Chapter 4:
Dood! Check the Requirements

Overview:

There are several requirements for ordinary least squares regression to be applicable. In this chapter, we cover them, their tests, and their relative importance. Focus on the relationship between the assumption/requirement and the tests used. As these assumptions (requirements) are used for some other fitting methods, not just ordinary least squares, the lessons learned here are helpful in the future.

However, realize that each fitting method has its own set of assumptions/requirements. For instance, median regression (Chapter 8) requires only the constant expected residual value. Maximum likelihood for Poisson regression (Chapter 13) requires that, plus a specific relationship between the expected value and variance.

Forsberg, Ole J. (November 18, 2023). "Dood! Check the Requirements." In *Linear Models and Řurità Kràlovstvì*. Version 0.704442ζ.

Chapter Contents

❧❧❧

In Chapter 2, we created the ordinary least squares regression technique. The mathematics of the situation required $\det \mathbf{X}'\mathbf{X} \neq 0$. Because this corresponds to the requirement that the design matrix $\mathbf{X}$ be of full column rank, the requirement is met when the independent variables are not linear combinations of each other.

Chapter 3 included the requirement

$$\varepsilon_i \stackrel{\text{iid}}{\sim} \mathcal{N}\left(0; \sigma^2\right) \tag{4.1}$$

This allowed us to move beyond the pure mathematics of Chapter 2 and begin making inferences about the population based on the sample.

equivalently

The requirement that $\varepsilon_i \stackrel{\text{iid}}{\sim} \mathcal{N}\left(0; \sigma^2\right)$, equivalently $\mathbf{E} \sim \mathcal{N}\left(\mathbf{0}; \sigma^2\mathbf{I}\right)$, has several parts to it: normality, constant expected value (of zero), constant variance, and independence. This chapter takes these assumptions and explores them. Both graphical and numeric tests are presented. Throughout it all, we will rely heavily on `R` to generate the residuals, create the graphics, and perform the tests.

> ***Note***: Some `R` functions require an additional package to be loaded. When such is the case, I indicate that in a special font. For example, the runs test (Bradley 1968) is implemented in the `lawstat` package as the function `runs.test`. The `lmtest` has the `bptest`, which performs the Breusch-Pagan test.

Some R functions are not included in any package and must be downloaded from the Internet each session. Such functions require you to run

```
source("http://rfs.kvasaheim.com/rfs.R")
```

at the start of the script. Once that line is run, you can then run several additional functions, like `overlay`, `hetero.test`, `shapiroTest`, and an advanced version of `runs.test`.

4.1: Normality

Let us tackle the requirement of normality first. In Chapter 3, this assumption allowed us to exactly determine the distribution of the test statistics, which allowed us to exactly calculate p-values and confidence intervals. In this section, let us look at how to test your model that this assumption/requirement is met.

4.1.1 Graphical Tests

From your elementary statistics course, you were most likely introduced to a pair of graphical tests of normality: Q-Q plots and histograms. In this section, we examine each when the residuals are generated from a normal process and when they are not. To do this, let's do some experiments using R.

To begin, let us generate 100 residuals from a normal process, with mean $\mu = 0$ and standard deviation $\sigma = 2$:

```
e = rnorm(100, m=0, s=2)
```

The `rnorm` function generates `n` random values from a normal distribution with mean `m` and standard deviation `s`. Running this line only stores those 100 random values in the `e` variable.

Warning: *Be aware that R, like most statistics programs, parameterizes the normal distribution using the standard deviation instead of the variance.*

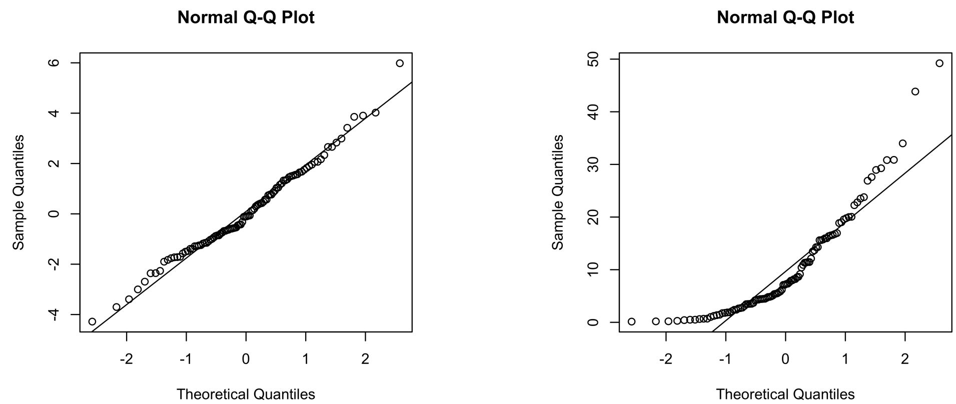

Figure 4.1: *Default normal quantile-quantile plots The left panel is for the randomly-generated normal data. Note that the circles fall close to the diagonal target line. The right panel is for the randomly-generated Exponential data. Note that the circles do not tend to fall close to the diagonal target line. There is a distinct bow in them, which signifies the residuals have a right (positive) skew.*

With that one line, we have some 'residuals' to play with that are generated "under the null hypothesis" that they are generated from a normal distribution. This will allow us to better understand what the normal distribution looks like.

Q-Q Plot: So, let us look at how to generate a normal-based quantile-quantile (Q-Q) plot using R.

```
qqnorm(e)
```

quantile-quantile

That's it. After running that one line, R creates the usual Q-Q plot for the normal quantiles. It is a default graphic, so it does not look awesome, but it does get the point across to the statistician.

One shortcoming of the default Q-Q plot in R is that it does not provide the diagonal line. You can add it by also running the command

```
qqline(e)
```

Figure 4.1, left panel, is the graphic produced by running these lines. Note that most of the circles cluster around the diagonal target line. Let us compare that graphic to a Q-Q plot from a non-normal distribution. The non-normal distribution will be the Exponential($\lambda = 0.10$) distribution.[1]

```
enon = rexp(100, rate=1/10)
qqnorm(enon)
qqline(enon)
```

Compare the two graphics in Figure 4.1. Note that the shape of the Q-Q plot from the normal residuals (left) is very different from the one generated from a skewed distribution (right). This particular shape indicates that the distribution of the residuals is positively skewed (right skewed).

Histogram: In the previous section, we examined quantile-quantile plots. We saw that a Q-Q plot with the dots aligning closely to the diagonal target line suggests Normality. In this section, we will use histograms to obtain a better view of the distribution of the data.

Again, let us generate Normally-distributed residuals.

```
e = rnorm(100, m=0, s=2)
```

The function to create a basic histogram is just

```
hist(e)
```

or

```
overlay(e)
```

This basic histogram produced is provided as in the left panel of Figure 4.2. Note that it has the stereotypical "bell shape" to it, thus suggesting the data come from a normal distribution. This is the shape you are seeking when using a histogram to explore the distribution of the residuals.

What does the histogram look like when the data come from a highly skewed distribution like the Exponential($\lambda = 0.10$) distribution above? See Figure 4.2, right panel.

Note the lack of bell shape in the left histogram of Figure 4.2 (and enhanced in Figure 4.3. In fact, one can easily see that the residuals are pos-

[1] This distribution is highly right-skewed. We will come back to the Exponential distribution several times to better understand how assumption violations affect our conclusions.

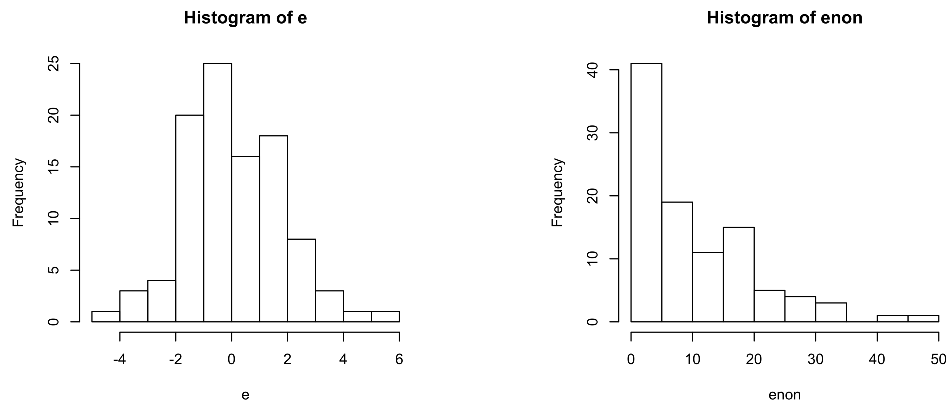

Figure 4.2: *Default histograms for the randomly-generated residuals. In the left panel, note the basic bell shape to the histogram. Such a shape suggests that the residuals come from a normal distribution. In the right panel, note the lack of a bell shape to the histogram. Such a shape suggests that the residuals do* not *come from a normal distribution.*

