

# Standard Error Estimation

AECN 396/896-002

## Before we start

### Learning objectives

Understand the consequences of the violation of the homoskedasticity assumption and how to deal with the problem

### Table of contents

1. Review on statistical hypothesis testing
2. Testing (linear model)
3. Confidence interval

## Heteroskedasticity

---

## Homoskedasticity and Heteroskedasticity

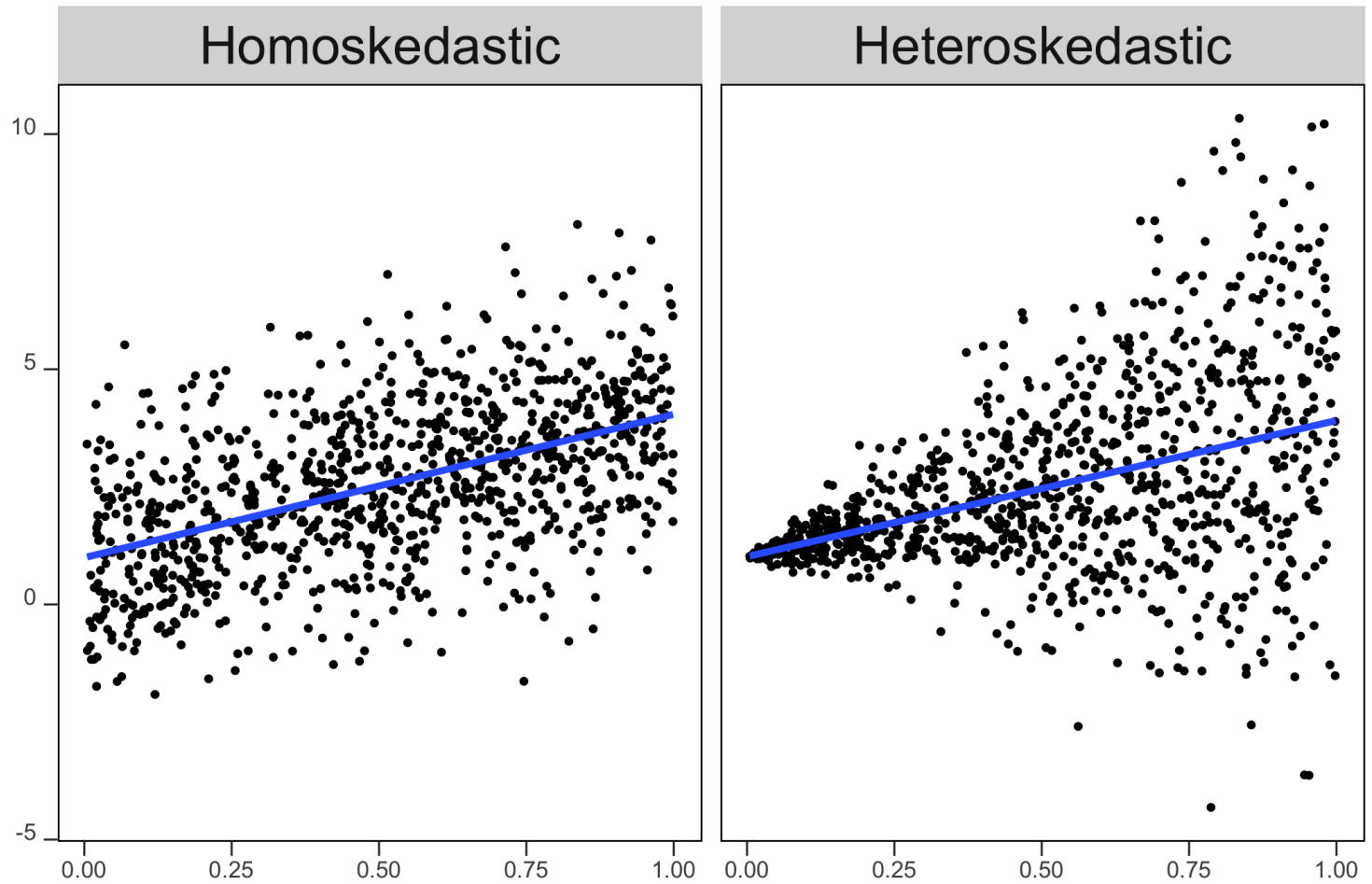
### Homoskedasticity

$$\text{Var}(u|x) = \sigma^2$$

### Heteroskedasticity

$$\text{Var}(u|x) = f(x)$$

## Visualization



## Central Questions

What are the consequences of assuming the error is homoskedastic when it is heteroskedastic in reality?

- Estimation of coefficients ( $\hat{\beta}_j$ )?
- Estimation of the variance of  $\hat{\beta}_j$ ?

## Coefficient estimators

### Question

## Coefficient estimators

### Question

Are OLS estimators unbiased when error is heteroskedastic?



## Coefficient estimators

### Question

Are OLS estimators unbiased when error is heteroskedastic?

### Answer

## Coefficient estimators

### Question

Are OLS estimators unbiased when error is heteroskedastic?

### Answer

Yes

## Coefficient estimators

### Question

Are OLS estimators unbiased when error is heteroskedastic?

