Heteroskedasticity and autocorrelation

Assumptions of the linear regression model

When we introduced the properties of regression, we made a number of assumptions:

1. **Linearity:** $y = \alpha + \beta x + \varepsilon$
2. **Mean-zero errors:** Conditional on the x's, $E(\varepsilon_i = 0)$ (i.e., ε is not correlated with x)

3. **Homoskedasticity:** Conditional on the x's, $Var(\varepsilon_i) = \sigma^2$ (i.e., the variance is the same for everyone, regardless of their x)
4. **No autocorrelation:** Conditional on the x's, $Cov(\varepsilon_i, \varepsilon_j) = 0$ for $i \neq j$

When we first went through these assumptions, you might have thought that some of them were not particularly realistic

(Even if you didn't think this, it's good practice to carefully examine assumptions)

We are now going to see how we can *relax* some of these assumptions to make our regression analysis more broadly applicable

We'll start with the last two assumptions: homoskedastic and non-autocorrelated errors

Note: For simplicity, we'll do this analysis using univariate regression, though our conclusions will also carry over to multivariate regression

Heteroskedasticity

Recall that **homoskedasticty** ("equal scatter") means that, conditional on the x's, $Var(\varepsilon_i) = \sigma^2$

In the model

$$y = \alpha + \beta x + \varepsilon,$$

this means that the "other factors" ε that affect y have the same variance for everyone, regardless of their value of x

As we discussed before, this might not be realisic

For example, if our model is

$$wage_i = \alpha + \beta educ_i + \varepsilon_i,$$

we might be concerned that people with more education can work in a greater variety of jobs, and therefore might have more variance in their error term

Heteroskedasticity ("unequal scatter") is when $Var(\varepsilon_i) = \sigma_i^2$, or the variance of the error varies across units (conditional on the x's)

Consequences of heteroskedasticity

What happens if ε is heteroskedastic?

The good news is that heteroskedasticity does not affect unbiasedness or consistency, so it is still true that

1. OLS is unbiased: $E(b) = \beta$, and
2. OLS is consistent: $b \to \beta$ as $n \to \infty$

These facts are easy to prove: When we derived unbiasedness and consistency, we didn't use the homoskedasticity assumption, so it doesn't affect them

Unfortunately, there is also bad news

The first piece of bad news is that, when ε is heteroskedastic, the "default" standard errors (which assume homoskedasticity) will be incorrect – they don't give a good estimate of the variance of b

Why does this matter?

One reason is that when we do hypothesis testing, we reject the null hypothesis that $\beta = \beta_0$ if the absolute value of the t-stat exceeds the critical value:

$$\left| \frac{b - \beta_0}{se(b)} \right| > t_{\alpha}^{(n-2)}$$

Also recall that we designed our test so that we only reject the null when it is true (i.e., commit a **Type I error**) 5% of the time (assuming our α level is .05)

Suppose that heteroskedasticity makes $se(b)$ smaller than it should be. Then our t-stat is larger than it should be, which means that we'll reject the null hypothesis when it is actually true more often than we want

In other words, our hypothesis tests will be inaccurate. This is bad science – we'll be more likely to conclude that a policy has an effect, even when it doesn't

As another example of this, recall that a 95% confidence interval takes the form

$$b \pm se(b) * t_{.025}^{(n-2)}$$

If we use a value of $se(b)$ that's smaller than it should be, our confidence intervals will also be too small (they'll become "overconfidence intervals")

So the probability that the true value of β lies in the confidence interval will be smaller than 95%

The second piece of bad news is that, when ε is heteroskedastic, the Gauss-Markov theorem no longer applies to OLS

Recall that the Gauss-Markov theorem says that b has the lowest variance among all linear, unbiased estimators (OLS is "BLUE")

Unfortunately (although we didn't go through it), the proof of the Gauss-Markov theorem relies on the assumptions that ε is homoskedastic and non-autocorrelated

Thus, under heteroskedasticity, OLS is no longer the unbiased (and linear) estimator with the smallest variance

Let’s illustrate some of these failures with a simulation

Previously, we did simulations to show that (under homoskedasticity), our Type-I error rates and confidence intervals were reliable