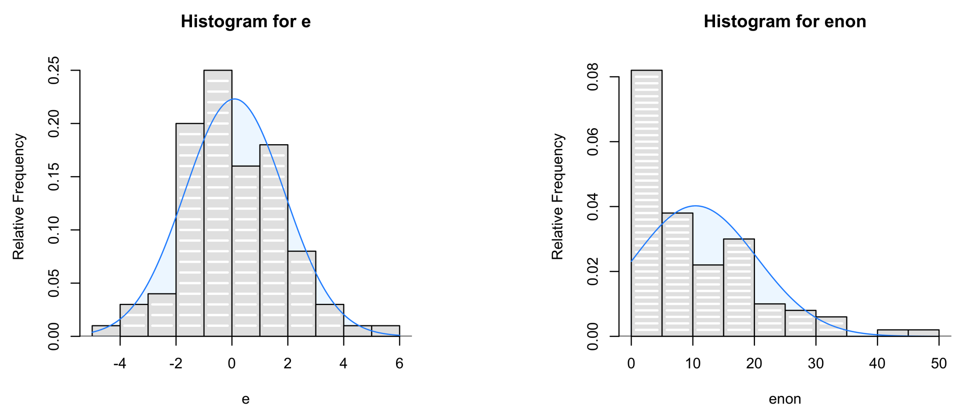

Figure 4.3: *Enhanced histograms for the randomly-generated residuals. These are the same data as in Figure 4.2. The difference is that these graphics also overlay the normal density function to aid in comparison.*

itively skewed. Recall that the direction of the skew is in the same direction as the long tail.

> ***Note***: I find using the histogram much easier than using the Q-Q plot. I can "see" how the residuals are distributed in the histogram. In the Q-

Q plot, I have to interpret much more, remembering what the different shapes indicate.

Note: With that said, however, a Q-Q plot is much more useful than a histogram when there are few data points. So, if your sample size is small, you will want to use a Q-Q plot. Otherwise, a histogram may be best.

4.1.2 Numeric Tests

Because of the importance of the normal distribution in Statistics, there are several Normality tests available, with more being created yearly. I mean, we gotta award those Ph.D. degrees for something, right? **PhDs? PhDd?**

The reality is that it matters little which Normality test you use. As you will see in Section 4.1.3, the Normality assumption in OLS reflects the Normality assumption in t-tests and the like. The Central Limit Theorem ensures that a sufficiently large sample size makes the distribution of the *data* largely irrelevant; the distribution of sample sums is "close" to normal. **CLT**

So, for me, I default to using the Shapiro-Wilk test as my Normality test. In base `R`, this test is implemented as the function `shapiro.test`. In the `RFS` package (or `rfs` file), it is implemented as `shapiroTest`. I recommend the latter, because it adds additional functionality.

So, let us generate some Normally-distributed residuals and see how we can use the Shapiro-Wilk test.

Let us first set the random number seed. Technically, there is no such thing as a truly random number when they are generated by a computer. This is because these pseudo-random numbers are functions of a number called a seed. If the seed is specified, then our "random numbers" will be the same. **prng**

To check this out, run the following three lines:

```
set.seed(370)
e = rnorm(100, m=0, s=2)
head(e, 5)
```

The numbers you get are

```
-0.5566842 -1.7264618  1.3320962 -0.5392740  0.3477423
```

Setting the random number seed ensured that your values are mine.

Note: The `head` function returns the first six values in the variable, by default. A similar function is the `tail` function, which returns the *last* six values in the variable. I included the number 5 to have `R` only return five numbers.

To perform the Shapiro-Wilk test, just run the following line

```
shapiroTest(e)
```

The output you get will be

```
  Shapiro-Wilk normality test

data:  e
W = 0.98554, p-value = 0.3472
```

The p-value is interpreted in the usual way: If the p-value is greater than α, then you fail to reject the null hypothesis. If the p-value is less than α, then you reject the null hypothesis. The null hypothesis in this case is that the data are generated from a normal process (that the data come from a normal distribution).[2]

Since the p-value is greater than $\alpha = 0.05$, we fail to reject the null hypothesis. We do not have sufficient evidence that the data are non-normal.

Note: We did *not* conclude that the data *do* come from a normal distribution. We only concluded that there is no significant evidence to the contrary.

[2]The normal distribution is identical to the Gaussian distribution. The only difference is the discipline. In much of statistics, it is known as the normal distribution. However, when we get to generalized linear models (Chapter 10), this same distribution will be called the Gaussian distribution. In the Francophone world, this distribution is routinely called the Laplace-Gauss distribution.

To drive home this point, let us generate our data from a non-normal distribution and perform the Shapiro-Wilk test: **golf or car?**

```
set.seed(370)
e = rt(100, df=100)
shapiroTest(e)
```

By the way that we generated the data, the residuals do not come from a normal distribution; they come from a Student's t distribution with 100 degrees of freedom ($\nu = 100$). Here is the output

```
  Shapiro-Wilk normality test

data:  y
W = 0.98979, p-value = 0.6474
```

Note that the p-value is greater than our α value of 0.05. As such, we again fail to reject the null hypothesis. There is not sufficient evidence that the residuals do not come from a normal distribution.

However, we absolutely *know* they don't.

Warning: *The Lesson— Never* accept *the null hypothesis. The p-value depends on how reasonable the null hypothesis is*, as well as *how good the test is and how large the sample is.*

> ***Note***: Remember that when making decisions, there are really three usual decisions: Yes, No, and Maybe. In statistics, since we are just testing how reasonable the null hypothesis is, we only have two possible decisions: Reject and Don't Reject. These correspond to "the null hypothesis is wrong" and "the null hypothesis is right *or* we don't know."

Let us now generate severely non-normal residuals and use the Shapiro-Wilk test to see if they do come from a normal distribution:

```
set.seed(370)
e = rexp(100, rate=1/10)
shapiroTest(e)
```

Here are the results:

```
	Shapiro-Wilk normality test

data:  y
W = 0.68002, p-value = 1.891e-13
```

Note that the p-value of $1.891 \times 10^{-13} = 0.000\,000\,000\,000\,189\,1$ is much less than $\alpha = 0.05$. That strongly indicates that the residuals were not generated from a normal process.

Note: Remember the Central Limit Theorem (Section S.6.4) and the effect of sample size on the Normality of the sample sums. In the next section, we will explore the effects of non-Normality on our estimators and their distributions.

the holy hand grenade of Antioch

Why should I invoke the holy CLT in this context? Recall that the CLT speaks to the distribution of sums (and means) of independent random variables. As the sample size increases, the distribution of that sum approaches Normality (Section S.6.4).

Look at the formulas for b_0 and b_1. They both include the *sum* of y_i. Thus, it is the distribution of $\sum y_i$ that we really care about. If the y_i are from a normal distribution, then this requirement is met. **However**, if the sample size is large enough, this condition is *also* met, as long as the data are from a distribution with a finite variance. In other words, the CLT rules the world.

peace and love

4.1.3 Exploration of the Effects of Non-Normality

Mathematically speaking, if the Normality assumption does not hold, then nothing in Chapter 3 is absolutely true for the model. However, the Central Limit Theorem tells us that a large sample size mitigates the effects of non-Normality in the residuals.

CLT

Let us explore that here...

An Appropriate Test?: Recall from your previous statistics course two main types of errors: Type I and Type II. A Type I Error happens when the null hypothesis is correct, but the test tells you to reject it. For a statistical test to be appropriate, the first requirement is that the Type I Error rate is equal (or sufficiently close to) the claimed α level.

Thus, to determine if OLS is an appropriate test, let us examine the Type I Error rate to check that it is (close enough to) α. This means we generate our data from the null hypothesis *while* meeting the requirements.

Here is the script to generate the data once

```
set.seed(30)
beta0 = 3
beta1 = 0

x = 1:20
e = rnorm(20, m=0, s=1)

y = beta0 + beta1*x + e
```

This tests the null hypothesis that $\beta_1 = 0$:

```
model = lm(y~ x)
summary(model)
```

The p-value corresponding to the test of our hypothesis is 0.3784. We can have R just give that number to us:

```
summary(model)[[4]][2,4]
```

That is one p-value. If the test is entirely appropriate, we would reject our true null hypothesis about α of the time. This statement is equivalent to the statement that the p-values follow a standard Uniform distribution.

$$P \sim \mathcal{U}(0,1) \tag{4.2}$$

Why?

If the test is appropriate, and if we reject when the p-value is less than α, what is the probability of rejecting? It should be α. Why?

appropriate

In statistical symbols, this is

$$\mathbb{P}\left[P \leq \alpha\right] = \alpha \tag{4.3}$$

This is just the cumulative distribution function for the standard Uniform distribution. Thus, $P \sim \mathcal{U}(0,1)$.

or millions

The code above gave us one p-value. To investigate the distribution of p-values, you need to obtain thousands of p-values. The easiest way to do this is to loop through the above steps and save the p-value from each test.