### Answer

Yes

### Why?

## Coefficient estimators

### Question

Are OLS estimators unbiased when error is heteroskedastic?

### Answer

Yes

### Why?

We do not use the homoskedasticity assumption to prove that the OLS estimator is unbiased.

## Variance of the coefficient estimators

We learned that when the homoskedasticity assumption holds, then,

$$Var(\hat{\beta}_j) = \frac{\sigma^2}{SST_x(1-R_j^2)}$$

## Variance of the coefficient estimators

We learned that when the homoskedasticity assumption holds, then,

$$Var(\hat{\beta}_j) = \frac{\sigma^2}{SST_x(1-R_j^2)}$$

We used the following as the estimator of  $Var(\hat{\beta}_j)$

$$\frac{\hat{\sigma}^2}{SST_x(1-R_j^2)} \text{ where } \hat{\sigma}^2 = \frac{\sum_{i=1}^N \hat{u}_i^2}{N-k-1}$$

## Variance of the coefficient estimators

We learned that when the homoskedasticity assumption holds, then,

$$Var(\hat{\beta}_j) = \frac{\sigma^2}{SST_x(1-R_j^2)}$$

We used the following as the estimator of  $Var(\hat{\beta}_j)$

$$\frac{\hat{\sigma}^2}{SST_x(1-R_j^2)} \text{ where } \hat{\sigma}^2 = \frac{\sum_{i=1}^N \hat{u}_i^2}{N-k-1}$$

**Important** : By default, R and other statistical software uses this formula to get estimates of the variance of  $\hat{\beta}_j$ .

**Note** : Remember, we let  $\widehat{Var}(\hat{\beta}_j)$  denote the estimator of the variance of  $\hat{\beta}_j$ .

## Variance of the coefficient estimators

But, under heteroskedasticity,

$$Var(\hat{\beta}_j) \neq \frac{\sigma^2}{SST_x(1-R_j^2)}$$



## Variance of the coefficient estimators

But, under heteroskedasticity,

$$\text{Var}(\hat{\beta}_j) \neq \frac{\sigma^2}{SST_x(1-R_j^2)}$$

**Question :**

Is  $E[\widehat{\text{Var}(\hat{\beta}_j)}] = E\left[\frac{\hat{\sigma}^2}{SST_x(1-R_j^2)}\right] = \text{Var}(\hat{\beta}_j)$  under heteroskedasticity?

## Variance of the coefficient estimators

But, under heteroskedasticity,

$$\text{Var}(\hat{\beta}_j) \neq \frac{\sigma^2}{SST_x(1-R_j^2)}$$

**Question :**

Is  $E[\widehat{\text{Var}(\hat{\beta}_j)}] = E\left[\frac{\hat{\sigma}^2}{SST_x(1-R_j^2)}\right] = \text{Var}(\hat{\beta}_j)$  under heteroskedasticity?

**Answer :** No

## Variance of the coefficient estimators

### Question :

So, what are the consequences of using  $\widehat{Var}(\hat{\beta}_j) = \frac{\hat{\sigma}^2}{SST_x(1-R_j^2)}$  under heteroskedasticity?

### Consequence

Your hypothesis testing is going to be biased.

What does it mean to have hypothesis testing biased? Roughly speaking, it means that you over-reject/under-reject the hypothesis than you intend to.

## Consequence of heteroskedasticity on testing

Let's run MC simulations to see the consequence of ignoring heteroskedasticity.

## Consequence of heteroskedasticity on testing

Let's run MC simulations to see the consequence of ignoring heteroskedasticity.

### Model

$$y = 1 + \beta x + u, \text{ where } \beta = 0$$

## Consequence of heteroskedasticity on testing

Let's run MC simulations to see the consequence of ignoring heteroskedasticity.

### Model

$$y = 1 + \beta x + u, \text{ where } \beta = 0$$

### Test of interest

- $H_0 : \beta = 0$
- $H_1 : \beta \neq 0$

## Consequence of heteroskedasticity on testing

Let's run MC simulations to see the consequence of ignoring heteroskedasticity.

### Model

$$y = 1 + \beta x + u, \text{ where } \beta = 0$$

### Test of interest

- $H_0 : \beta = 0$
- $H_1 : \beta \neq 0$

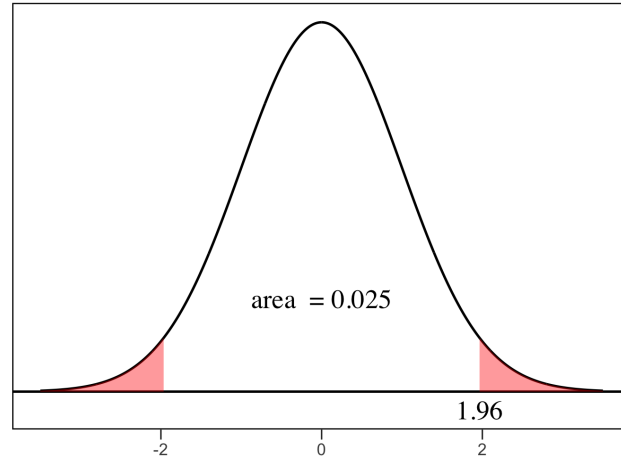
### Question

If you test the null hypothesis at the 5% significance level, what should be the probability that you reject the null hypothesis when it is actually true?