Let’s look at how those simulations hold up under homoskedasticity

```r
set.seed(12345)
t <- rep(0,1000)
t2 <- rep(0,1000)
for (i in 1:1000) {
  x <- rnorm(100)
  e <- rnorm(100)
  y <- 1 + 2*x + e # homoskedastic
  y2 <- 1 + 2*x + 2*x*e # heteroskedastic
  b <- lm(y ~ x)$coef[2]
  se <- sqrt(vcov(lm(y ~ x))[2,2])
  b2 <- lm(y2 ~ x)$coef[2]
  se2 <- sqrt(vcov(lm(y2 ~ x))[2,2])
  t[i] <- (b-2)/se
  t2[i] <- (b2-2)/se2
}
mean(abs(t)>1.96)
```

```
[1] 0.055
```

```r
mean(abs(t2)>1.96)
```

```
[1] 0.28
```

With homoskedastic errors, we only reject the null (which is true) 5% of the time. With heteroskedastic errors, we reject 28% of the time!

Correcting for heteroskedasticity

In sum, heteroskedasticity makes it so that the "default" standard errors are wrong and OLS is no longer BLUE

We will take the view that the first problem is very important, but the second one not so much

Put differently, just because OLS is no longer the "best" estimator doesn't mean that it's not still *good*—after all, it's still unbiased and consistent

So we're not so worried about the Gauss-Markov theorem no longer holding

On the other hand, even if OLS is still good, in order to use it, we need accurate standard errors that we can use to conduct valid hypothesis tests, create accurate confidence intervals, etc.

Recall that the standard error is an estimate of the standard deviation, which is the square root of the variance of b

So as a first step, we need to revisit the variance of b under homoskedasticity

In this case, using the deviations-from-means trick, we have

$$
\begin{aligned}
Var(b) &= Var\left(\beta + \frac{\sum_i x_i \varepsilon_i}{\sum_i x_i^2}\right) = Var\left(\frac{\sum_i x_i \varepsilon_i}{\sum_i x_i^2}\right) \\
&= \frac{1}{\left(\sum_i x_i^2\right)^2} Var\left(\sum_i x_i \varepsilon_i\right) = \frac{1}{\left(\sum_i x_i^2\right)^2} \sum_i Var(x_i \varepsilon \\
&= \frac{1}{\left(\sum_i x_i^2\right)^2} \sum_i x_i^2 Var(\varepsilon_i) = \frac{\sum_i x_i^2 \sigma_i^2}{\left(\sum_i x_i^2\right)^2}
\end{aligned}
$$

This only differs from our previous deviation by one line. If ε were homoskedastic, we would get back our old formula:

$$Var(b) = \frac{\sum_i x_i^2 \sigma_i^2}{\left(\sum_i x_i^2\right)^2} = \frac{\sigma^2 \sum_i x_i^2}{\left(\sum_i x_i^2\right)^2} = \frac{\sigma^2}{\sum_i x_i^2}$$

Also note that if our variables are not in deviations from mean form, we can put them in that form, so our formula becomes:

$$Var(b) = \frac{\sum_i (x_i - \bar{x})^2 \sigma_i^2}{\left[\sum_i (x_i - \bar{x})^2\right]^2}$$

We can use these formulae to see how using the default standard errors will lead us astray in the presence of heteroskedasticity

Under homoskedasticity, there is no relationship between $\sigma_i^2 = \sigma^2$ and x_i

Under heteroskedasticity,

$$Var(b) = \frac{\sum_i (x_i - \bar{x})^2 \sigma_i^2}{\left[\sum_i (x_i - \bar{x})^2\right]^2}$$

If those with large values of $(x_i - \bar{x})$ tend to have large values of σ_i^2 , the numerator will be very large (and vice versa)

Thus, using the default standard errors in the presence of heteroskedasticity can *under-* or *over-*state the true variance, depending on the relationship between x_i and σ_i^2

Robust standard errors

So how do we “fix” the standard errors when there is heteroskedasticity?

Again, under homoskedasticity,

$$Var(b) = \frac{\sum_i (x_i - \bar{x})^2 \sigma_i^2}{\left[\sum_i (x_i - \bar{x})^2\right]^2}$$

If we knew σ_i^2, we could just plug into the formula. The problem is that we don’t

But recall that $\sigma_i^2 = Var(\varepsilon_i) = E[\varepsilon_i - E(\varepsilon_i)]^2 = E(\varepsilon_i^2)$, where the last line follows because $E(\varepsilon_i) = 0$

Also recall that $e_i = y_i - a - bx_i$ is our estimate of $\varepsilon_i = y_i - \alpha - \beta x_i$