That is what the following code does:

```
set.seed(30)

pval = numeric()

beta0 = 3
beta1 = 0
x = 1:20

for(i in 1:1e4) {
  e = rnorm(20, m=0, s=1)
  y = beta0 + beta1*x + e
  model = lm(y~ x)
  pval[i] = summary(model)[[4]][2,4]
}
```

At the end of running this code, the variable `pval` contains 10,000 (`1e4` = 1×10^4) observed p-values.

To determine if it follows a standard Uniform distribution, we can create a histogram. When you do so, you note that it appears to closely follow a standard Uniform distribution, although not exactly. We would not expect it to follow the distribution *exactly*; those p-values are based on random samples and are therefore random values and the results will therefore be random.

Understanding the meaning of the p-value (above), we can check if the test is appropriate for $\alpha = 0.05$ by checking that the rejection rate is sufficiently close to 0.05. In other words, we can test if the proportion of p-values less than α is close enough to α. The Binomial test can accomplish this:[3]

```
binom.test( x=sum(pval<0.05), n=length(pval), p=0.05 )
```

The null hypothesis is that the proportion of rejections is equal to 0.05. The p-value of 0.9817 indicates that the test seems reasonable. In other words, the test appears to be find for $\alpha = 0.05$.

[3]The Binomial test is the exact test for checking that the rejection rate is equal to the claimed rate, 0.05. In your introductory course, you may have learned either the proportions test or the Wald test. Both are approprimate tests that rely on the normal approximation to the Binomial distribution.

To determine if the test is appropriate for all possible α-values, we could repeat the above for every possible value of α. That would only take ∞ years. On the other hand, we can test all possible α values at once by recognizing that the p-values should follow a standard Uniform distribution and testing if they do.

We can perform a statistical test to determine if the distribution of the observed p-values is sufficiently non-Uniform:

```
ks.test(pval, "punif")
```

This line performs the Kolmogorov-Smirnov test (Massey 1951). Its null hypothesis is that the observed values follow the distribution stated. Because the p-value is 0.4361, we fail to reject the null hypothesis that the p-values follow a standard Uniform distribution.

In other words: the test appears to be universally appropriate. That is, it seems to be appropriate for *any* value of α you select.

Thus, we know that the test associated with testing the null hypothesis $\beta_1 = 0$ is appropriate for our usual value of α as well as for all values of α.

It is good to know that OLS works when the assumptions are met.

Warning: *When running tests, you will tend to just look for the p-value and draw conclusions based on that one number. If you do, be very clear what that p-value measures. In the previous example, there were 10,002 different p-values. The first 10,000 were p-values for the test that* $\beta_1 = 0$ *for 10,000 different sets of data. The 10,001*th *p-value determined if the distribution of those p-value was standard Uniform. The last p-value determined if the proportion of those p-values less than* $\alpha = 0.05$ *was* 0.05.

There were 10,002 different tests in this example.

Non-Normal Residuals I: Now, what about if the Normality assumption is not met? What if the residuals follow an Exponential distribution? The following code generates 10,000 p-values where the residuals follow an Exponential($\lambda = 1$) distribution. Be able to compare it to the previous code listing and find the *one* difference.

```
set.seed(30)

pval = numeric()

beta0 = 3
beta1 = 0
x = 1:20

for(i in 1:1e4) {
  e = rexp(20, rate=1)
  y = beta0 + beta1*x + e
  model = lm(y~ x)
  pval[i] = summary(model)[[4]][2,4]
}

binom.test( x=sum(pval<0.05), n=length(pval), p=0.05 )

ks.test(pval, "punif")
```

The Binomial test returns a p-value of 0.422, which suggests to us that the OLS test is appropriate for $\alpha = 0.05$. Furthermore, the Kolmogorov-Smirnov test returns a p-value of 0.2867. Thus, even if the residuals are skewed this much ($\gamma_1 = 2$), the tests arising from the ordinary least squares estimation method (Theorem 3.8) appear universally appropriate.[4]

> ***Note***: The sample size in these test is $n = 20$. This is much lower than the usual "rule of thumb" of $n = 30$.

> ***Note***: Also note that what we actually discovered is that OLS is very robust to violations of some of its requirements.

[4]The parameter γ_1 is a measure of skew. It is defined as the third standardized central moment. See Appendix S.6.5.

Non-Normal Residuals II: Now, let's create a different skewed distribution for the residuals that is even more skewed: a Chi-Square($\nu = 1$) distribution ($\gamma_1 = \sqrt{8} \approx 2.8$). Again, be able to compare it to the previous code listing and find the one difference.

```
set.seed(30)

pval = numeric()

beta0 = 3
beta1 = 0
x = 1:20

for(i in 1:1e4) {
  e = rchisq(20, df=1)
  y = beta0 + beta1*x + e
  model = lm(y~ x)
  pval[i] = summary(model)[[4]][2,4]
}
```

The Kolmogorov-Smirnov test returns a p-value of 0.2667. Thus, even if the residuals are this skewed, the tests arising from OLS appear universally appropriate. The Binomial test returns a p-value of 0.0939, which tells us that the OLS test appears to be appropriate for $\alpha = 0.05$.2f@`R` functions!`ks.test`

Thus, even when the residuals follow *this* heavily skewed distribution, the conclusions based on our OLS tests (Theorem 3.8) seem to be appropriate for a sample size of $n = 20$.

Non-Normal Residuals III: Now, let's create a symmetric distribution for the residuals. Because its variance is not finite, the Central Limit Theorem does not apply: It is the Cauchy distribution (Section S.4.7). Again, be able to compare it to the previous code listing and find the one difference.

```
set.seed(30)

pval = numeric()

beta0 = 3
beta1 = 0
x = 1:20

for(i in 1:1e4) {
  e = rcauchy(20)
  y = beta0 + beta1*x + e
```

```
  model = lm(y~ x)
  pval[i] = summary(model)[[4]][2,4]
}
```

The Kolmogorov-Smirnov test returns a p-value of essentially 0. The Binomial test also returns a p-value of essentially 0. This tells us that the OLS test is not appropriate everywhere or even at $\alpha = 0.05$ when the residuals come from a Cauchy distribution.

Increasing the sample size will not fix this issue, either:

```
set.seed(30)

pval = numeric()

beta0 = 3
beta1 = 0
x = 1:5000

for(i in 1:1e4) {
  e = rcauchy(5000)
  y = beta0 + beta1*x + e
  model = lm(y~ x)
  pval[i] = summary(model)[[4]][2,4]
}

hist(pval)
```

In this code, the sample size is 5000. The conclusions are the same.

rejection rate

Looking at the histogram, you see that the first bar is much smaller than the others. This means the OLS tests reject at a lower rate than 0.05.

Note: When the underlying distribution does not have a finite variance, such as the Cauchy distribution, the Central Limit Theorem does not apply. That means increasing the sample size has absolutely no effect on the normality of the sums. The sample sums are *never* normally distributed.

4.2: Constant Expected Value

A second requirement of ordinary least squares (OLS) is that the expected value of the residuals is constant (and zero). From the gut, this means that the residuals evenly bounce above and below our estimates (regression curve). If the residuals are above (or below) our estimates more than expected, then the curve should be moved up (or down) to provide a "better" fit.

We used this requirement in many places in Chapters 2 and 3. This requirement is entirely equivalent to the assumption that the underlying model (expected/predicted values) consistently fits the data, that there is no systematic error. **equivalent**

> ***Note***: Be aware, however, that the mathematics behind OLS will force the average residual to be zero. This means two things. First, the OLS model is "self-correcting," in that the "line of best fit" will provide the best linear fit. Second, the OLS model ensure that it is impossible to detect a systematic error in the measurements.

4.2.1 Graphical Test

Graph the residuals against each of the independent variables. Look for non-linear patterns in the plot (parabolas, cubics, etc.). If such exists, your model is misspecified. A fix is to transform the independent variable to eliminate that pattern. This is one place where the graphical "test" is superior to the numeric test. If you can identify the pattern, you have the fix.

