Some answers for the lab on simple linear regression

Part 1

Sourcing this code once

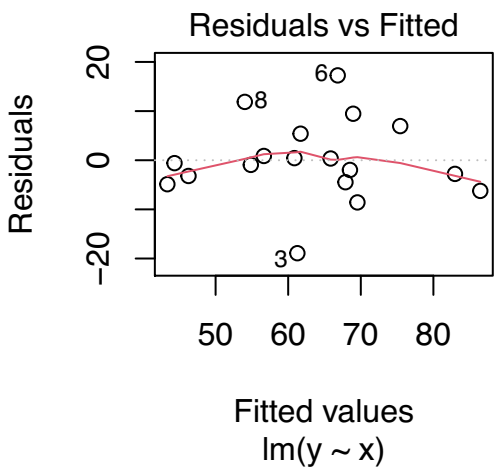
```
sigma <- 8
sampleSize <- 18
beta0 <- -17
beta1 <- 2.5

xdata <- rnorm(sampleSize, 30, 6)
ydata <- beta1 * xdata + beta0 + rnorm(sampleSize, 0, sigma)
myLabData <- data.frame( x = xdata, y = ydata )
lmResult <- lm(y ~ x, data=myLabData)
```

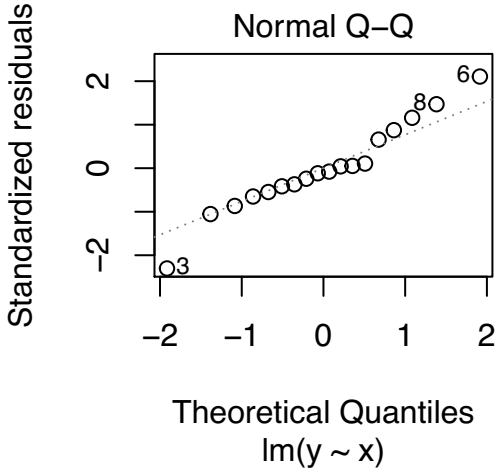
will build data that has been sampled directly from a population in which the Simple Linear Model assumptions are met. At that point, the command

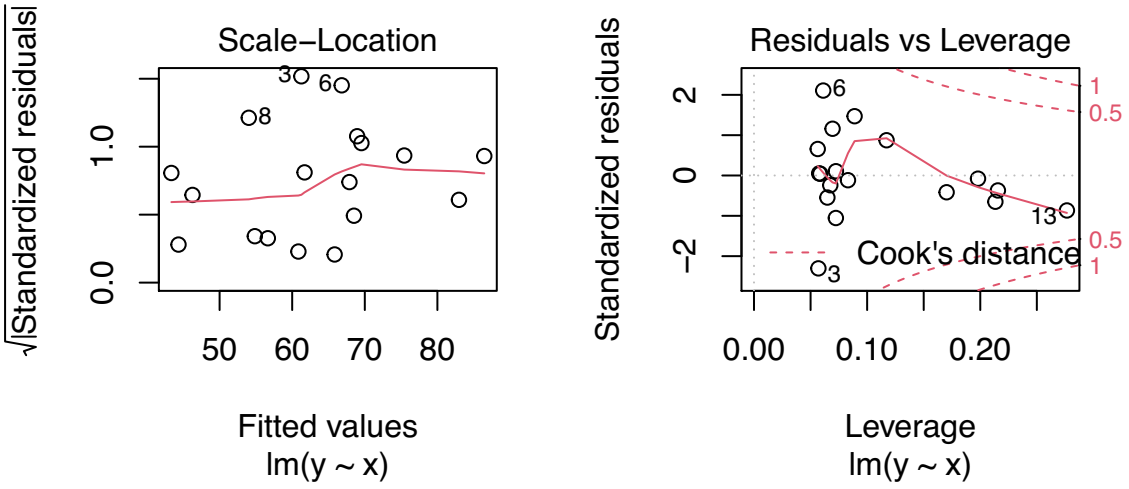
```
plot(lmResult)
```

allows the user to step through four plots. For Number 1, a typical run may result in plots that look like these:

```
plot(lmResult, which=1)
plot(lmResult, which=2)
plot(lmResult, which=3)
plot(lmResult, which=5)
```







Students are not required to include these plots in their write-ups, simply to make observations. And since those observations are highly individualized, there isn't a *right* answer for Problems 1–5. The desirable outcome is to develop intuition about how these plots can suggest abnormalities even when the data is ideal.

1–3. I think what the plots I've included above show is fairly typical.

(a) It seems like there are (nearly?) always numbered points in Plots 1, 2 (top right) and 3, so it's quite likely the portion they give for part (a) of Numbers 1–4 is five-out-of-five. Don't be a hawk, but at the same time, a student who is consistently answering 3-out-of-five or less in part (a) may be faking it, or doing something wrong.