$$Pr(\text{reject } H_0 | H_0 \text{ is true}) = ?$$

### Question

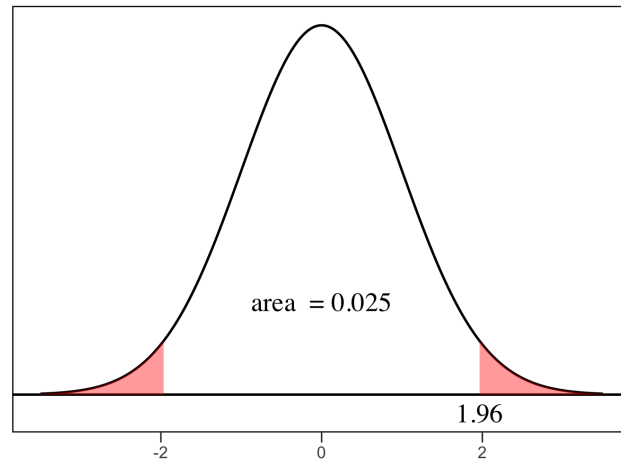
If you test the null hypothesis at the 5% significance level, what should be the probability that you reject the null hypothesis when it is actually true?





### Question

If you test the null hypothesis at the 5% significance level, what should be the probability that you reject the null hypothesis when it is actually true?



### Answer

Since the null is true (we generate the data that way!), the probability you reject the null should be the same as the significance level, which is 5.

## MC simulation: conceptual steps

- generate a dataset so that  $\beta_1$  (the coefficient on  $x$ ) is zero

$$y = \beta_0 + \beta_1 x + v$$

- estimate the model and find  $\hat{\beta}_1$  and  $\widehat{se}(\hat{\beta}_1)$
- calculate  $t$ -statistic  $(\hat{\beta}_x - 0 / \widehat{se}(\hat{\beta}_x))$  and decide whether you reject the null or not
- repeat the above 1000 times
- check how often you reject the null (should be close to 50 times)

## MC simulation: R code

```
set.seed(927834)

N <- 1000 # number of observations
B <- 1000 # number of simulations

b_hat_store <- rep(0, B) # beta hat storage
t_stat_store <- rep(0, B) # t-stat storage
c_value <- qt(0.975, N - 2) # critical value

x <- runif(N, 0, 1) # x (fixed across iterations)

for (i in 1:B){
  #--- generate data ---#
  het_u <- 3 * rnorm(N, mean = 0, sd = 2 * x) # heteroskedastic error
  y <- 1 + het_u # y
  data_temp <- data.frame(y = y, x = x)

  #--- regression ---#
  ols_res <- lm(y ~ x, data = data_temp)

  b_hat <- ols_res$coef['x'] # coef estimate on x
  b_hat_store[i] <- b_hat # save the coef estimate
  vcov_ols <- vcov(ols_res) # get variance covariance matrix
  t_stat_store[i] <- b_hat / sqrt(vcov_ols['x', 'x']) # calculate t-stat
}
```

## MC simulation: Results

```
### how many times do you reject? ###  
reject_or_not <- abs(t_stat_store) > c_value  
mean(reject_or_not)
```

```
## [1] 0.108
```

## MC simulation: Results

```
#--- how many times do you reject? ---#  
reject_or_not <- abs(t_stat_store) > c_value  
mean(reject_or_not)
```

```
## [1] 0.108
```

### Consequence of ignoring heteroskedasticity

We rejected the null hypothesis 10.8% of the time, instead of 5%.

## MC simulation: Results

```
#--- how many times do you reject? ---#  
reject_or_not <- abs(t_stat_store) > c_value  
mean(reject_or_not)
```

```
## [1] 0.108
```

### Consequence of ignoring heteroskedasticity

We rejected the null hypothesis 10.8% of the time, instead of 5%.

- So, you are more likely to claim that  $x$  has a statistically significant impact than you are supposed to.

## MC simulation: Results

```
### how many times do you reject? ###  
reject_or_not <- abs(t_stat_store) > c_value  
mean(reject_or_not)
```

```
## [1] 0.108
```

### Consequence of ignoring heteroskedasticity

We rejected the null hypothesis 10.8% of the time, instead of 5%.

- So, you are more likely to claim that  $x$  has a statistically significant impact than you are supposed to.
- The use of the formula  $\frac{\hat{\sigma}^2}{SST_x(1-R_j^2)}$  seemed to (over/under)-estimate the true variance of the OLS estimators?

## MC simulation: Results

```
### how many times do you reject? ###  
reject_or_not <- abs(t_stat_store) > c_value  
mean(reject_or_not)
```

```
## [1] 0.108
```

### Consequence of ignoring heteroskedasticity

We rejected the null hypothesis 10.8% of the time, instead of 5%.

- So, you are more likely to claim that  $x$  has a statistically significant impact than you are supposed to.
- The use of the formula  $\frac{\hat{\sigma}^2}{SST_x(1-R_j^2)}$  seemed to (over/under)-estimate the true variance of the OLS estimators?
- In general, the direction of bias is ambiguous.