For instance, if the residuals have the pattern in Figure 4.4, then the solution may be to use x^2 in place of (or in addition to) x. To see this, run the following code to obtain Figure 4.4. **residuals plot**

```
set.seed(370)
x = seq(0,3,length=20)
n = length(x)
e = rnorm(n)
y = 4 + 2*x^2 + e

mod = lm(y ~  x)
E = residuals(mod)

plot(x, E)                ## residuals plot
```

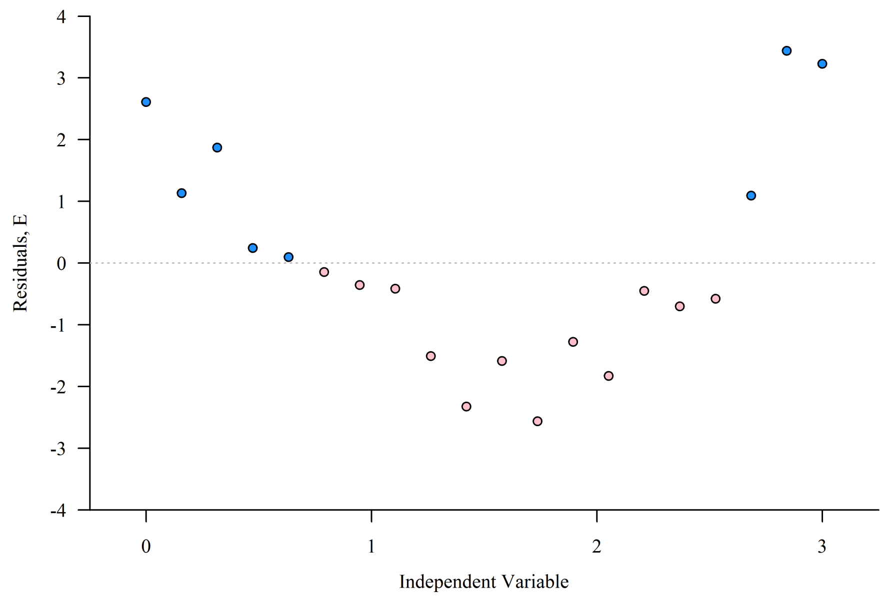

Figure 4.4: *A residuals plot for a misspecified model. Note that the residuals show a definite quadratic form to them. Fixing this issue may be as simple as including x^2 as an additional independent variable.*

Note the strong quadratic shape to the residuals plot (Figure 4.4). This strongly suggests that the model is misspecified.

run

Note: As an aside, note that there are only three 'runs' in the residuals. A run is a sequence of values on one side of the prediction line. According to Figure 4.4, the first run consists of the first 5 values; the second, the next 12 values; and the third, the last 3 values.

Since the number of runs is based on the Binomial distribution, we can calculate the probability of observing this number of runs under the null hypothesis. Thus, we can calculate a p-value for the hypothesis that the model is properly specified (see Section S.6.3).

This example clearly shows that the model is misspecified. There is still some information contained in the residuals. It would be wrong to ignore that information.

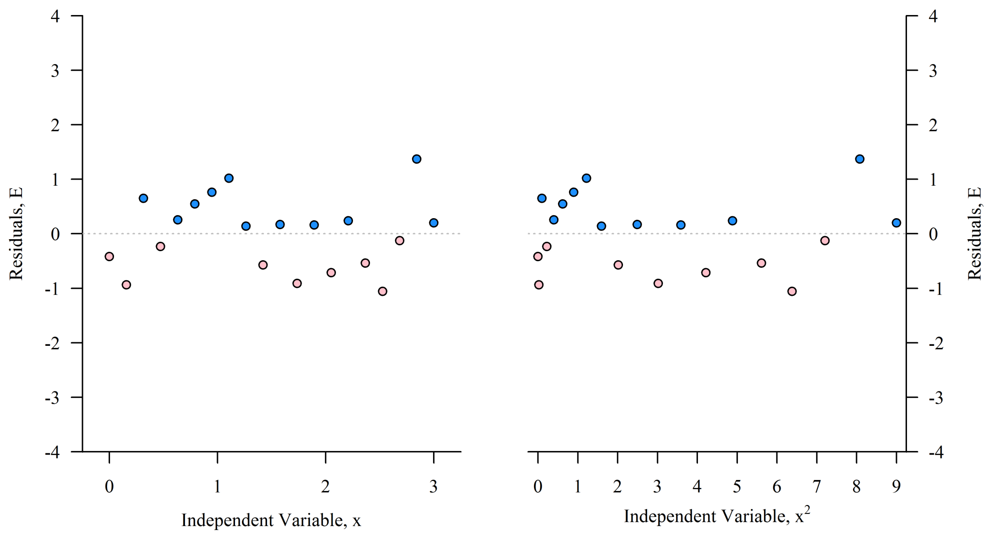

Figure 4.5: *Residuals plots for the properly specified model. There is one residuals plot per independent variable. Note that the residuals in neither plot suggest anything other than a lack of pattern.*

Since the residuals plot has a prominent quadratic shape, a solution is to include x^2 in the model:

```
x2 = x^2
mod = lm(y ~  x + x2)
E2 = residuals(mod)

plot(x, E2)                ## residuals plot
```

With this change, we re-examine the residuals plot — one residuals plot for each independent variable. Since we have two (one?), we need to examine two (one?) residuals plots (Figure 4.5). Note the transformation was successful. Neither plot shows anything other than random bouncing across the line $y = 0$.

one?

Note: There is a habit to feel sad that some requirement is not met by the model/data, such as above. However, do not feel sad. Feel happy, because you have learned something new about the relationships in the data! We know more! Celebrate!

Happiness!

run

4.2.2 A Numeric Test

That last sentence leads us to a numeric test. Compare the plots of Figure 4.4 and 4.5. The residual is colored blue if it is positive and pink otherwise. In Figure 4.4, there are long unbroken streaks (runs) of blue and pink. In Figure 4.5, the length of those runs is much reduced *and* the number is increased.

The test suggested by the above is not-surprisingly called the runs test (Section S.6.3). It is implemented in the `lawstat` package as the function `runs.test`. It takes just one piece of information: the residuals in the order of the independent variable.

It is also implemented in the `randtests` and `snpar` packages, as well as the `rfs` add-on, which is what I use in this book.

Here, I demonstrate the `runs.test` function in the `rfs` add-on. Note that this function currently requires the `lawstat` package to be installed. This restriction may change in the future.

```
source("http://rfs.kvasaheim.com/rfs.R")
library(lawstat)

set.seed(370)
x = runif(100)
e = rnorm(100)

runs.test(e, order=x)       ## The runs test
```

In this version of the `runs.test` function, the first slot goes to the residuals, and the second slot goes to the independent variable.

The output of this code is

```
        Runs Test - Two sided

data:  e, as ordered by x
Standardized Runs Statistic = 1.6081, p-value = 0.1078
```

As usual, check the p-value. Since the p-value of 0.1078 is greater than the α level of 0.05, we fail to reject the null hypothesis that the expected value of the residuals is constant and zero. Thus, since the p-value is greater than α, the model passes this test.

4.2.3 Exploration of the Effects of Non-Constant Expected Value

To see the effect of a non-constant expected value, let us revisit one of the proofs from Chapter 2.

What is the expected value of b_1 (the slope)?

$$\mathbb{E}[b_1] = \mathbb{E}\left[\frac{\sum_{i=1}^n (x_i - \overline{x})(Y_i - \overline{Y})}{\sum_{i=1}^n (x_i - \overline{x})^2}\right] \tag{4.4}$$

$$= \mathbb{E}\left[\frac{\sum_{i=1}^n (x_i - \overline{x})Y_i}{\sum_{i=1}^n (x_i - \overline{x})^2}\right] \tag{4.5}$$

$$= \frac{\sum_{i=1}^n (x_i - \overline{x})\mathbb{E}[Y_i]}{\sum_{i=1}^n (x_i - \overline{x})^2} \tag{4.6}$$

$$= \frac{\sum_{i=1}^n (x_i - \overline{x})(\beta_0 + \beta_1 x_i + \mathbb{E}[\varepsilon_i])}{\sum_{i=1}^n (x_i - \overline{x})^2} \tag{4.7}$$

$$= \beta_0 \frac{\sum_{i=1}^n (x_i - \overline{x})}{\sum_{i=1}^n (x_i - \overline{x})^2} + \beta_1 \frac{\sum_{i=1}^n (x_i - \overline{x})x_i}{\sum_{i=1}^n (x_i - \overline{x})^2} + \frac{\sum_{i=1}^n (x_i - \overline{x})\mathbb{E}[\varepsilon_i]}{\sum_{i=1}^n (x_i - \overline{x})^2} \tag{4.8}$$

$$= \beta_0 \frac{0}{\sum_{i=1}^n (x_i - \overline{x})^2} + \beta_1 \frac{\sum_{i=1}^n (x_i - \overline{x})(x_i - \overline{x})}{\sum_{i=1}^n (x_i - \overline{x})^2} + \frac{\sum_{i=1}^n (x_i - \overline{x})\mathbb{E}[\varepsilon_i]}{\sum_{i=1}^n (x_i - \overline{x})^2} \tag{4.9}$$

$$= \beta_1 + \frac{\sum_{i=1}^n (x_i - \overline{x})\mathbb{E}[\varepsilon_i]}{\sum_{i=1}^n (x_i - \overline{x})^2} \tag{4.10}$$

Now, this last line is β_1 if ε is independent of x. So, if the expected value is constant zero, the OLS estimator of β_1 is clearly unbiased. I leave it as an exercise to show that if it is constant, but non-zero, then the OLS estimator of β_1 remains unbiased. **exercise**

Think of it this way: If the residuals are also a function of x, then their effect is also captured in the b_1 estimator.

