Lab #7 - Causal Regression I

Econ 224

September 18th, 2018

Introduction

We'll use the package `stargazer` to generate pretty tables of results like the ones you see in journal articles. Make sure to install this package before proceeding.

```
library(stargazer)
```

I chose to output my `.Rmd` file to a pdf using LaTeX, so I used the option `type = latex`. If you're using `html` you'll need to change this to `type = 'html'`. If you want to see a "preview" of the table within R studio without compiling, choose `type = 'text'`.

```
stargazer(mtcars, type = 'latex', title = 'Descriptive Statistics')
```

% Table created by stargazer v.5.2.2 by Marek Hlavac, Harvard University. E-mail: hlavac at fas.harvard.edu
% Date and time: Fri, Sep 07, 2018 - 01:43:44 PM

Table 1: Descriptive Statistics

Statistic	N	Mean	St. Dev.	Min	Pctl(25)	Pctl(75)	Max
mpg	32	20.091	6.027	10	15.4	22.8	34
cyl	32	6.188	1.786	4	4	8	8
disp	32	230.722	123.939	71	120.8	326	472
hp	32	146.688	68.563	52	96.5	180	335
drat	32	3.597	0.535	2.760	3.080	3.920	4.930
wt	32	3.217	0.978	1.513	2.581	3.610	5.424
qsec	32	17.849	1.787	14.500	16.892	18.900	22.900
vs	32	0.438	0.504	0	0	1	1
am	32	0.406	0.499	0	0	1	1
gear	32	3.688	0.738	3	3	4	5
carb	32	2.812	1.615	1	2	4	8

Robust Standard Errors

Your reading assignment from Chapter 3 of ISL briefly discussed two ways that the standard regression inference formulas built into R can go wrong: (1) non-constant error variance, and (2) correlation between regression errors. Today we'll briefly look at the first of these problems and how to correct for it.

Consider the simple linear regression $y_i = \beta_0 + \beta_1 x_i + \epsilon_i$. If the variance of ϵ_i is unrelated to the value of the predictor x_i, we say that the regression errors are *homoskedastic*. This is just a fancy Greek work for *constant variance*. If instead, the variance of ϵ_i depends on the value of x_i, we say that the regression errors are *heteroskedastic*. This is just a fancy Greek word for *non-constant variance*. Heteroskedasticity does not invalidate our least squares estimates of β_0 and β_1, but it does invalidate the formulas used by `lm` to calculate standard errors and p-values.

Let's look at a simple simulation example:

```
set.seed(4321)
n <- 100
x <- runif(n)
e1 <- rnorm(n, mean = 0, sd = sqrt(2 * x))
e2 <- rnorm(n, mean = 0, sd = 1)
intercept <- 0.2
slope <- 0.9
y1 <- intercept + slope * x + e1
y2 <- intercept + slope * x + e2
library(tidyverse)
mydat <- tibble(x, y1, y2)
rm(x, y1, y2)
```