## How should we address this problem?

- Now, we understand the consequence of heteroskedasticity:

$\frac{\hat{\sigma}^2}{SST_x(1-R_j^2)}$  is a biased estimator of  $Var(\hat{\beta})$ , which makes any kind of testings based on it invalid.

## How should we address this problem?

- Now, we understand the consequence of heteroskedasticity:

$\frac{\hat{\sigma}^2}{SST_x(1-R_j^2)}$  is a biased estimator of  $Var(\hat{\beta})$ , which makes any kind of testings based on it invalid.

- Can we credibly estimate the variance of the OLS estimators?

## How should we address this problem?

- Now, we understand the consequence of heteroskedasticity:

$\frac{\hat{\sigma}^2}{SST_x(1-R_j^2)}$  is a biased estimator of  $Var(\hat{\beta})$ , which makes any kind of testings based on it invalid.

- Can we credibly estimate the variance of the OLS estimators?

**White-Huber-Eicker heteroskedasticity-robust standard error estimator**

## How should we address this problem?

- Now, we understand the consequence of heteroskedasticity:

$\frac{\hat{\sigma}^2}{SST_x(1-R_j^2)}$  is a biased estimator of  $Var(\hat{\beta})$ , which makes any kind of testings based on it invalid.

- Can we credibly estimate the variance of the OLS estimators?

### White-Huber-Eicker heteroskedasticity-robust standard error estimator

- valid in the presence of heteroskedasticity of **unknown form**
- heteroskedasticity-robust standard error estimator in short

### Heteroskedasticity-robust standard error estimator

$$\widehat{Var}(\hat{\beta}_j) = \frac{\sum_{i=1}^n \hat{r}_{i,j}^2 \hat{u}_i^2}{SSR_j^2}$$

- $\hat{u}_i$ : residual from regressing  $y$  on all the independent variables
- $\hat{r}_{i,j}$ : residual from regressing  $x_j$  on all other independent variables for  $i$ th observation
- $SSR_j^2$ : the sum of squared residuals from regressing  $x_j$  on all other independent variables

We spend **NO** time to try to understand what's going on with the estimator.

What you need is

- understand the consequence of heteroskedasticity
- know there is an estimator that is appropriate under heteroskedasticity, meaning that it will give you the correct estimate of the variance of the OLS estimator
- know how to use the heteroskedasticity-robust standard error estimator in practice using  $R$  (or some other software)

## In practice

Here is the well-accepted procedure in econometric analysis:

## In practice

Here is the well-accepted procedure in econometric analysis:

- Estimate the model using OLS (you do nothing special here)



## In practice

Here is the well-accepted procedure in econometric analysis:

- Estimate the model using OLS (you do nothing special here)
- Assume the error term is heteroskedastic and estimate the variance of the OLS estimators
  - There are tests to whether error is heteroskedastic or not: Breusch-Pagan test and White test
  - In practice, almost nobody bothers to conduct these tests
  - We do not learn how to run these tests

## In practice

Here is the well-accepted procedure in econometric analysis:

- Estimate the model using OLS (you do nothing special here)
- Assume the error term is heteroskedastic and estimate the variance of the OLS estimators
  - There are tests to whether error is heteroskedastic or not: Breusch-Pagan test and White test
  - In practice, almost nobody bothers to conduct these tests
  - We do not learn how to run these tests
- Replace the estimates from  $\widehat{Var}(\hat{\beta})_{default}$  with those from  $\widehat{Var}(\hat{\beta})_{robust}$  for testing

## In practice

Here is the well-accepted procedure in econometric analysis:

- Estimate the model using OLS (you do nothing special here)
- Assume the error term is heteroskedastic and estimate the variance of the OLS estimators
  - There are tests to whether error is heteroskedastic or not: **Breusch-Pagan** test and **White** test
  - In practice, almost nobody bothers to conduct these tests
  - We do not learn how to run these tests
- Replace the estimates from  $\widehat{Var}(\hat{\beta})_{default}$  with those from  $\widehat{Var}(\hat{\beta})_{robust}$  for testing
- But, we do not replace coefficient estimates (remember, coefficient estimation is still unbiased under heteroskedasticity)

## Implementation in R

We use

- the `fixest::se()` function from the `fixest` package to estimate heteroskedasticity-robust standard errors
- the `summary()` function do tests of  $\beta_j = 0$

## Implementation in R

We use

- the `fixest::se()` function from the `fixest` package to estimate heteroskedasticity-robust standard errors
- the `summary()` function do tests of  $\beta_j = 0$

Let's run a regression using `MLB1.dta`.

```
#--- library ---#
library(readstata13)

#--- import the data ---#
mlb_data <- read.dta13('MLB1.dta')

#--- regression ---#
reg_mlb <- feols(log(salary) ~ years + bavg, data = mlb_data)
```

# Obtaining Heteroskedasticity-robust SE estimates

## General Syntax

Here is the general syntax to obtain various types of VCOV (and se) estimates:

```
## vcov  
vcov(regression result, vcov = "type of vcov")  
  
## only the standard errors  
se(regression result, vcov = "type of vcov")
```