This suggests that we might be able to "estimate" $\sigma_i^2 = E(\varepsilon_i^2)$ with the squared *residual* e_i^2

Since it's only based on one observation, e_i^2 is not a particularly good estimate of $E(\varepsilon_i^2)$, but it turns out that the average of many bad estimates produces a good one

White's heteroskedasticity-robust standard error estimator (aka "White SEs" or "robust SEs") does just that:

$$\frac{\sum_i (x_i - \bar{x})^2 e_i^2}{\left[\sum_i (x_i - \bar{x})^2\right]^2}$$

This estimate is named for Hal White, who introduced the idea into econometrics

Let's see how do to this in R. First, we'll look at the default SEs:

```r
library(tidyverse)
gril <- read_csv("griliches.csv")
gril$w <- exp(gril$lw)
model <- lm(w ~ s, data=gril)
summary(model)
```

```
Call:
lm(formula = w ~ s, data = gril)

Residuals:
    Min      1Q  Median      3Q     Max 
-344.94  -84.91  -20.59   60.90  676.12 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -125.295     28.458  -4.403 1.22e-05 ***
s             33.511      2.094  16.002  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 128.6 on 756 degrees of freedom
```

We can get robust SEs using the `sandwich` and `lmtest` packages:

```
library(sandwich)
library(lmtest)
coeftest(model, vcov = vcovHC(model, type="HC1"))
```

```
t test of coefficients:

             Estimate Std. Error t value  Pr(>|t|)
(Intercept) -125.2950    31.4868 -3.9793 7.578e-05 ***
s             33.5107     2.4562 13.6434 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The `"HC1"` option stands for “heteroscedasticity-consistent” (it turns out that there are a few different vairations on these, this is the most common one)

The standard error increases from 2.09 to 2.46, a fairly substantial increase

Note that the coefficient estimates don’t change. We’re still

estimating b the same way, we're only estimating the SEs differently

Now that we know how to do these in R, let's see how they do in our simulation:

```r
set.seed(12345)
t <- rep(0,1000)
t2 <- rep(0,1000)
t3 <- t2
for (i in 1:1000) {
  x <- rnorm(100)
  e <- rnorm(100)
  y <- 1 + 2*x + e # homoskedastic
  y2 <- 1 + 2*x + 2*x*e # heteroskedastic
  b <- lm(y ~ x)$coef[2]
  se <- sqrt(vcov(lm(y ~ x))[2,2])
  b2 <- lm(y2 ~ x)$coef[2]
  se2 <- sqrt(vcov(lm(y2 ~ x))[2,2])
  t[i] <- (b-2)/se
  t2[i] <- (b2-2)/se2
  se3 <- sqrt(vcovHC(lm(y2 ~ x), type="HC1")[2,2])
  t3[i] <- (b2-2)/se3
}
```

Here are the results:

```
1 mean(abs(t)>1.96)
```

```
[1] 0.055
```

```
1 mean(abs(t2)>1.96)
```

```
[1] 0.28
```

```
1 mean(abs(t3)>1.96)
```

```
[1] 0.087
```

The robust SEs aren't perfect here (with more observations, they'd be better), but 8.7% is a huge improvement on 28%

Testing for heteroskedasticity

How do we know whether we should use robust SEs?

Most of the time, we don't need to worry about this question. We can just *always* use robust SEs, since they are valid even if ε is homoskedastic

However, it is possible to test for heteroskedasticity

To see how, recall that heteroskedasticity occurs when $\sigma_i^2 = E(\varepsilon_i^2)$ is related to x_i

Also recall that the linear regression model implies that

$$E(y_i) = \alpha + \beta x_i$$

Thus, if we *knew* ε_i, we could test for heteroskedasticity by regression ε_i^2 on x_i, since this would tell us how $\sigma_i^2 = E(\varepsilon_i^2)$ is related to x_i

Although we don't have data on ε_i, we know that we can use e_i as it's "stand in"

White's test for heteroskedasticity[1] consists of regressing e_i^2 on x_i and x_i^2

We include x_i^2 to allow for a *nonlinear* relationship between e_i and x^2 (we'll say more about allowing for nonlinearity later)

Under homoskedasticity, e_i^2 and x_i shouldn't be related, so the R^2 from this regression should be small. If R^2 is large, we reject the null hypothesis of homoskedasticity (and conclude that ε is heteroskedastic)

1. You can probably guess who invented this

Technically, the test statistic is nR^2, which has a χ^2 ("chi square") distribution with 2 degrees of freedom. So we reject the null if nR^2 is bigger than the $\chi^2_\alpha(2)$ critical value, which we can look up online (or we can get R to tell us, see below)

Alternatively, we can use an F-test of the joint null hypothesis that the coefficient on x and x^2 are zero