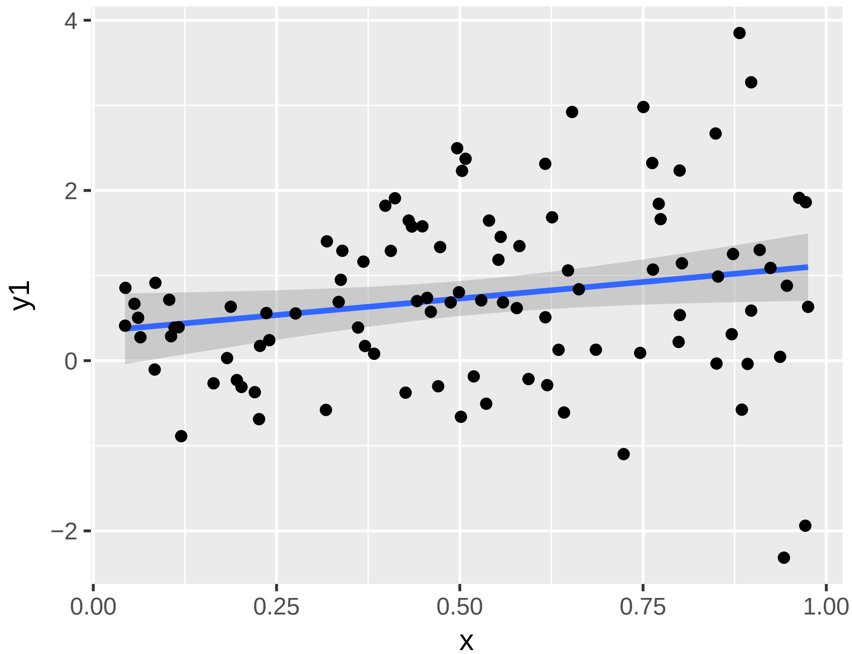
Exercise #1

1. Read through my simulation code and make sure you understand what each step is going. What is the distribution of the errors? What is the distribution of `x`? In the simulation design, is there a relationship between `x` and `y1`? What about `y2`?
2. For each of the two simulated outcome variables `y1` and `y2`, plot the outcome against `x` along with the linear regression line.
3. Based on your plots from part 2 and the simulation code, which errors are heteroskedastic: `e1`, `e2`, both, or neither? How can you tell?

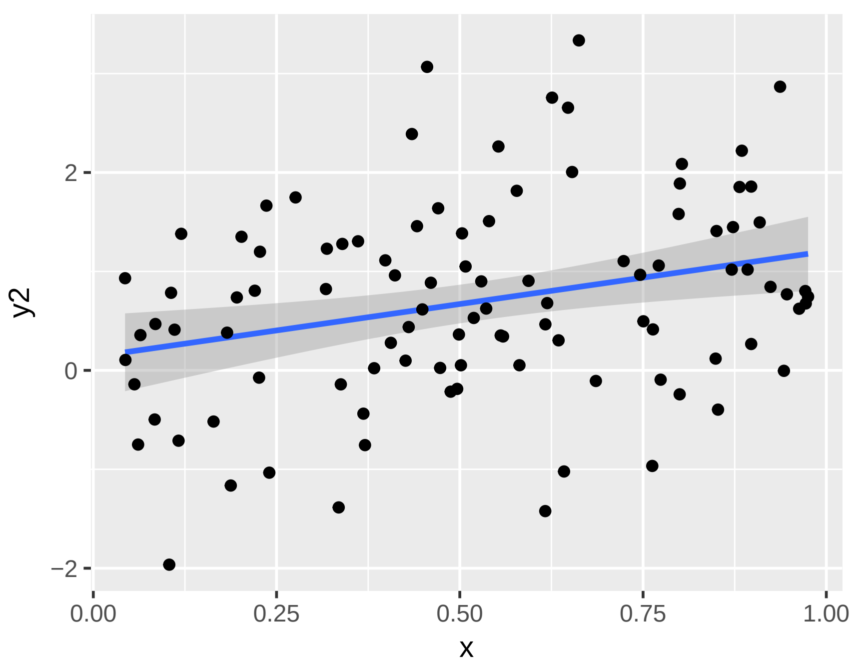
Solution to Exercise #1

1. `x` is uniform and the errors are normally distributed. There is indeed a relationship between `x` and `y`: the conditional mean of `y1` given `x` is `0.2 + 0.4 x` and the same is true of `y2`
2. Here is a simple way to make the plots:

```
library(ggplot2)
ggplot(mydat, aes(x, y1)) +
  geom_smooth(method = 'lm') +
  geom_point()
```




```r
ggplot(mydat, aes(x, y2)) +
  geom_smooth(method = 'lm') +
  geom_point()
```



3. The errors `e1` are heteroskedastic while the errors `e2` are homoskedastic. We can see this both from plotting the data which "fan out" around the regression line for `y1` and from the simulation code: to generate `e1` we multiplied some normal random draws by the value of `x` so the variance clearly depends on `x`