If the assumption of a constant expected residual value is violated, the OLS estimate of β_1 is biased. This is not a good thing. It means that your predictions are wrong… even "on average."

But, what about the OLS estimator of β_0, the y-intercept? What effect does a non-constant expected value have on it? To see, let us revisit the proof of the unbiasedness of b_0.

$$
\begin{aligned}
\mathbb{E}[b_0] &= \mathbb{E}\left[\overline{Y} - \overline{x} b_1\right] && (4.11)\\
&= \mathbb{E}\left[\overline{Y}\right] - \overline{x}\,\mathbb{E}[b_1] && (4.12)\\
&= (\beta_0 + \overline{x}\beta_1 + \mathbb{E}[\varepsilon_i]) - \overline{x}\left(\beta_1 + \frac{\sum_{i=1}^n (x_i - \overline{x})\mathbb{E}[\varepsilon_i]}{\sum_{i=1}^n (x_i - \overline{x})^2}\right) && (4.13)
\end{aligned}
$$

The last term is the expected value of b_1 from above. With that, we have

$$
\mathbb{E}[b_0] = \beta_0 + \mathbb{E}[\varepsilon_i] - \overline{x}\frac{\sum_{i=1}^n (x_i - \overline{x})\mathbb{E}[\varepsilon_i]}{\sum_{i=1}^n (x_i - \overline{x})^2} \tag{4.14}
$$

Thus, there are two places that the non-constant variance affects the OLS estimator of β_0. If $\mathbb{E}[\varepsilon_i] = 0$, the expected value of the errors is zero, then b_0 is unbiased for β_0. If $\mathbb{E}[\varepsilon_i]$ is constant, but non-zero, then $\mathbb{E}[b_0] = \beta_0 + \mathbb{E}[\varepsilon_i]$. If $\mathbb{E}[\varepsilon_i]$ is a function of x, then $\mathbb{E}[b_0]$ is $\beta_0 + \mathbb{E}[\varepsilon_i]$ plus some function of x.

exercise

I leave it as an exercise for you to show that $\mathbb{E}[b_0] = \beta_0 + \mathbb{E}[\varepsilon_i]$ if $\overline{x} = 0$, regardless of whether the residuals are correlated with the independent variable. This is yet another reason some disciplines center their data before analyzing it.

Warning: *The actual assumption is that the expected value of the residual is constantly zero. However, because of the mathematics of OLS, the mean residual is* guaranteed *to be zero (page 41). So, there is no way to test if the expected value of the residuals is constantly* zero, *only that it is constant.*

important?

Of all assumptions/requirements, this is the most important to meet. If your residuals depend on the value of x, then both the b_0 and b_1 estimators are biased. If the expected value of the residuals is not zero, then the b_0 estimator is biased.

It is even worse. Because OLS mathematically forces $\overline{e} = 0$, one cannot test if the expected value of the residuals *really* is zero (page 41). One must rely on the assumption that the data were collected without systematic error. That is, the statistician must trust the scientist to measure things correctly.

Note: When would the residuals be a function of the residuals? This is an excellent question that you need to grapple with. It fundamentally means that you are missing an important variable from your model. Perhaps that variable is not an independent variable. Perhaps it is a confounding variable. Perhaps it is another dependent variable, which requires multivariate regression (beyond the scope of this book). It definitely means your model is fatally weak and should be rejected.

4.3: Constant Variance

The last assumption/requirement we will explore is the assumption that the residuals have a constant variance, σ^2. We used this in many places in Chapter 3. In fact, every place you saw a σ^2, we relied on the assumption it was a constant, that it was not really σ_i^2.

If the variance is not constant with respect to the independent variables, neither the test statistic nor the confidence intervals will be correct.

Note: A note on vocabulary. If the variance of the residuals is constant, we claim the residuals are "homoskedastic." Otherwise, they are "heteroskedastic." Recall that 'homo' means same, 'hetero' means different, and 'skedastic' means scatter.

4.3.1 Graphical Test

The graphical test is very similar to that for checking constant expected value. In that assumption, a residual plot was created. The middle of the vertical spread was traced out and checked to see if it was always near the zero line.

For testing constant variance, a residual plot can be used. The vertical spread of the data is traced out and checked that it does not vary too much. Note that the "vertical spread" is not the range, but the middle portion that contains about two-thirds of the data.

There are three stereotypical shapes that suggest heteroskedasticity. These three shapes, along with a shape suggesting homoskedasticity, are presented in Figure 4.6.

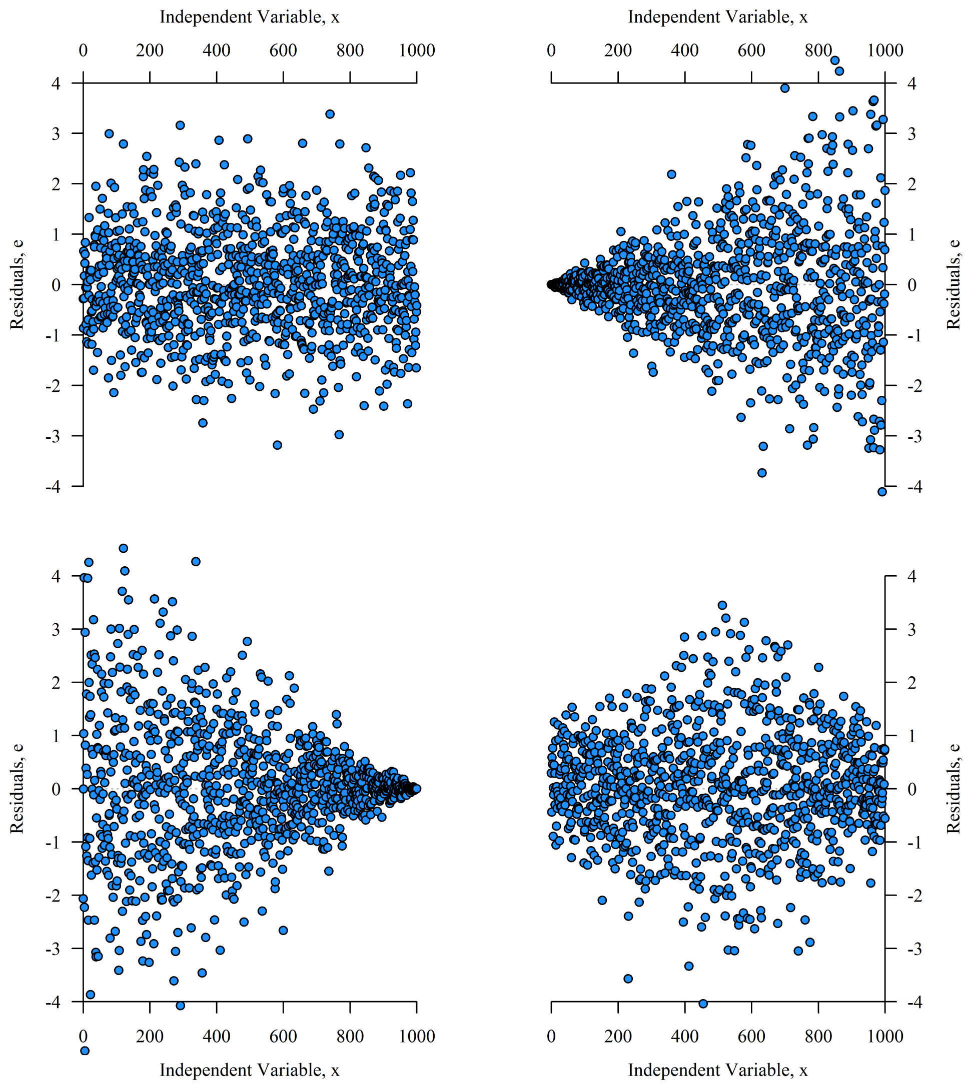

Figure 4.6: *Residual plots illustrating homoskedasticity (top-left) and three typical types of heteroskedasticity: trumpet (top-right), funnel (bottom-left), and bulge (bottom-right).*

4.3.2 Numeric Test

In addition to what your eyes may tell you, it may be better to *also* perform a numeric test of homoskedasticity.[5] There are a couple. The main test used in the regression area is the Breusch-Pagan test.