## heteroskedasticity-robust standard error estimation

Specifically for White-Huber heteroskedasticity-robust VCOV and se estimates,

```
## vcov  
vcov(reg_mlb, vcov = "hetero")
```

```
##           (Intercept)          years          bavg  
## (Intercept) 0.495103882 0.0058059916 -2.080065e-03  
## years      0.005805992 0.0003117152 -2.976110e-05  
## bavg       -0.002080065 -0.0000297611 8.892009e-06
```

```
## only the standard errors  
se(reg_mlb, vcov = "hetero")
```

```
## (Intercept)      years      bavg  
## 0.703636186 0.017655458 0.002981947
```

## Compare with the Default

### Default

```
(  
  se_hom <- se(reg_mlb)  
)
```

```
## (Intercept)      years      bavg  
## 0.343071420 0.013222511 0.001335294
```

### Heteroskedasticity-robust

```
## (Intercept)      years      bavg  
## 0.703636186 0.017655458 0.002981947
```

## Updating the test of coefficients being zero

### Default

```
tidy(reg_mlb)
```

```
## # A tibble: 3 × 5
##   term      estimate std.error statistic  p.value
##   <chr>      <dbl>    <dbl>    <dbl>    <dbl>
## 1 (Intercept) 11.0      0.343     32.2 1.25e-106
## 2 years       0.166     0.0132    12.6 3.16e- 30
## 3 bavg        0.00539    0.00134     4.04 6.59e- 5
```

### Heteroskedasticity-robust

```
tidy(reg_mlb, vcov = "hetero")
```

```
## # A tibble: 3 × 5
##   term      estimate std.error statistic  p.value
##   <chr>      <dbl>    <dbl>    <dbl>    <dbl>
## 1 (Intercept) 11.0      0.704     15.7 2.13e-42
## 2 years       0.166     0.0177     9.43 5.96e-19
## 3 bavg        0.00539    0.00298     1.81 7.13e- 2
```



In presenting the regression results in a nicely formatted table, we used `modelsummary::msummary()`.

We can easily swap the default `se` with the heteroskedasticity-robust `se` using the `statistic_override` option in `msummary()`.

```
vcov_het <- vcov(reg_mlb, vcov = "hetero")
vcov_homo <- vcov(reg_mlb)

modelsummary::msummary(
  list(reg_mlb, reg_mlb),
  statistic_override = list(vcov_het, vcov_homo),
  # keep these options as they are
  stars = TRUE,
  gof_omit = "IC|Log|Adj|F|Pseudo|Within"
)
```

	Model 1	Model 2
(Intercept)	11.042***	11.042***
	(0.704)	(0.343)
bavg	0.005***	0.005***
	(0.003)	(0.001)
years	0.166***	0.166***
	(0.018)	(0.013)
Num.Obs.	353	353
R2	0.367	0.367
Std. errors	IID	IID
* p < 0.1, ** p < 0.05, *** p < 0.01		

```
reg_mlb <- feols(log(salary) ~ years + bavg, vcov = "hetero", data = mlb_data)
tidy(reg_mlb)
```

```
## # A tibble: 3 × 5
##   term      estimate std.error statistic  p.value
##   <chr>      <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept) 11.0       0.704      15.7 2.13e-42
## 2 years      0.166      0.0177     9.43 5.96e-19
## 3 bavg       0.00539    0.00298     1.81 7.13e- 2
```

```
modelsummary::msummary(
  list(reg_mlb),
  # keep these options as they are
  stars = TRUE,
  gof_omit = "IC|Log|Adj|F|Pseudo|Within"
)
```

Model 1	
(Intercept)	11.042***
	(0.704)
bavg	0.005*
	(0.003)
years	0.166***
	(0.018)
Num.Obs.	353
R2	0.367
Std. errors	Heteroskedasticity-robust
* p < 0.1, ** p < 0.05, *** p < 0.01	

## Het-robust SE estimator: validation

Does the heteroskedasticity-robust se estimator really work? Let's see using MC simulations:

```
set.seed(478954)

#--- x fixed across iterations ---#
x <- runif(N,0,1) # x

for (i in 1:B){
  #--- generate data ---#
  het_u <- 3 * rnorm(N, mean = 0, sd = 2 * x) # heteroskedastic error
  y <- 1 + het_u # y
  data_temp <- data.frame(y = y, x = x)

  #--- regression ---#
  ols_res <- feols(y ~ x,data = data_temp)
  b_hat <- ols_res$coefficient['x'] # coef estimate on x
  b_hat_store[i] <- b_hat # save the coef estimate
  se_het <- se(ols_res, vcov = "hetero")["x"] # get variance covariance matrix
  t_stat_store[i] <- b_hat/se_het # calculate t-stat
}
```

## MC simulation results

```
reject_or_not <- abs(t_stat_store) > c_value  
mean(reject_or_not)
```

```
## [1] 0.053
```

Okay, not perfect. But, certainly better.