Robust Standard Errors using `lm_robust`

Install the package `estimatr`. Provides a replacement for `lm` called `lm_robust` that allows us to choose robust standard errors

```
library(estimatr)
reg1_classical <- lm_robust(y1 ~ x, mydat, se_type = 'stata')
summary(reg1_classical)
```

```
Call:
lm_robust(formula = y1 ~ x, data = mydat, se_type = "stata")

Standard error type:  HC1

Coefficients:
            Estimate Std. Error t value Pr(>|t|)  CI Lower CI Upper DF
(Intercept)   0.3418     0.1739   1.966  0.05215 -0.003241   0.6868 98
x             0.7766     0.4068   1.909  0.05919 -0.030707   1.5839 98

Multiple R-squared:  0.04119 ,  Adjusted R-squared:  0.0314
F-statistic: 3.644 on 1 and 98 DF,  p-value: 0.05919
```

```
reg1_robust <- lm_robust(y1 ~ x, mydat, se_type = 'classical')
summary(reg1_robust)
```

```
Call:
lm_robust(formula = y1 ~ x, data = mydat, se_type = "classical")

Standard error type:  classical

Coefficients:
            Estimate Std. Error t value Pr(>|t|) CI Lower CI Upper DF
(Intercept)   0.3418     0.2240   1.526  0.13027 -0.10273   0.7863 98
x             0.7766     0.3785   2.052  0.04286  0.02548   1.5277 98

Multiple R-squared:  0.04119 ,  Adjusted R-squared:  0.0314
F-statistic:  4.21 on 1 and 98 DF,  p-value: 0.04286
```

The nice thing about using `lm_robust` is that it plays nicely with `linearHypothesis` for carrying out F-tests. In an example with only one regressor the F-test is completely superfluous (the F-test statistic is simply the square of the t-test statistic for the slope!) but just to see that it works:

```
library(car)
summary(lm(y1 ~ x, mydat))$fstatistic
```

```
    value     numdf     dendf
 4.209829  1.000000 98.000000
```

```r
linearHypothesis(reg1_classical, 'x = 0')
```

```
Linear hypothesis test

Hypothesis:
x = 0

Model 1: restricted model
Model 2: y1 ~ x

  Res.Df Df  Chisq Pr(>Chisq)
1     99
2     98  1 3.6442    0.05626 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
linearHypothesis(reg1_robust, 'x = 0')
```

```
Linear hypothesis test

Hypothesis:
x = 0

Model 1: restricted model
Model 2: y1 ~ x

  Res.Df Df  Chisq Pr(>Chisq)
1     99
2     98  1 4.2098    0.04019 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Exercise #2

Repeat my inference comparison from above for the regression `y2 ~ x` using classical and robust standard errors. Explain your results.

Solution to Exercise #2

```r
reg2_classical <- lm_robust(y2 ~ x, mydat, se_type = 'stata')
summary(reg1_classical)
```

```
Call:
lm_robust(formula = y1 ~ x, data = mydat, se_type = "stata")

Standard error type:  HC1
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)  CI Lower CI Upper DF
(Intercept)   0.3418     0.1739   1.966  0.05215 -0.003241   0.6868 98
x             0.7766     0.4068   1.909  0.05919 -0.030707   1.5839 98

Multiple R-squared:  0.04119 ,  Adjusted R-squared:  0.0314
F-statistic: 3.644 on 1 and 98 DF,  p-value: 0.05919
```

```r
reg2_robust <- lm_robust(y2 ~ x, mydat, se_type = 'classical')
summary(reg1_robust)
```

```
Call:
lm_robust(formula = y1 ~ x, data = mydat, se_type = "classical")

Standard error type:  classical

Coefficients:
            Estimate Std. Error t value Pr(>|t|) CI Lower CI Upper DF
(Intercept)   0.3418     0.2240   1.526  0.13027 -0.10273   0.7863 98
x             0.7766     0.3785   2.052  0.04286  0.02548   1.5277 98

Multiple R-squared:  0.04119 ,  Adjusted R-squared:  0.0314
F-statistic:  4.21 on 1 and 98 DF,  p-value: 0.04286
```

Angrist and Lavy (1999)

https://economics.mit.edu/faculty/angrist/data1/data/anglavy99