```r
gril$e2 <- model$residuals^2
r2 <- summary(lm(e2 ~ s + I(s^2), data=gril))$r.squared
(white.test.stat <- 758*r2)
```

```
[1] 34.51379
```

```r
qchisq(p=.05, df=2, lower.tail=FALSE) # critical value
```

```
[1] 5.991465
```

```r
pchisq(white.test.stat, df=2, lower.tail=FALSE)
```

```
[1] 3.202025e-08
```

The test statistic is 34.51. The `qchisq` command tells us the critical value, which is 6, so we can reject the null of homoskedasticity. The `phisq` command tells us the p-value, which is very close to zero, again telling us that we can reject

Since the test gets a little more compicated in the multivariate case, it's convenient to have software that can implement it. In R, this can be done using the `whitestrap` package:

```
library(whitestrap)
white_test(model)
```

```
White's test results

Null hypothesis: Homoskedasticity of the residuals
Alternative hypothesis: Heteroskedasticity of the residuals
Test Statistic: 34.51
P-value: 0
```

Autocorrelation

The **no autocorrelation** assumption holds that errors for different units are uncorrelated

Autocorrelation occurs when this assumption fails, so that $Cov(\varepsilon_i, \varepsilon_j) \neq 0$ for $i \neq j$

Autocorrelation arises naturally in a number of cases

For example, suppose that we have a **time-series model**:

$$GDP_t = \alpha + \beta Exports_t + \varepsilon_t,$$

where now we are indexing observations by t instead of i to reflect the fact that they are observations on one country at different points in time

In this model, ε_t represents everything *besides* exports that affects GDP. Since it's likely that these macroeconomic factors are correlated from year to year, we might expect this model to exhibit autocorrelation

As another example, consider the familiar model

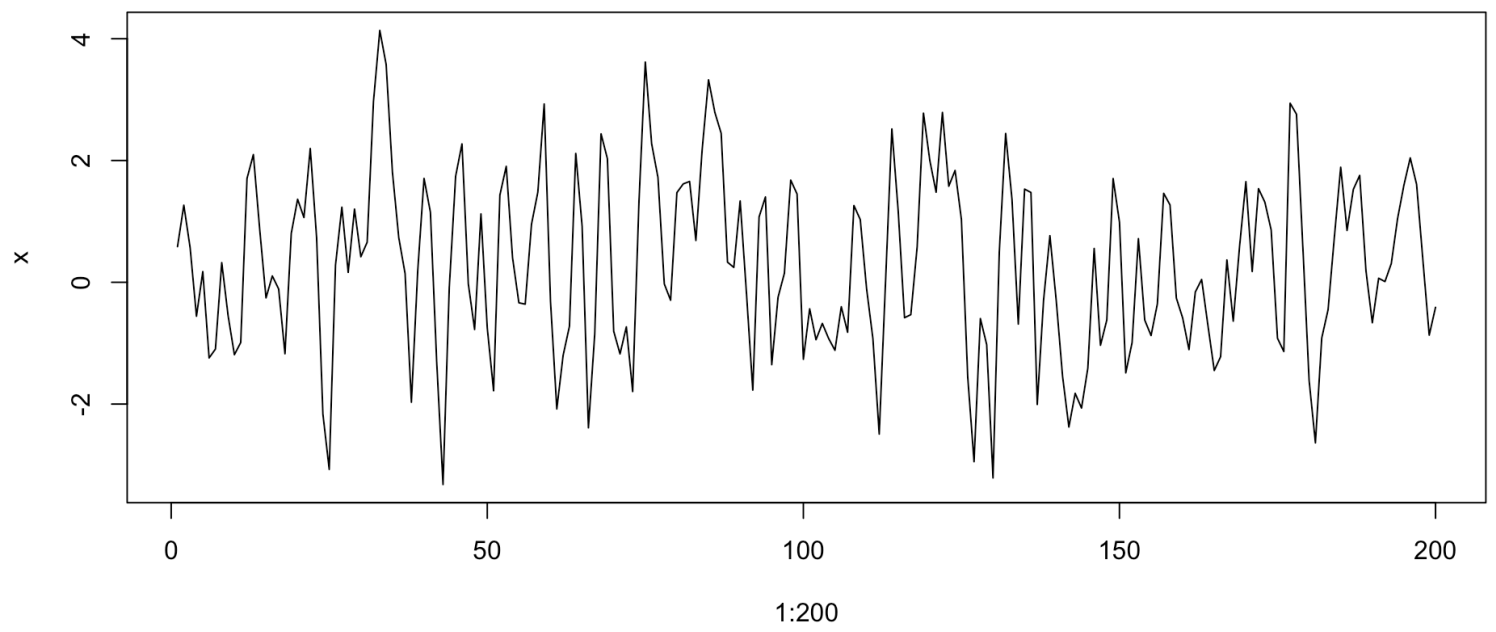
$$wage_i = \alpha + \beta educ_i + \varepsilon_i$$

Here, ε_i represents all of the factors *besides* education that affects the wage

Since individuals living in the same state may face similar local economic conditions (another factor affecting wages), their errors might be correlated

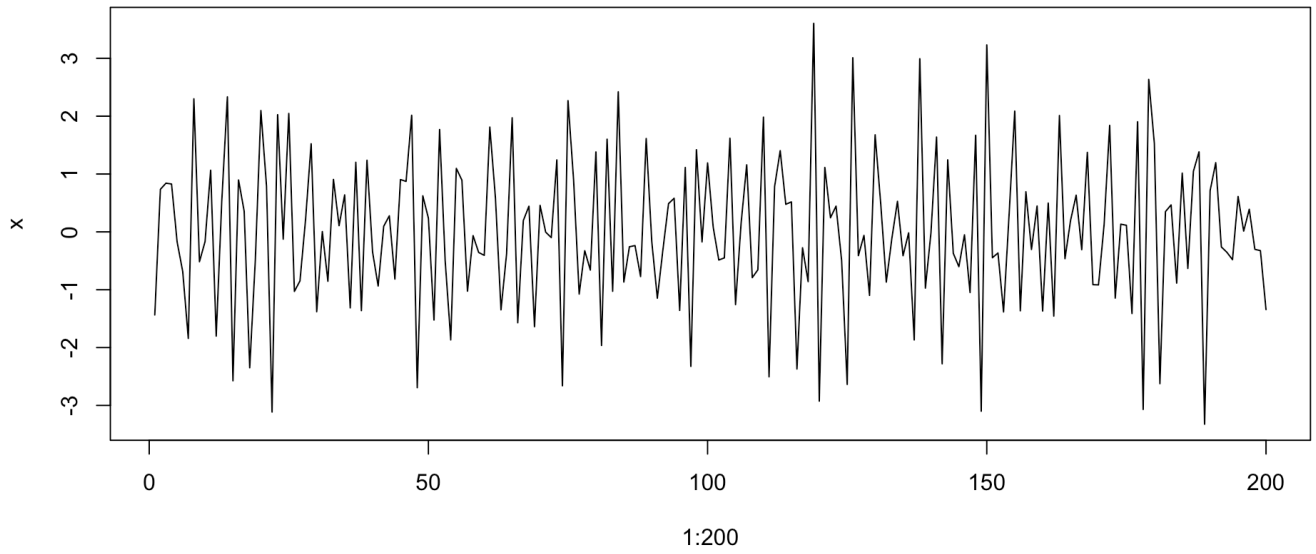
This form of autocorrelation is known as **clustering**

