Section: Exploring `lm` function in R

PLSC 503

2019-04-04

Regression Concepts: Review

Let us continue with a general recap of concepts from linear regression, first with two variables, then more.

Question for today: What is a linear regression? What do coefficients represent? Why is using statistical software convenient and in our case, what are we asking R to do?

Two variables

The CEF:

$$g(x) = \mathrm{E}[Y|X = x], \forall x \in \mathbb{R}, f_X(x) > 0.$$

The BLP:

$$g(x) = \alpha + \beta x.$$

Minimizing the loss function

$$\arg\min \mathrm{E}[U^2] = \mathrm{E}[(Y - a + bX)^2],$$

we get

$$b = \frac{\mathrm{Cov}(X, Y)}{\mathrm{V}[X]} a = \mathrm{E}[Y] - b\mathrm{E}[X]$$

More than two variables

Same thing for the multivariate BLP, just with more variables. If we have function $g(x_1, x_2, \ldots, x_k)$, we define a linear function that looks like:

$$g(x_1, x_2, \ldots, x_k) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_k X_k.$$

Function `lm`

When we use the function `lm`, we are commanding R to make these transformations, i.e. into the form of a linear function.

Load data

Let's look at newest American National Election Studies (ANES) pilot data. See this link for the codebook: ANES 2018 Pilot Study link. The data we will be using is a reduced version of the original data (since there are many variables and certain exceptions with missing values, etc. See Rmd for details on how I cleaned this, although I explained in section as well).

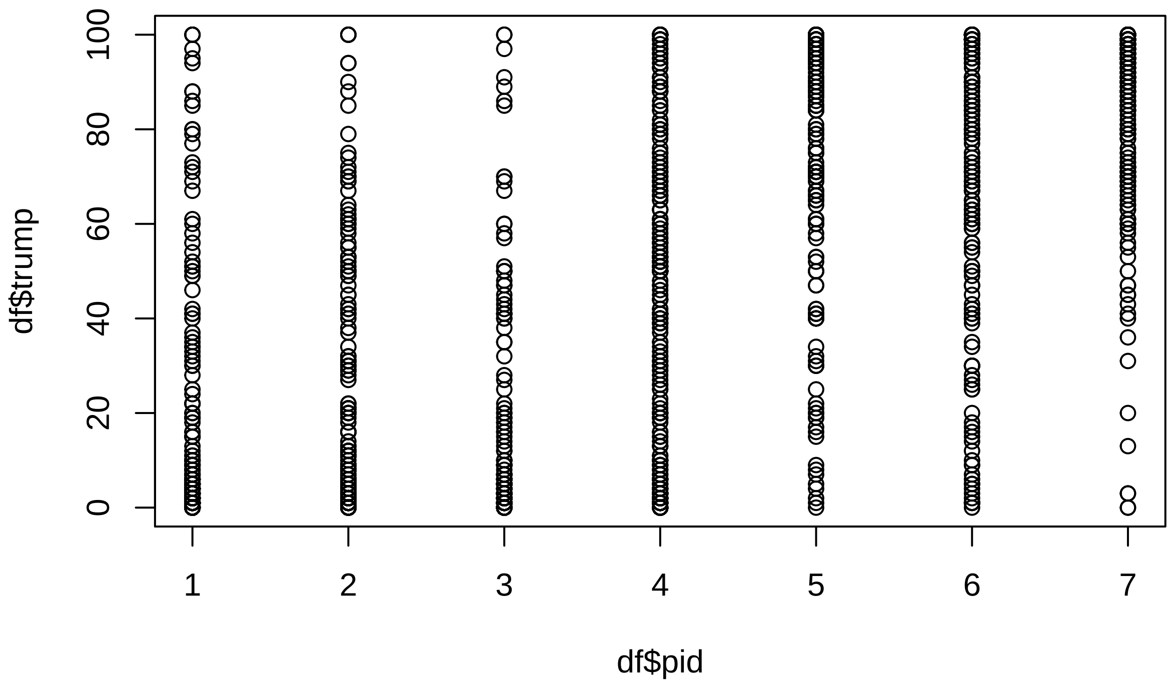
Load data

```r
df <- read.csv("./df_anes_503.csv")
```

Take a minute to look through the variables and read through the relevant entries in the codebook.

Simple plots

```r
plot(df$pid, df$trump)
```



What do we see?