(b) The red lines above are not so straight, nor completely horizontal. So one has to be trained not to jump too quickly and conclude a shape or upward/downward pattern exists. A student might describe the red "lines" in Graphs 1 and 3 above as not pattern-free, but hopefully they come to realize model-adhering data does often put false shapes in front of your eyes.

(c) The quantile-quantile plot above (Plot 2) isn't perfectly line-hugging either. What might/should be observed is that the "false shapes" are more prevalent from this kind of data with smaller sample sizes (so will probably be worst for Problem 2, and best for Problem 3).

(d) In my Graph 4 above, no point is outside the "fences" (red dashed lines) marked as 0.5 or 1. I don't think anyone will see a point outside either boundary more than once (twice?) in five runs, and then only in the case of Problem 2, with the smallest sample size.

4. They must change the code, but are possibly not going to show you their altered lines. I do not really expect answers to parts (a)–(d) to be markedly different than for Number 1. Mostly this is to give them a scenario closely matched to the sort of data they work with in Number 5.

5. (a) It seems there continue to consistently be a few data points flagged in Plots 1–3.

(b) It may be dismissed by some as another false shape, but theere may be something to the "U"-shaped pattern in Plot 1. The red path in Problem 3 is perhaps less horizontal (or less patternless) than in the previous runs with ideal data.

(c) I'm not sure there is anything to note in the quantile-quantile plot (Plot 2). It may be a bit more wiggly than viewed in Numbers 1–4.

(d) Again, no data point appears outside the 0.5 or 1 "fences".

6. (a) The relevant code here is

```
hdAndWine <- read.csv("http://scofield.site/teaching/data/csv/heartDiseaseDeathsAndWine.csv")
lmresult <- lm(formula = hddeaths ~ winealc, data = hdAndWine)
hdDeathPredictor <- makeFun(lmresult)    # requires mosaic package
hdDeathPredictor(winealc=5, interval="confidence", level=.92)
```

```
       fit      lwr      upr
1 145.7195 124.9188 166.5203
```

(b)

```
filter(hdAndWine, country=="Canada")
```

```
  country winealc hddeaths
1  Canada     2.4      191
```

```
hdDeathPredictor(winealc=2.4, interval="prediction", level=.96)
```

```
       fit     lwr      upr
1 205.4383 118.872 292.0047
```

Part 2

1. Running the code

```
set.seed(1200)
myLabData <- tibble(
  x = rnorm(18, 30, 6),
  y = 2.5 * x - 17 + rnorm(18, 0, 8)
)
modelResult <- lm(y ~ x, data=myLabData)
summary(modelResult)

Call:
lm(formula = y ~ x, data = myLabData)

Residuals:
     Min       1Q   Median       3Q      Max
-18.9818  -7.4008  -0.3815   6.9702  20.0011

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -22.7654    18.8219  -1.210 0.244032
x             2.6986     0.5943   4.541 0.000334 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 10.32 on 16 degrees of freedom
Multiple R-squared:  0.563,Adjusted R-squared:  0.5357
F-statistic: 20.62 on 1 and 16 DF,  p-value: 0.0003343

anova(modelResult)

Analysis of Variance Table

Response: y
          Df Sum Sq Mean Sq F value    Pr(>F)
x          1 2197.3 2197.27  20.616 0.0003343 ***
Residuals 16 1705.2  106.58
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```


2. The first row of `manyRuns` and the 95% confidence interval are

```
manyRuns <- do(5000) * sampleAndRegress()
manyRuns[1, 1:8]

         b0       b1  SSModel SSResiduals  SSTotal sigmaHat    SE.b0     SE.b1
1 -27.53055 2.927562 6178.945    1887.978 8066.923 10.86272 11.51446 0.4045642

cInt <- manyRuns$b1[1] + c(-1,1)*qt(0.975, df=16)*manyRuns$SE.b1[1]
```

```
cInt

[1] 2.069924 3.785199

cInt[1] <= 2.5 & 2.5 <= cInt[2]