## Clustered Error

---

### Clustered Error

- Often times, observations can be grouped into clusters
- Errors within the cluster can be correlated

### Example 1

College GPA: cluster by college

$$GPA_{col} = \beta_0 + \beta_1 income + \beta_2 GPA_{hs} + u$$

- Your observations consist of students' GPA scores across many colleges
- Because of some unobserved (omitted) school characteristics, error terms for the individuals in the same college might be correlated.
  - grading policy

### Example 2

Education Impacts on Income: cluster by individual

- Your observations consist of 500 individuals with each individual tracked over 10 years
- Because of some unobserved (omitted) individual characteristics, error terms for time-series observations within an individual might be correlated.
  - innate ability



## Consequences of clustered error

### Question

Are the OLS estimators of the coefficients biased in the presence of clustered error?

## Consequences of clustered error

### Question

Are the OLS estimators of the coefficients biased in the presence of clustered error?

### Answer

## Consequences of clustered error

### Question

Are the OLS estimators of the coefficients biased in the presence of clustered error?

### Answer

No, the correlation between  $x$  and  $u$  would hurt you, but not correlation among  $u$ .

## Consequences of clustered error

### Question

Are  $\widehat{Var}(\hat{\beta})_{default}$  unbiased estimators of  $Var(\hat{\beta})$ ?

## Consequences of clustered error

### Question

Are  $\widehat{Var}(\hat{\beta})_{default}$  unbiased estimators of  $Var(\hat{\beta})$ ?

### Answer

## Consequences of clustered error

### Question

Are  $\widehat{Var}(\hat{\beta})_{default}$  unbiased estimators of  $Var(\hat{\beta})$ ?

### Answer

No,  $\widehat{Var}(\hat{\beta})_{default}$  is unbiased only under homoskedasticity assumption, which assumes no correlation between errors.

## Consequences of clustered error

### Question

Which has more information?

- two errors that are independent
- two errors that are correlated

### Consequences

- If you were to use  $\widehat{Var}(\hat{\beta})_{default}$  to estimate  $Var(\hat{\beta})$  in the presence of clustered error, you would (under/over)-estimate the true  $Var(\hat{\beta})$ .
- This would lead to rejecting null hypothesis (more/less) often than you are supposed to.

## MC simulations: conceptual steps

Here are the conceptual steps of the MC simulations to see the consequence of clustered error.

- generate data according to the generating process in which the error terms ( $u$ ) within the cluster (two clusters in this example) is correlated and  $\beta_1$  is set to 0 in the model below:

$$y = \beta_0 + \beta_1 x + u$$

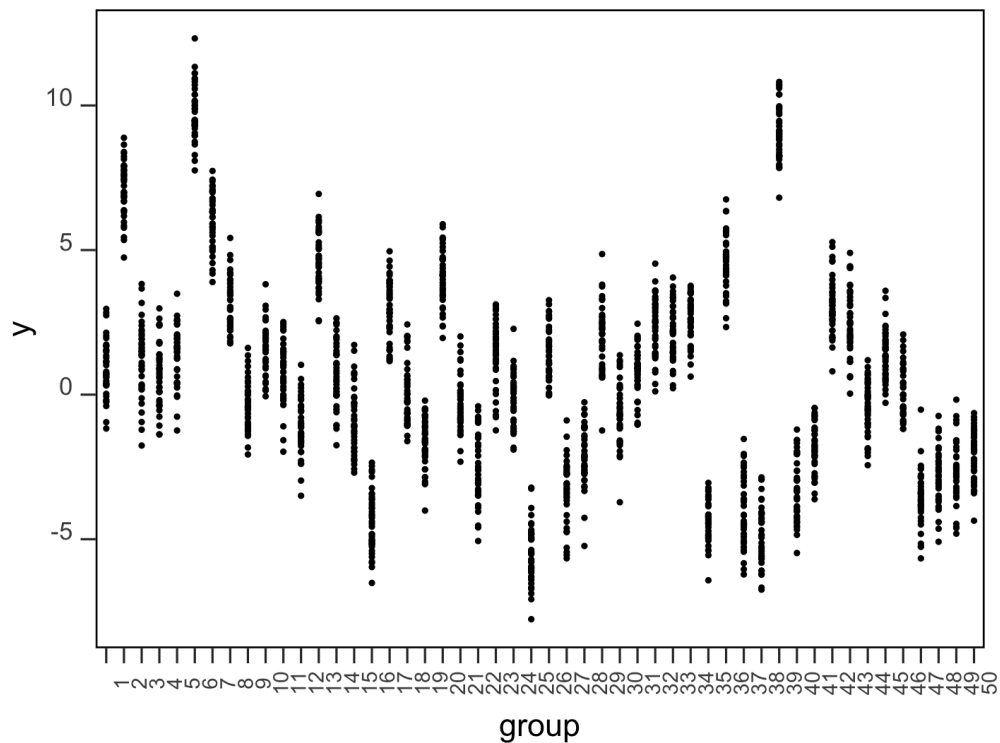
- estimate the model and find  $\hat{\beta}_x$  and  $\widehat{se}(\hat{\beta}_x)$
- calculate  $t$ -statistic ( $\hat{\beta}_x / \widehat{se}(\hat{\beta}_x)$ ) for the (correct) null hypothesis of  $\beta_1 = 0$
- repeat steps 1-3 for 1000 times
- see how many times out of 1000 times you reject the null hypothesis:  $H_0 : \beta_x = 0$