The way that the Breusch-Pagan test works is to refit the model including higher-order terms of the independent variables and compare the ratio of the two SSE values to a specific distribution. Since the null hypothesis is homoskedasticity (constant variance), a ratio close to one (large p-value) indicates that the model passes this test (the higher-order terms add little to the predictive ability of the model). A sufficiently small p-value indicates the presence of heteroskedasticity (those larger-order terms should be included to the model), which means the model should not be used.

So, to examine the Breusch-Pagan test, let us generate our data, first according to our assumptions, then in violation of them. The Breusch-Pagan test is contained in the `lmtest` package as the function `bptest`. **`bptest`**

```
set.seed(30)
x = seq(0,100)
n = length(x)
e = rnorm(n, m=0, s=1)
y = 3 + 2*x + e

plot(x,y, pch=21, bg="dodgerblue")
```

The scatter plot of this data does not really seem to suggest a changing variation in the residuals. Let us perform the Breusch-Pagan test to see if this numeric test also fails to detect a problem with the constant-variance assumption.

```
mod = lm(y~ x)
bptest(mod)
```

If you get a `Error: could not find function "bptest"` message, then you need to load (or install and load) the `lmtest` package first.

[5]I recommend performing both, when possible. If the p-value on the numeric test is too low, then the graphical test either gives you clues on where the problem is, or that the problem is practically minor and can be ignored.

Once you do, your output should look exactly like this

```
        studentized Breusch-Pagan test

data:  mod
BP = 0.23429, df = 1, p-value = 0.6284
```

Because the p-value is greater than α, we cannot reject the null hypothesis of homoskedasticity. The model passes the test.

Let us look at what happens when the residuals are heteroskedastic.

```
set.seed(30)
x = seq(0,100)
n = length(x)
e = rnorm(n, m=0, s=sqrt(x))
y = 3 + 2*x + e

plot(x,y, pch=21,bg="dodgerblue")
```

The scatter plot of this data definitely suggests increasing variation in the residuals, a trumpet shape that indicates heteroskedasticity. Let us perform the Breusch-Pagan test to see if this numeric test also detects a problem with the constant-variance assumption.

```
mod = lm(y~ x)
bptest(mod)
```

Your output should look like this

```
        studentized Breusch-Pagan test

data:  mod
BP = 8.0992, df = 1, p-value = 0.004428
```

Because the p-value is less than α, we reject the null hypothesis of homoskedasticity. The model does not pass the test.

> ***Note***: Recall from your previous statistics course that a true null hypothesis will be rejected α proportion of the time and a false null hypothesis will be accepted β proportion of the time, where α is the Type I error rate and β is the Type II error rate. Neither of these numbers can be zero without making the other 1. So, be aware that you may be accepting or rejecting incorrectly.

4.3.3 Exploration of the Effects of Non-Constant Variance (coverage)

Instead of looking at proofs or the distribution of p-values, let us generate data and look at the confidence intervals estimated using OLS. While it is easy to see that there *is* an effect using proofs, this method may make it easier to see *how serious* those effects are.

The first thing to do is to generate heteroskedastic data, calculate the 95% confidence intervals, and check if the true population parameters β_0 and β_1 are in the intervals. The second thing is to repeat this step many, many, many times to approximate the probability that the confidence intervals contains the population parameters. If the coverage of the estimated intervals are close to 95%, then the effect of heteroskedasticity is minor. **coverage**

Here is the entire code

```
set.seed(30)
b0covered = numeric()
b1covered = numeric()

for(i in 1:1e4) {
  x = seq(0,10,0.5)
  n = length(x)
  e = rnorm(n, m=0, s=sqrt(x))
  y = 3 + 2*x + e
  mod = lm(y~ x)
  ci = confint(mod)
  b0covered[i] = (3>ci[1,1]) && (3<ci[1,2])
  b1covered[i] = (2>ci[2,1]) && (2<ci[2,2])
}
```

Running this may take anywhere from a few seconds to a minute or more. At the end, you will have two variables, `b0covered` and `b1covered`, that contain 10,000 Boolean values (`TRUE` and `FALSE`). A `TRUE` indicates the population parameter was covered by the confidence interval. A `FALSE` indicates it was not. The proportion of `TRUE` values can quickly be calculated using `mean(b0covered)` and `mean(b1covered)`.

The proportion of times the β_0 parameter was covered by the confidence interval was 0.996. Thus, the estimated coverage for β_0 when the data are heteroskedastic *in this manner* is 99.6%. The coverage rate for β_1 is about 95.1%. Both *should* be close to $1-\alpha$, 95%.

From these results, we know a couple of things. First, this amount **interpretation** of heteroskedasticity did not (practically) significantly affect the confidence intervals for the slope parameter, β_1. Second, it *did* (practically) significantly affect it for the intercept parameter. In fact, because the reported confidence interval is much wider that the "true" confidence interval, a researcher will reject too *in*frequently. This means the power of the test is too low.

Let us try a greater level of heteroskedasticity. And, let us make it funnel instead of the above trumpet:

```
set.seed(30)
b0covered = b1covered = numeric()

for(i in 1:1e4) {
  x = seq(0,10,0.5)
  n = length(x)
  e = rnorm(n, m=0, s=15-x)
  y = 3 + 2*x + e
  mod = lm(y~ x)
  ci = confint(mod)
  b0covered[i] = (3>ci[1,1]) && (3<ci[1,2])
  b1covered[i] = (2>ci[2,1]) && (2<ci[2,2])
}
```

Note what changed: the standard deviation of the residuals. It starts high and gets smaller — a funnel shape.

interpretation

This scheme produces a coverage estimate of 88.8% for the intercept parameter and 93.9% for the slope parameter. Thus, the confidence interval for the intercept is again affected more than that of the slope parameter. In fact, the slope parameter is relatively unchanged.

Let us try once more. This time, let us look at bulge heteroskedasticity.

```
set.seed(30)
b0covered = b1covered = numeric()

for(i in 1:1e4) {
  x = seq(0,10,0.5)
  n = length(x)
  e = rnorm(n, m=0, s=11-2*abs(x-5))
  y = 3 + 2*x + e
  mod = lm(y~ x)
  ci = confint(mod)
  b0covered[i] = (3>ci[1,1]) && (3<ci[1,2])
  b1covered[i] = (2>ci[2,1]) && (2<ci[2,2])
}
```

Again, note the common code and the slight changes. Here, the variance starts low (1), gets higher (11), then decreases again to 1 — a bulge heteroskedasticity.

interpretation The coverage for both parameters are far from 95%. For β_0, the coverage is 99.3%; for β_1, 99.8%. Both indicate that the researcher will reject the hypotheses far too infrequently.

Test Statstics: The test statistics for β_0 and β_1 are calculated using the MSE. That test statistic, if all of the assumptions are true, follows the Student's t distribution with $n - p$ degrees of freedom. However, if the variance is a function of the independent variable, the distribution of the test statistic is also a function of the independent variable. This is a problem, because the "correct" p-values and confidence intervals would also be functions of x.

rejection The analysis of the effect of heteroskdasticity can be repeated for the distribution of test statistics. *Or*, we can realize that we reject the null hypothesis when the confidence interval does not contain our hypothesized value. That is, the analysis for the confidence interval is sufficient for our understanding of the p-value:

1. If the confidence interval is too wide, then the p-values will be too high (reject the null hypothesis too infrequently).

2. If the confidence intervals are too narrow, then the p-values are too low (reject too frequently).

Of the two possibilities, the first (which produces a higher Type I error rate) may be better from the researcher's point of view in some cases: While the researcher would end up rejecting too infrequently, those rejections are more sure because the true p-value is less than the observed p-value. One may prefer this when it worse to commit a Type I error (rejecting a true null hypothesis).

On the other hand, however, the second may be better in certain circumstances. It will have a lower Type II error rate. If it is more important to protect against a Type II error, then this will be the better scenario.

Note: These findings regarding the hypothesis tests about β *strictly* relate only to these particular three types of heteroskedasticity at these three particular levels. If you find your heteroskedasticity is much higher than those used in these examples, you should run a coverage analysis similar to these to see how bad the heteroskedasticity really affects the confidence intervals.

The above analysis examined the effects of heteroskedasticity on the parameter estimates. It did *not* examine its effect on the prediction of y-values. Those effects are *much more pronounced* and are also a function of the variance *at the x-value*.

To see this in one example, let us generate trumpet-shaped heteroskedastic residuals, predict the value of y at three points along the x-axis, and determine how frequently those points are covered by the calculated confidence intervals.

Before we do this, let us see if we can determine what to expect. For low values of x, there is very little true variation in the predicted value. Thus, we would expect the predicted value to fall in the calculated confidence interval very frequently.

For middling values of x, the true and estimated variances are close to each other. Thus, we would expect coverage to be close to the nominal 95%.