[1] TRUE
```

And the last line evaluated `TRUE`, indicating $\beta_1 = 2.5$ is inside the confidence interval.

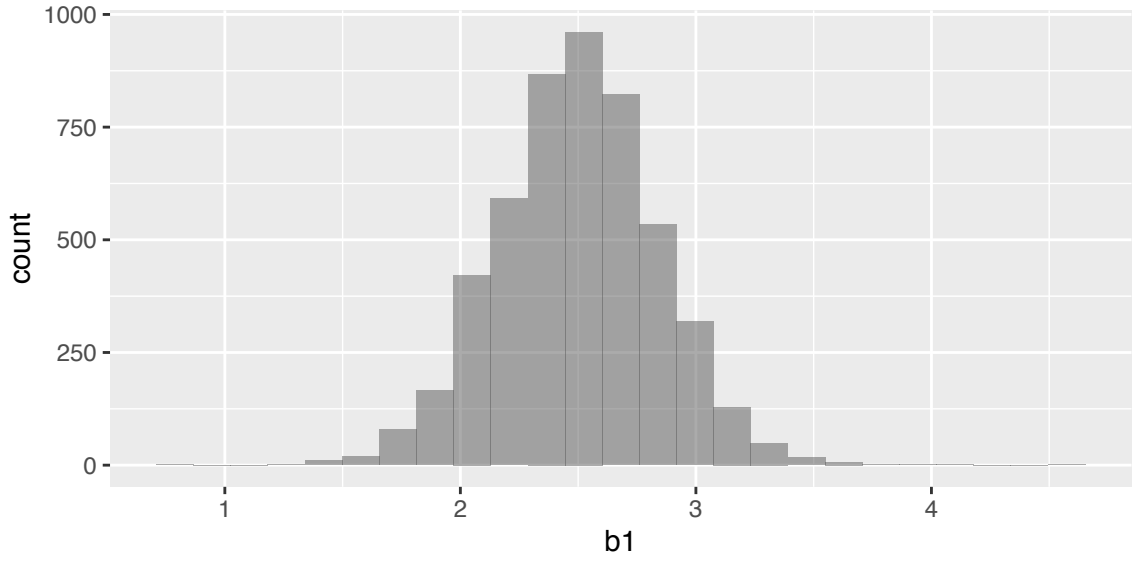
3. The effective coverage rate is

```
nrow(filter(manyRuns, b1-qt(.95,16)*SE.b1 <= 2.5 & 2.5 <= b1+qt(.95,16)*SE.b1)) / 5000

[1] 0.9066
```

which is close to 90%, as expected.

4. We have a approximate **sampling distribution** for b_1, with appearance

```
gf_histogram(~b1, data=manyRuns)
```



The standard deviation of this distribution estimates SE_{b_1}.

```
sd(~b1, data=manyRuns)

[1] 0.3379909
```

5. The smallest and largest estimates stored in in `SE.b1` are

```
min(~SE.b1, data=manyRuns)

[1] 0.1077774

max(~SE.b1, data=manyRuns)

[1] 0.8692807
```

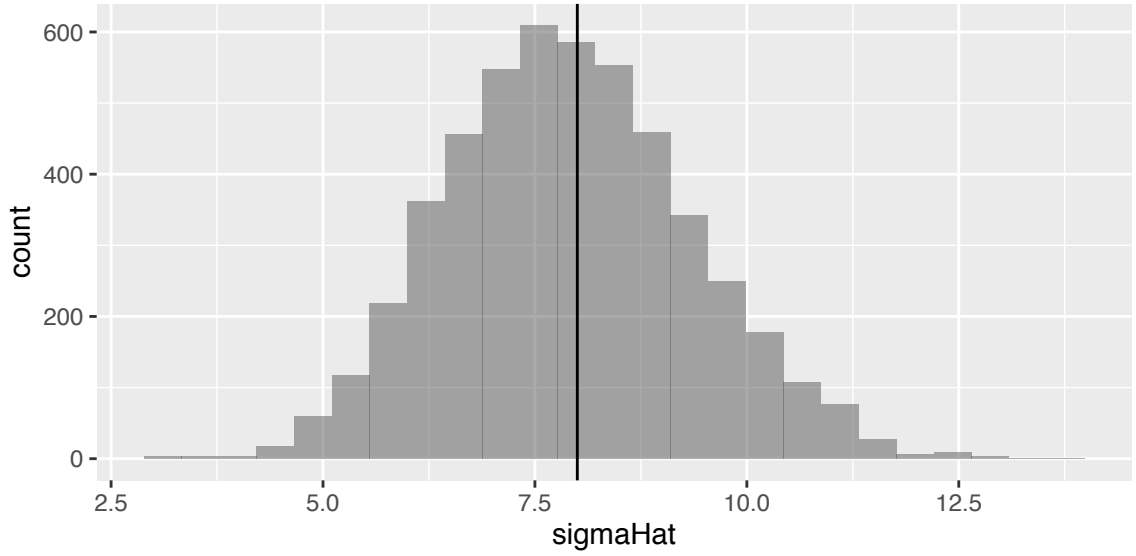
There are many in-between values, making for quite a spread. The margin of error is

proportional to SE_{b_1}, so some of the 50000 90% confidence intervals are far less wide than others. To my mind, that makes it surprising that we wind up with an overall effective coverage rate so close to the advertised value of 90%.

6. Here is a histogram of the sampling distribution of `sigmaHat` values with a vertical line drawn in at the population parameter $\sigma = 8$ it estimates.

```
gf_histogram(~sigmaHat, data=manyRuns) %>% gf_vline(xintercept=8)
```

```
Warning: geom_vline(): Ignoring `mapping` because `xintercept` was provided.
```



It *looks* as if 8 may be at the center (mean) of this not-quite-symmetric distribution. However, when we actually calculate the mean, we get

```
mean(~sigmaHat, data=manyRuns)
```

```
[1] 7.901023
```

which undershoots $\sigma = 8$ enough to conclude `sigmaHat` is a *biased* estimator of σ.