## R code: Data Genrating Process

```
#--- setup ---#  
library(MASS) # to use the mvrnorm() function later  
  
N <- 2000 # total number of observations  
G <- 50 # number of groups  
Ng <- N/G # number of observations per group  
  
#--- error correlated within group ---#  
u <-  
  mvrnorm(  
    G, mu = rep(0, Ng),  
    Sigma = matrix(10, nrow = Ng, ncol = Ng) + diag(Ng)  
  ) %>% t() %>% c()  
  
#--- x correlated within group ---#  
x <-  
  mvrnorm(  
    G, mu = rep(0, Ng),  
    Sigma = matrix(1, nrow = Ng, ncol = Ng) + diag(Ng) * .2  
  ) %>% t() %>% c()  
  
#--- other variables ---#  
y <- 1 + 0 * x + u  
  
#--- data.frame ---#  
data <- data.frame(y = y, x = x, group = rep(1:G, each = Ng))
```

## Visualization of the clustered nature of the data



## R code: MC simualtion

```
set.seed(58934)
B <- 1000
t_stat_store <- rep(0,B)
N <- 2000 # total number of observations
G <- 50 # number of groups
Ng <- N/G # number of observations per group

for (i in 1:B){
  #--- error correlated within group ---#
  u <-
  mvrnorm(
    G, mu = rep(0, Ng),
    Sigma = matrix(10, nrow = Ng, ncol = Ng) + diag(Ng)
  ) %>% t() %>% c()

  #--- x correlated within group ---#
  x <-
  mvrnorm(
    G, mu = rep(0, Ng),
    Sigma = matrix(1, nrow = Ng, ncol = Ng) + diag(Ng) * .2
  ) %>% t() %>% c()

  #--- other variables ---#
  y <- 1 + 0 * x + u

  #--- data.frame ---#
  data <- data.frame(y = y, x = x, group = rep(1:G, each = Ng))

  #--- OLS ---#
  reg <- feols(y ~ x, data = data)

  #--- get vcov ---#
  se_default <- se(reg)["x"]

  #--- calculate t-stat ---#
  t_stat <- reg$coefficient['x']/se_default
  t_stat_store[i] <- t_stat
}
```

## MC simulations: results

```
c_value <- qt(0.975, N - 2)

#--- how often do you reject the null ---#
mean(abs(t_stat_store) > c_value)
```

```
## [1] 0.748
```

## MC simulations: results

```
c_value <- qt(0.975, N - 2)

#--- how often do you reject the null ---#
mean(abs(t_stat_store) > c_value)
```

```
## [1] 0.748
```

### Important :

- clustered error can severely bias your test results
- it tends to make the impact of explanatory variables more significant than they truly are because the default estimator of the variance of the OLS estimator tends to greatly **under**-estimate the true variance of the OLS estimator.

## What to do?

### Cluster-robust standard error estimation

There exist estimators of  $Var(\hat{\beta})$  that take into account the possibility that errors are clustered.

- We call such estimators **cluster-robust variance covariance estimator** denoted as  $\widehat{Var(\hat{\beta})}_{cl}$
- We call estimates from such estimators **cluster-robust variance**

## Cluster-robust standard error estimation

I neither derive nor show the mathematical expressions of these estimators.

### This is what you need to do

- understand the consequence of clustered errors
- know there are estimators that are appropriate under clustered error
- know that the estimators we will learn take care of heteroskedasticity at the same time (so, they really are cluster- and heteroskedasticity-robust standard error estimators)
- know how to use the estimators in  $R$  (or some other software)

## R implementation

### Cluster-robust standard error

Similar with the `vcov` option for White-Huber heteroskedasticity-robust se, we can use the `cluster` option to get cluster-robust se.



## Before an R demonstration

Let's take a look at the MLB data again.

```
mlb_data[, c("salary", "years", "bavg", "nl")] %>% head()
```

```
##      salary years bavg nl
## 1 6329213     12  289  1
## 2 3375000      8  259  1
## 3 3100000      5  299  1
## 4 2900000      8  245  1
## 5 1650000     12  258  1
## 6  700000     17  286  1
```

`nl` (1 if in the National league, 0 if in the American league) is the group variable we cluster around.