For large values of x, the true variation is much higher than the estimated variation. Thus, a lot of the predicted y-values should fall outside the confidence interval. We should expect the coverage to be relatively low. Let's see if these expectations are met by reality.

coverage

Here is the entire code

```
set.seed(30)

y1covered = numeric()
y2covered = numeric()
y3covered = numeric()

for(i in 1:1e4) {
  x = seq(0,10,0.5)
  n = length(x)
  e = rnorm(n, m=0, s=sqrt(x))
  y = 3 + 2*x + e
  mod = lm(y~ x)

  ypr = predict(mod, newdata=data.frame(x=c(0,5,10)), interval=
    "confidence")

  y1covered[i] = ( 3>ypr[1,2]) && ( 3<ypr[1,3])
  y2covered[i] = (13>ypr[2,2]) && (13<ypr[2,3])
  y3covered[i] = (23>ypr[3,2]) && (23<ypr[3,3])
}

mean(y1covered)
mean(y2covered)
mean(y3covered)
```

The coverage for y when x is very low is 99.6%, which is very high. The coverage for y when x is in the middle is 94.5%, which is about what we want. The coverage for y when x is very high is 88.7%, which is relatively low. **interpretation**

These results are what we expected.

Note: Realize again the connection between p-values and confidence intervals. To ensure that our p-values are "protected" — are no more than estimated — we need to limit ourselves to the areas where the true spread is no larger than the average.

4.4: Multicollinearity

Recall that there was just one requirement of the mathematics on ordinary least squares estimation: The independent variables could not be linear combinations of each other (Section 2.2.1). If this is the case, then we cannot do the mathematics behind OLS estimation.

supercollinearity

However, such perfect multicollinearity is not the only problem that can occur. If two (or more) independent variables are highly correlated, then problems can arise. These problems can be examined in terms of either the computer science or the experimental logic.

4.4.1 The CS of Multicollinearity

Note that we are using a computer to perform our calculations. Because a computer is not able to store numbers in memory perfectly, rounding errors creep into the calculations. While this will always happen when the number does not have a finite binary representation, it is most important to understand when the decimal is close to zero, when it is less than the "machine epsilon." When this is the case, the number rounds to zero. In other words, if the value is between $-\varepsilon$ and $+\varepsilon$, the computer treats it as a zero.

On 64-bit computers, the value of epsilon is approximately 2.220446×10^{-16}. However, if the determinant of the matrix is this value or less in magnitude, the computer will claim it is singular (Appendix page 428).

```
solve( matrix( c(1,0, 0,2.220446e-16), ncol=2) )
```

Mathematically, we can calculate the determinant as 2.220446×10^{-16}, which means the matrix is *not* singular. However, calculating the inverse returns the following:

```
Error in solve.default(matrix(c(1, 0, 0, 2.220446e-17), ncol
     = 2)) :
  system is computationally singular: reciprocal condition
    number = 2.22045e-17
```

The lesson to take beyond this specific case is that the matrix does not have to be singular for the computer to tell you it is. All that is needed is that the determinant of the $\mathbf{X}'\mathbf{X}$ matrix is close to zero.

This is the Computer Science result. The next section looks at what multicollinearity means in terms of the logic of experimental design and interpretation.

4.4.2 The Logic of Multicollinearity

The previous section examined the effects of multicollinearity on calculations, specifically of the inverse of the $\mathbf{X}'\mathbf{X}$ matrix. If its determinant is sufficiently close to zero, then the computer will treat is as a zero, meaning the matrix will be effectively singular. However, this is only a CS problem. A good statistician will pay attention to the conditions that cause multicollinearity… even minor amounts of it.

Recall that multicollinearity occurs when a column in the data matrix is a linear combination of the other columns; that is, it happens when one variable is a linear combination of the others; that is, it happens when one variable adds no new information beyond what the other variables contain. For instance, if one variable is a person's height in inches and another variable is a person's height in centimeters, then multicollinearity exists. The first variable offer no information that is not contained in the second.

A statistician cares about the independent variables in the model. They are designed to explain the dependent variable. Each independent variable is supposed to be independent of the others, because each is designed to explain a different aspect of the response variable. If two explanatory variables are highly correlated with each other, then it will be logically impossible to determine which of the two is causing the change in the dependent variable:

- Is it the logarithm of a person's height in inches or the logarithm of the square of a person's height in inches that can be used to estimate weight?
- Is it average daily temperature or ice cream consumption that can be used to estimate the violent crime rate?
- Is it educational attainment or parental income that can be used to estimate a person's future income?

These three exemplify the issue with multicollinearity in practice. The first example produces mathematical ("super-") multicollinearity because the logarithm of the square of a variable is exactly twice the logarithm of the variable. The first column is twice the second.

The second example does not exemplify mathematical multicollinearity. There is no function of average daily temperature that gives the ice cream consumption. However, there is a very strong linear relationship between the two. Because of this, one cannot statisticall tell if it is the temperature or the ice cream that is affecting violent crime. With this said, unless the dairy farmers are attacking the very foundation of society, the substantive scientific theory suggests that the temperature is likely the factor affecting the crime rate, not the cold dairy.

The third example is more subtle. There is also a strong relationship between a person's education attainment and the parent's income (at least in the United States). Because of this, we are unable to statistically determine if it is the person's educational attainment or the income of the person's parents that affects the person's future income. Social science theories suggest each. The statistics with each explanatory variable also suggest each. What can we do in this case?

Indications of Multicollinearity: To see some statistical indications of multicollinearity, try the following code.

```
set.seed(30)

b0 = 3
b1 = 2
b2 = 3

x1 = seq(0,10,length=8)
x2 = c(1,2,3,4,6,7,8,9)
e  = rnorm(8)

y = b0 + b1*x1 + b2*x2 + e

mod1 = lm(y~ x1)
mod2 = lm(y~ x2)
modA = lm(y~ x1+x2)
```

Clearly, from how this experiment is set up, we know the following:

- There is a strong relationship between $x1$ and y.
- There is a strong relationship between $x2$ and y.
- There is a strong relationship between $x1$ and $x2$.

Running `summary(mod1)` shows us that the first statement is true, *if we ignore the effect* of $x2$. Similarly, running `summary(mod2)` shows us that the second statement is true, *if we ignore the effect* of $x1$. Combining the two explanatory variables in `modA` is confusing if we do not think about multicollinearity. `summary(modA)` gives the following results:

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    2.814      1.898   1.483    0.198
x1             2.145      1.681   1.276    0.258
x2             2.861      2.009   1.424    0.214

Residual standard error: 1.386 on 5 degrees of freedom
Multiple R-squared:  0.9946,    Adjusted R-squared:  0.9924
F-statistic: 458.7 on 2 and 5 DF,  p-value: 2.164e-06
```

Note that this model shows that *neither* independent variable is valuable in modeling the dependent variable. Since a scientist will usually put all the explanatory variables in the model, this is a lesson for us to pay attention to the relationships among the independent variables. By the way, the correlation between the two independent variables is $\rho = 0.9960238$, which is incredibly high.

A Test of Multicollinearity: How do we statistically detect this type of multicollinearity? A simple correlation test will not suffice if we have more than two independent variables because correlation is between only two variables.

The answer comes from the cause of the multicollinearity: If there is multicollinearity, then one independent variable should be linearly related to the others. A linear regression will be able to detect this. *Technically*, a linear regression for each independent variable will detect this. To make this process easy, there is the `vif` function in the `car` package. This function calculates the "variance inflation factor" for each independent variable. The variance inflation factor for independent variable i is defined as

$$\mathrm{VIF}_i := \frac{1}{1-R_i^2} \tag{4.15}$$

Here, R_i^2 is the R-squared value for the model regressing the independent variables on independent variable i.

In our example above, one can calculate the VIF by hand:

