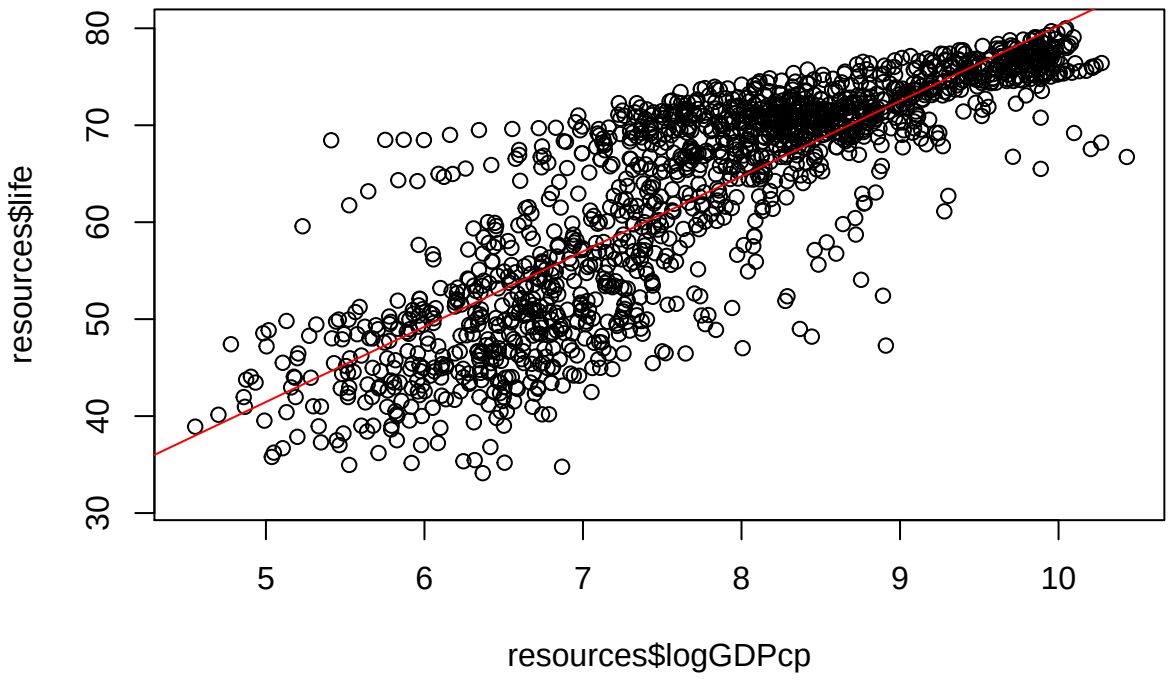
```
my.r2 = summary(my.lm)$r.squared
```

4. Visualize Results

Now we have estimated a linear model, and get the results. Since we are doing a bivariate regression (i.e. examining the relationship between the dependent variable and another independent variable), we can visualize the results, in which the estimated model will be represented by a line.

Luckily, we can very easily plot the line representing our regression with the `abline()` function, and use the result from our regression (`my.lm`) as the argument:

```
#First we produce a scatter plot
##Note that we usually put the dependent variable on the y-axis
plot(x = resources$logGDPcp,
     y= resources$life)
##Add the regression line
abline(my.lm, col = "red")
```



5. Use Results from `lm()` to predict

We might also want to use our regression results to *predict*. Let's first check what are our estimated coefficients:

```
my.coef
```

```
## (Intercept)    logGDPcp
##    2.655596    7.760333
```

From the results above, our regression estimates the following relationship between logged GDP per capita and life expectancy:

$$\text{Life Expectancy}_i = 2.655596 + 7.760333 \times \text{logged GDP per capita}$$

What if we have another dataset that contains values of `logGDPcp`, but not `life`? We can use our regression results to predict the value of `life` with the value of `logGDPcp`. To do that, we use the `predict()` function. Say we have a new dataset `new.resources` that contains the value of `logGDPcp` for 10 observations:

```
new.resources
```

```
##     logGDPcp
## 1  25.194152
## 2  35.288780
## 3  67.000697
## 4   3.277112
## 5  13.265485
## 6  25.170359
## 7   4.466717
```

```
## 8   6.152726
## 9  12.141280
## 10 23.195572
```

Since we already have our estimated model, we can calculated our predicted values of `life` with the coefficients. Using the coefficients we have stored in `my.coef`, we can do it with the following code:

```
my.coef[1] + new.resources$logGDPcp * my.coef[2]
```

```
##  [1] 198.17061 276.50828 522.60332  28.08708 105.60018 197.98596  37.31881
##  [8]  50.40280  96.87597 182.66096
```

However, a more direct way to do that is by using the `predict()` function, where the first argument is our objct of the result (`my.lm`), and set the value of the `newdata` argument to the dataset contains our new values of `logGDPcp`:

```
predict(my.lm, newdata = new.resources)
```

```
##         1         2         3         4         5         6         7
## 198.17061 276.50828 522.60332  28.08708 105.60018 197.98596  37.31881
##         8         9        10
##  50.40280  96.87597 182.66096
```

However, please note that when you are using `predict()`, the newdata must contains the column with the same name (`logGDPcp`) that you use to estimate the model. Otherwise you are likely to get an error.