## R Demonstration

### Step 1

Run a regression

```
reg_mlb <- feols(log(salary) ~ years + bavg, data = mlb_data)
```

## R Demonstration

### Step 1

Run a regression

```
reg_mlb <- feols(log(salary) ~ years + bavg, data = mlb_data)
```

### Step 2

Apply `vcov()` or `se()` with the `cluster =` option.

```
## vcov clustered by nl  
vcov(reg_mlb, cluster = ~ nl)  
  
## se clustered by nl  
se(reg_mlb, cluster = ~ nl)
```

## Compare

### Default

```
se(reg_mlb)
```

```
## (Intercept)      years      bavg  
## 0.343071420 0.013222511 0.001335294
```

### Cluster-robust standard error

```
se(reg_mlb, cluster = ~ nl)
```

```
## (Intercept)      years      bavg  
## 0.2495162012 0.0125500623 0.0007155029
```

## R Demonstration

Or, you could add the `cluster` option like below inside `feols()`.

```
reg_mlb <- feols(log(salary) ~ years + bavg, cluster = ~ nl, data = mlb_data)
tidy(reg_mlb)
```

```
## # A tibble: 3 × 5
##   term      estimate std.error statistic p.value
##   <chr>      <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept) 11.0       0.250      44.3    0.0144
## 2 years       0.166     0.0126     13.3    0.0479
## 3 bavg       0.00539   0.000716    7.54   0.0840
```

## In practice

Just like the heteroskedasticity-present case before,

- Estimate the model using OLS (you do nothing special here)
- Assume the error term is clustered and/or heteroskedastic, and estimate the variance of the OLS estimators ( $Var(\hat{\beta})$ ) using cluster-robust standard error estimators
- Replace the estimates from  $\widehat{Var(\hat{\beta})}_{default}$  with those from  $\widehat{Var(\hat{\beta})}_{cl}$  for testing
- But, we do not replace coefficient estimates.

## But does it really work?

Let's run MC simulations to see if the use of the cluster-robust standard error estimation method works

## MC simulation results: R code

```
set.seed(58934)
B <- 1000
t_stat_store <- rep(0, B)
N <- 2000 # number of observations per cluster
G <- 50 # number of groups
Ng <- N/G # number of observations per group

for (i in 1:B){
  #--- error correlated within group ---#
  u <-
  mvrnorm(
    G, mu = rep(0, Ng),
    Sigma = matrix(10, nrow = Ng, ncol = Ng) + diag(Ng)
  ) %>% t() %>% c()

  #--- x correlated within group ---#
  x <-
  mvrnorm(
    G, mu = rep(0, Ng),
    Sigma = matrix(1, nrow = Ng, ncol = Ng) + diag(Ng) * .2
  ) %>% t() %>% c()

  #--- other variables ---#
  y <- 1 + 0 * x + u

  #--- data.frame ---#
  data <- data.frame(y = y, x = x, group = rep(1:G, each = Ng))

  #--- OLS with cluster-robust se---#
  reg <- feols(y ~ x, data = data, cluster = ~ group)

  #--- get vcov ---#
  se_cl <- se(reg)["x"]

  #--- calculate t-stat ---#
  t_stat <- reg$coefficient['x']/se_cl
  t_stat_store[i] <- t_stat
}
```



## MC simulation results

```
### critical value ###  
c_value <- qt(0.95, N-2)  
  
### how often do you reject the null ###  
mean(abs(t_stat_store) > c_value)
```

```
## [1] 0.151
```

## MC simulation results

```
### critical value ###  
c_value <- qt(0.95, N-2)  
  
### how often do you reject the null ###  
mean(abs(t_stat_store) > c_value)
```

```
## [1] 0.151
```

Well, we are still rejecting too often than we should, but it is much better than the default VCOV estimator that rejected 74% of the time.

## MC simulation results

```
### critical value ###
c_value <- qt(0.95, N-2)

### how often do you reject the null ###
mean(abs(t_stat_store) > c_value)
```

```
## [1] 0.151
```

Well, we are still rejecting too often than we should, but it is much better than the default VCOV estimator that rejected 74% of the time.

### Important

- Cluster-robust standard error estimation gets better as the number of groups gets larger
- The number of groups of 2 is too small (the MLB case)
- As a rule of thumb, # of groups larger than 50 is sufficiently large, but we just saw we still over-rejected the null of  $\beta = 0$  three times more than we should.

```

set.seed(58934)
B <- 1000
t_stat_store <- rep(0,B)
N <- 20000 # total number of observations
G <- 1000 # number of groups
Ng <- N/G # number of observations per group

for (i in 1:B){
  #--- error correlated within group ---#
  u <-
  mvrnorm(
    G, mu = rep(0, Ng),
    Sigma = matrix(10, nrow = Ng, ncol = Ng) + diag(Ng)
  ) %>% t() %>% c()

  #--- x correlated within group ---#
  x <-
  mvrnorm(
    G, mu = rep(0, Ng),
    Sigma = matrix(1, nrow = Ng, ncol = Ng) + diag(Ng) * .2
  ) %>% t() %>% c()

  #--- other variables ---#
  y <- 1 + 0 * x + u

  #--- data.frame ---#
  data <- data.frame(y = y, x = x, group = rep(1:G, each = Ng))

  #--- OLS with cluster-robust se---#
  reg <- feols(y ~ x, data = data, cluster = ~ group)

  #--- get vcov ---#
  se_cl <- se(reg)["x"]

  #--- calculate t-stat ---#
  t_stat <- reg$coefficient['x']/se_cl
  t_stat_store[i] <- t_stat
}

```

## MC simulation results

```
### critical value ---#  
c_value <- qt(0.95, N-2)  
  
### how often do you reject the null ---#  
mean(abs(t_stat_store) > c_value)
```

```
## [1] 0.091
```

Better. But, we are still over-rejecting. Don't forget it is certainly better than using the default!