In a time series context, here is what autocorrelation looks like:



When x is positive, it tends to stay positive for awhile, until a big “shock” comes and makes it negative for awhile

Actually, the previous plot shows “positive” autocorrelation. Here’s what negative autocorrelation looks like:



Now, the graph varies more wildly, because any time x is positive today, it tends to be negative tomorrow, and vice versa

Consequences of autocorrelation

Once again, there's good news and bad news

The good news is that, under autocorrelation, OLS is still unbiased and consistent (again, this is because we didn't use autocorrelation when deriving those properties)

However, it is also true that, when ε is autocorrelated,

1. The default standard error estimate (which assumes non-autocorrelation) is wrong, and
2. The Gauss-Markov theorem (which also assumes non-autocorrelation) no longer holds

As before, we're going to be less concerned about the second problem. OLS may not be the "best," but it's still unbiased and consistent, and "good" is good enough for us

But we are concerned about the second problem, since if we're going to use OLS, we need to be able to accurately estimate the standard errors

To see the problem with the default standarda errors in the presence of autocorrelation, we need to revisit our derivation of the variance of b (we'll go back to assuming that ε are homoskedastic, so we only have to tackle one problem at a time)

Revisiting our derivation, we have

$$Var(b) = Var\left(\beta + \frac{\sum_i x_i \varepsilon_i}{\sum_i x_i^2}\right) = Var\left(\frac{\sum_i x_i \varepsilon_i}{\sum_i x_i^2}\right)$$
$$= \frac{1}{\left(\sum_i x_i^2\right)^2} Var\left(\sum_i x_i \varepsilon_i\right)$$

When we were assuming non-autocorrelation, at this point we used the fact that if $w_1, \ldots, w_n$ are *independent* RVs, then $Var(\sum_i w_i) = \sum_i Var(w_i)$

But autocorrelation makes it so that the variables $x_i \varepsilon_i$ are not independent, so we have to change our derivation

How do we change the derivation? Well, it turns out that if $w_1, \ldots, w_n$ are correlated, then

$$Var(\sum_i w_i) = \sum_i Var(w_i) + 2 \sum_{i=2}^{n} \sum_{j=1}^{i-1} Cov(w_i, w_j).$$