Linear regression with `lm`

First, let's look at the function `lm`. What does it do? We can first look up documentation with `?lm`. Then, let's jump in by trying the following specification:

```r
lm(trump ~ pid, data = df)
```

```
##
## Call:
## lm(formula = trump ~ pid, data = df)
##
## Coefficients:
## (Intercept)          pid
##      -13.13        14.35
```

Question: What do these values represent? (Remember the opening discussion.)

```r
(fit <- lm(trump ~ pid, data = df))
```

```
##
## Call:
## lm(formula = trump ~ pid, data = df)
```

```
##
## Coefficients:
## (Intercept)          pid
##      -13.13        14.35
```

We can find out more about these estimated values: `summary` will give us a synopsis:

```r
summary(fit)
```

```
##
## Call:
## lm(formula = trump ~ pid, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -87.298 -15.560  -0.213  12.702  98.787
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -13.1348     1.0806  -12.15   <2e-16 ***
## pid          14.3476     0.2519   56.96   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 26.42 on 2377 degrees of freedom
##   (121 observations deleted due to missingness)
## Multiple R-squared:  0.5772, Adjusted R-squared:  0.577
## F-statistic:  3245 on 1 and 2377 DF,  p-value: < 2.2e-16
```

Classical SEs

We can see a shorter summary like this:

```r
summary(fit)$coef
```

```
##              Estimate Std. Error   t value    Pr(>|t|)
## (Intercept) -13.13476  1.0806185 -12.15486 5.05146e-33
## pid          14.34759  0.2518819  56.96158 0.00000e+00
```

What standard errors are these? (What assumptions underlie classical standard errors?)

Before we move on to robust standard errors, let's save the coefficients in `coef` and the classical standard errors in `classical_ses`.

```r
# We can extract coefficients with `fit$coefficients`, `fit$coef`, `coef(fit)`
(coefs <- fit$coef)
```

```
## (Intercept)         pid
##   -13.13476    14.34759
```

```r
(classical_ses <- summary(fit)$coef[, 2])
```

```
## (Intercept)         pid
##   1.0806185   0.2518819
```

Package `sandwich`

To find robust standard errors, we first find the heteroskedasticity consistent (HC) variance-covariance matrix using the `vcovHC()` function from the `sandwich` package. Then, we can compute the standard errors.

The default is `type = "HC3"`. Let's specify the type as HCO (i.e. `vcovHC(fit, type = "HC0")`) for now (hint: may appear on problem set).

```
library("sandwich")
```

```
## Warning: package 'sandwich' was built under R version 3.4.4
```

```
# Compare standard and HC0 standard errors:
sqrt(diag(vcovHC(fit, type = "const")))
```

```
## (Intercept)         pid
##   1.0806185   0.2518819
```

```
sqrt(diag(vcovHC(fit, type = "HC0")))
```

```
## (Intercept)         pid
##   0.8712220   0.1992632
```

```
(robust_ses <- sqrt(diag(vcovHC(fit, type = "HC0"))))
```

```
## (Intercept)         pid
##   0.8712220   0.1992632
```

So we get the following coefficients, classical SEs, and robust SEs:

```
coef
```

```
## function (object, ...)
## UseMethod("coef")
## <bytecode: 0x7fe43ae8cd88>
## <environment: namespace:stats>
```

```
classical_ses
```

```
## (Intercept)         pid
##   1.0806185   0.2518819
```

```
robust_ses
```

```
## (Intercept)         pid
##   0.8712220   0.1992632
```

Other models

Create your own models and let's discuss them. See board for layout of results in table form.

```
(model_1 <- lm(trump ~ pid + age, data = df))
```

```
##
## Call:
## lm(formula = trump ~ pid + age, data = df)
##
## Coefficients:
## (Intercept)          pid          age
##    -24.2519      14.2318       0.2304
```

```r
(model_2 <- lm(trump ~ pid*age, data = df))
```

```
##
## Call:
## lm(formula = trump ~ pid * age, data = df)
##
## Coefficients:
## (Intercept)          pid          age      pid:age
##    -5.76898      9.20934     -0.12782      0.09615
```

```r
# See difference between model_3_compare and model_3:
# (model_3_compare <- lm(trump ~ pid + (age^2), data = df))
(model_3 <- lm(trump ~ pid + I(age^2), data = df))
```

```
##
## Call:
## lm(formula = trump ~ pid + I(age^2), data = df)
##
## Coefficients:
## (Intercept)          pid     I(age^2)
##  -18.982353    14.230272     0.002245
```

```r
(model_4 <- lm(trump ~ male1*age, data = df))
```

```
##
## Call:
## lm(formula = trump ~ male1 * age, data = df)
##
## Coefficients:
## (Intercept)        male1          age    male1:age
##      9.3219       9.3877       0.8976      -0.3724
```