```
summary( lm(x1~ x2) )
1/(1-0.9921)
```

The higher the value of R^2, the more independent variable i can be explained by the other independent variables. In other words, the higher the VIF, the less new information that variable adds to the model.

As in much of the field, this description leads to the question **How high is too high?** The "rule of thumb" depends on the discipline. Typical cut-offs are 5, 8, and 10. If the VIF for any of the variables is greater than the cut-off, then there is "too much multicollinearity in the the model."

Fixing Multicollinearity: So, let's say that you have detected multicollinearity in your model. What can you do?

The presence of multicollinearity means that one of your independent variables is highly correlated with a linear function of the others. It adds little to the understanding of the response variable. However, is it variable i that should be examined or the others?

From a statistical standpoint, the model is not helpful. Multiple variables are trying to explain the same aspects of the dependent variable. In other words,

- Is it educational attainment or parent's income that affects the respondent's income?
- Is it race or poverty that affects violent crime?
- Is it intelligence or birth position (oldest, youngest, middle, etc. child) that affects success?
- Is it the ranch or the cattle feed that affects the weight?
- Is it race or income or religion or parental income orhome state that affects voting behavior?
- Is it nordic ancestry or blood type or neanderthal genes that affect the severity of CoViD-19?

In each of these, the explanatory variables are highly correlated and have been used to model the response variable. Because of the correlations, con-

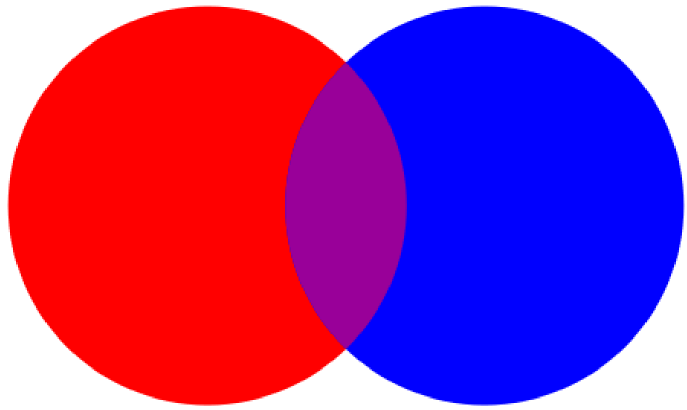

Figure 4.7: *Diagram to illustrate multicollinearity. The left circle is the effect of the first independent variable on the dependent variable; the right, the second. The purple overlap is the similarity of the two variables.*

clusions about what *really* affects the dependent variable are unclear. Statistically, the answer is "yes, each does."

However, since each of the independent variables above are correlated, their effects overlap. This is represented as the purple overlapping area in Figure 4.7. While each variable has an effect on the dependent variable (red and blue), that effect is also split with the other variable (or variables). As such, the key is trying to separate the three sections to determine whether it is the red, the blue, or the purple that is affecting the response variable.

Unfortunately, this is beyond the scope of this course. For those who are interested, you may want to investigate factor analysis (FA) and principal component analysis (PCA). These are two methods for dealing with that overlap (purple area). The first focuses on estimating the purple area; the second, on creating two other variables that combine the two independent variables into their independent components (the parts that are purely red and blue). The advantage is that the independent variables become independent (VIF = 0). The disadvantage is that the newly-created independent variables are only related to the original explanatory variables; thus, interpretation is made more complicated.

4.5: Conclusion

In this chapter, we looked at violating the primary assumptions of ordinary least squares, as well as the effects of multicollinearity. Once we violated those requirements, we looked to see what effect that violation had on our parameters of interest.

We saw that violations of the Normality assumption cause very minor changes to our test statistics, unless the residuals were generated from a distribution with non-finite variance. In such a case, the tests were all but worthless.

Violations of constant expected value (proper model fit) were bad. They cause the OLS estimators, test statistics, *and* confidence intervals to be biased. The lesson here is to make sure that your measuring stick is properly calibrated.

Heteroskedasticity is not an issue for the estimators. They remain unbiased. They are an issue, however, for any testing done. This includes both the test statistic and the confidence interval. This is because the standard error used is the average, not the actual value for a general point.

Multicollinearity arises from independent variables that measure highly similar concepts. Thus, the statistical effect is that the standard errors are inflated. The logical effect is that we are unsure which of the two variables actually affects the dependent variable.

Warning: *Again, be aware of the multiple comparisons issue discussed in Appendix S.6.2. It explains why you need to either adjust your p-value or your alpha level when performing multiple tests, such as when you are testing* both $\beta_0 = 4$ and $\beta_1 = 0$.

4.6: End-of-Chapter Materials

4.6.1 R Functions

In this chapter, we were introduced to many, many, many R functions that will be useful in regression. In fact, this chapter uses more R functions than any other chapter in this book. Here are the many.

Packages:

car This package provides several statistical tests used in the book "An R Companion to Applied Regression" by J. Fox and S. Weisberg. It is a great package that provides a lot of additional functionality for R.

lawstat This package provides several statistical tests used in law and public policy analysis. It provides the `runs.test` function for us.

lmtest This package provides many tests related to linear models. It provides an implementation of the Breusch-Pagan test, `bptest`, which tests for heteroskedasticity in the residuals.

RFS This package does not yet exist. It is a package that adds much general functionality to R. In lieu of using `library(RFS)` to access these functions, run the following line in R:
`source("http://rfs.kvasaheim.com/rfs.R")`

Statistics:

source(filename) This function runs an R script from a separate file. That file may be local or on the Internet.

runs.test(E, order) This alteration to the `lawstat` function tests whether the variable `E`, as ordered by `order` exhibits fit issues.

shapiroTest(E) This tests the null hypothesis that the variable `E` comes from a Normal distribution. It is based on the `shapiro.test` function in the basic R installation. It adds capabilities to test Normality in several groups.

lm(formula) This is the function that performs ordinary least squares estimation on linear models.

bptest(mod) This function from the `lmtest` package performs the Bresuch-Pagan test for heteroskedasticity.

confint(mod) This calculates confidence intervals for the parameters in ordinary least squares regression.

mean(x) This calculates the mean of a sample.

summary(x) This produces the six-number summary or a frequency table of the provided variable, depending on the type of variable.

summary.lm(mod) When applied to a linear model fit using either the `aov` function or the `lm` function, provides estimates of the effects of the numeric variables and the levels of the categorical variables in the model.

summary.aov(mod) When applied to a linear model fit using either the `aov` function or the `lm` function, provides estimates of the statistical significance of the variables in the model.

predict(mod) This predicts the values of the dependent variable at each point in the dataset *or* for the values specified.

fligner.test(formula) This tests for heteroskedasticity when the independent variable is categorical.

aov(formula) This function performs ordinary least squares estimation on linear models.

vif(model) This function calculates the variance inflation factor (VIF) for each of the independent variables in the model.

set.base(var,level) This `RFS` package function redefines the base category in the provided level. By default, the base category is the first according to the alphabet.

PROBABILITY:

set.seed(x) This sets the random number seed.

rexp(n, rate) This generates n random values from an Exponential distribution with the specified rate parameter.

rnorm(n, mean, sd) This generates *n* random values from a Normal distribution with specified mean and standard deviation. By default the mean is 0 and the standard deviation is 1.

runif(n, min, max) This generates *n* random values from a Uniform distribution with specified minimum and maximum values. By default, the minimum is 0 and the maximum is 1.

Mathematics:

head(x) This returns the first six values in the variable.

foot(x) This returns the last six values in the variable.

seq(from, to, by, length) This returns a vector of sequential values, where `by` indicates the step size and `length` specifies the vector length.

length(x) This calculates the length of a vector (variable), which is the sample size, *n*.

residuals(mod) This calculates the residuals in the model, which is the difference between the observed and the predicted.

Graphics:

qqnorm(x) This creates a Normal quantile-quantile plot for the given values.

qqline(x) This adds the diagonal line to the quantile-quantile plot.

overlay(x) This, from the `RFS` package, produces a histogram with a Normal curve overlaying it.

par(. . .) This sets parameters on the next graphic started. Look through the help page for this function to see all you can specify.

plot(x,y) This produces a scatter plot of the y-values against the x-values.

axis(side) When a plot is already drawn, this adds values along axis number `side`.

title(. . .) When a plot is already drawn, this adds the x- and y-labels.

lines(x,y) When a plot is already drawn, this draws lines between each subsequent (x, y) pair.

points(x,y) When a plot is already drawn, this draws points at each (x, y) pair.

Programming:

attach(dataframe) This allows you to access the variables in the `datarame` without having to prefix each with `dataframe$`.

library(package) This loads an external package that you have already installed on your computer. It allows access to all functions and data sets in the `package` package.

as.character(x) This changes the values in variable `x` to be characters.

as.numeric(x) This changes the values in variable `x` to be numbers.

4.6.2 Exercises

1. Show that if the expected value of the residuals is constant, but non-zero, then the OLS estimator of β_1 remains unbiased.

2. Show that $\mathbb{E}[b_0] = \beta_0 + \mathbb{E}[\varepsilon]$ if $\overline{x} = 0$, regardless of whether the residuals are correlated with the independent variable.

4.6.3 Theory Readings

- George E. P. Box. (1976) "Science and Statistics," *Journal of the American Statistical Association*, 71(): 791–799. doi:10.1080/01621459.1976.10480949.

- James V. Bradley. (1968) *Distribution-Free Statistical Tests*. New York: Prentice-Hall.

- John Fox and Harvey Sanford Weisberg. (2010) *An R Companion to Applied Regression*, second edition. Thousand Oaks, CA: SAGE Publications.

- Frank J. Massey, Jr. (1951) "The Kolmogorov-Smirnov Test for Goodness of Fit." *Journal of the American Statistical Society*. 46(**253**): 68–78.

- Abraham Wald and Jack Wolfowitz. (1940) "On a test whether two samples are from the same population." *The Annals of Mathematical Statistics* **11**: 147–162.