The second term is basically 2 times the covariance between all possible combinations of our n variables. For example, if $n = 3$, that term becomes

$$2[Cov(w_2, w_1) + Cov(w_3, w_1) + Cov(w_3, w_2)]$$

In light of this, our derivation becomes

$$
\begin{aligned}
V(b) &= \frac{1}{\left(\sum_i x_i^2\right)^2} Var\left(\sum_i x_i \varepsilon_i\right) \\
&= \frac{1}{\left(\sum_i x_i^2\right)^2} \left[\sum_i Var(x_i \varepsilon_i) + 2 \sum_{i=2}^{n} \sum_{j=1}^{i-1} Cov(x_i \varepsilon_i, x_j \varepsilon_j)\right] \\
&= \frac{1}{\left(\sum_i x_i^2\right)^2} \left[\sum_i x_i^2 \sigma^2 + 2 \sum_{i=2}^{n} \sum_{j=1}^{i-1} x_i x_j Cov(\varepsilon_i, \varepsilon_j)\right] \\
&= \frac{\sigma^2}{\sum x_i^2} + 2 \frac{\sum_{i=2}^{n} \sum_{j=1}^{i-1} x_i x_j Cov(\varepsilon_i, \varepsilon_j)}{\left(\sum_i x_i^2\right)^2}
\end{aligned}
$$

So, under autocorrelation,

$$V(b) = \frac{\sigma^2}{\sum x_i^2} + 2\frac{\sum_{i=2}^{n}\sum_{j=1}^{i-1} x_i x_j Cov(\varepsilon_i, \varepsilon_j)}{\left(\sum_i x_i^2\right)^2}$$

(Of course, if the observations aren't in deviation from means forms, we can replace them with deviations from means)

Bottom line: We need an estimate of the second term

Can you guess how we're going to estimate it?

If you guessed that we're going to use $e_i e_j$ to estimate $Cov(\varepsilon_i, \varepsilon_j)$, you're *almost* right

It turns out that for technical reasons, we need to do something a little more complex

The **Newey-West autocorrelation-consistent variance estimator** of the second term is

$$2\frac{\sum_{i=1}^{n}\sum_{j=\max(1,i-L)}^{i-1}\left(1-\frac{i-j}{L+1}\right)(x_i-\bar{x})(x_j-\bar{x})e_i e_j}{\left(\sum_i (x_i-\bar{x})^2\right)^2}$$

$$2\frac{\sum_{i=1}^{n}\sum_{j=\max(1,i-L)}^{i-1}\left(1-\frac{i-j}{L+1}\right)(x_i-\bar{x})(x_j-\bar{x})e_ie_j}{\left(\sum_i(x_i-\bar{x})^2\right)^2}$$

What's going on here? This assumes that when i is closer to j, $Cov(\varepsilon_i, \varepsilon_j)$ is larger. It only includes covariances between observations that are within L periods of eachother (where L is some number like $\sqrt[4]{n}$). It also puts more weight on observations that are closer together, so that $\left(1-\frac{i-j}{L+1}\right)$ is larger

Let’s see how this works. First, we’ll generate some autocorrelated data:

```
library(lmtest)
library(sandwich)
e0 <- rnorm(1000)
e <- .25*dplyr::lag(e0) + e0
e[1] <- e0[1]
x0 <- rnorm(1000)
x <- .25*dplyr::lag(x0) + x0
x[1] <- x0[1]
y <- 1 + 2*x + e
```

Next, we'll compare default and Newey-West SEs (we can get these using the `coeftest` and `vcovHAC` commands from the `lmtest` and `sandwich` packages; `HAC` stands for "heteroskedasticity and autocorrelation consistent"):

```
1 summary(lm(y ~ x))
```

```
Call:
lm(formula = y ~ x)

Residuals:
    Min      1Q  Median      3Q     Max 
-3.3671 -0.6749  0.0122  0.7290  3.4865 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  0.95875    0.03267   29.35   <2e-16 ***
x            1.98199    0.03180   62.32   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.032 on 998 degrees of freedom
```

```r
coeftest(lm(y ~ x), vcov=vcovHAC)
```

```
t test of coefficients:

            Estimate Std. Error t value  Pr(>|t|)
(Intercept) 0.958749   0.039614  24.202 < 2.2e-16 ***
x           1.981993   0.035973  55.097 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Now, we'll compare the default and Newey-West SEs in a simulation:

```r
set.seed(12345)
t <- rep(0,1000)
t2 <- t
for (j in 1:1000) {
  e0 <- rnorm(1000)
  e <- .25*dplyr::lag(e0) + e0
  e[1] <- e0[1]
  x0 <- rnorm(1000)
  x <- .25*dplyr::lag(x0) + x0
  x[1] <- x0[1]
  y <- 1 + 2*x + e
  b <- lm(y ~ x)$coef[2]
  se <- sqrt(vcov(lm(y ~ x))[2,2])
  t[j] <- (b - 2)/se
  se2 <- sqrt(vcovHAC(lm(y ~ x))[2,2])
  t2[j] <- (b - 2)/se2
}
mean(abs(t)>1.96)
```

```
[1] 0.063
```

```r
mean(abs(t2)>1.96)
```

```
[1] 0.058
```

Here, the default standard errors reject too often, while the

autocorrelation-consistent standard errors do better, although they don’t get us all the way to 5 (they would if the sample were larger)

Clustering

Recall that clustering is a version of autocorrelation where errors are correlated between units in the same “cluster”

For example, we might want to allow individuals living in the same state to have correlated errors

Or if we’re analyzing student success, we might want to allow for the possibiltiy that students in the same school (who have the same teachers) to be correlated

Since this is really just a form of clustering, we don't really need to do anything different

But in this case, the Newey-West consistent standard error estimator of the second term takes the particular form

$$2\frac{\sum_{i=1}^{n}\sum_{j=1}^{i-1} d_{ij}(x_i-\bar{x})(x_j-\bar{x})e_i e_j}{\left(\sum_i (x_i-\bar{x})^2\right)^2},$$

where d_{ij} equals one if units i and j are in the same cluster, and zero otherwise

(Actually, for all Newey-West-type variances, there are variants that are also robust to heteroskedasticity)

To see how to do this in R, let's revisit the capital punishment example we looked at before

In this data, observations represent murder rates and executions for different states at different points in time

If we think that the error terms (other factors that affect murder rates) for a given state are correlated over time, we might want to allow for clustering

Here are the default SEs:

```r
library(haven)
mur <- read_dta("MURDER.dta")
mur.mod1 <- lm(mrdrte ~ exec + unem, data=mur)
summary(mur.mod1) # default SEs
```

```
Call:
lm(formula = mrdrte ~ exec + unem, data = mur)

Residuals:
   Min     1Q Median     3Q    Max
-9.175 -3.472 -1.416  1.114 69.143

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.3481     2.6872   0.130  0.89710
exec          0.1650     0.1939   0.851  0.39601
unem          1.2589     0.4374   2.878  0.00458 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We can get clustered SEs using the `lmtest` and `sandwich` packages

```
1  coeftest(mur.mod1, vcov=vcovCL(mur.mod1, cluster = ~state))
```

```
t test of coefficients:

             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.34812    2.66137  0.1308  0.89611
exec         0.16502    0.14981  1.1015  0.27243
unem         1.25891    0.63329  1.9879  0.04865 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In this case, accounting for clustering decreases the SE on `exec`, but increases the SE on `unem`