```
Stat 145, Wed 28-Apr-2021 -- Wed 28-Apr-2021
Biostatistics
Spring 2021



-------------------------------
Wednesday, April 28th 2021
-------------------------------
Due::   PS15 due at 11 pm

-------------------------------
Wednesday, April 28th 2021
-------------------------------
Wk 13, We
Topic:: Confidence and prediction intervals

Confidence intervals for beta1 (slope):
 - need
    estimate b1
    estimate of SE.b1
    t-star (critical) value
      take df = n - 1 - #{predictor variables}
      qt(...)
   the first two are reported in summary( lm( ... ) )
 - do a 94% CI from  lm(PctTip ~ Bill, data=RestaurantTips)
 - Students do a 90% CI from  lm(Prise ~ PPM, data=InkjetPrinters)

Introduce script sampleAndRegress()
 - found at  http://scofield.site/teaching/Rscripts/mySLMSampler.R
 - what script does:
    manufactures data for which SLM assumptions hold
    produces various regression-related statistics from that data
 - can be used to produce approximate sampling distributions
     manyRuns <- do(5000) * sampleAndRegress()
   slightly different than bootstrapping, since samples drawn from population
 - some useful(?) R commands
    manyRuns[1,]    displays the outputs of first run/sample
    manyRuns[58,]   displays the outputs of 58th run/sample
    manyRuns$b1     displays b1 from all runs
    manyRuns$b1[58] displays b1 from the 58th run
      manyRuns[58, 2] does this as well
    max(manyRuns$b1)   displays the biggest of the b1 values in 5000 runs
      with(manyRuns, max(b1))   does the same

 - use results to view the many estimated regression lines and true one
    plot(c(5,55),c(-20,110),pch=19,cex=.01)
    abline(manyRuns$b0[1], manyRuns$b1[1])
    abline(manyRuns$b0[2], manyRuns$b1[2])
    abline(manyRuns$b0[3], manyRuns$b1[3])
    abline(manyRuns$b0[4], manyRuns$b1[4])
    for (j in 5:5000) {abline(manyRuns$b0[j], manyRuns$b1[j])}
    abline(-17, 2.5, col="magenta")
```

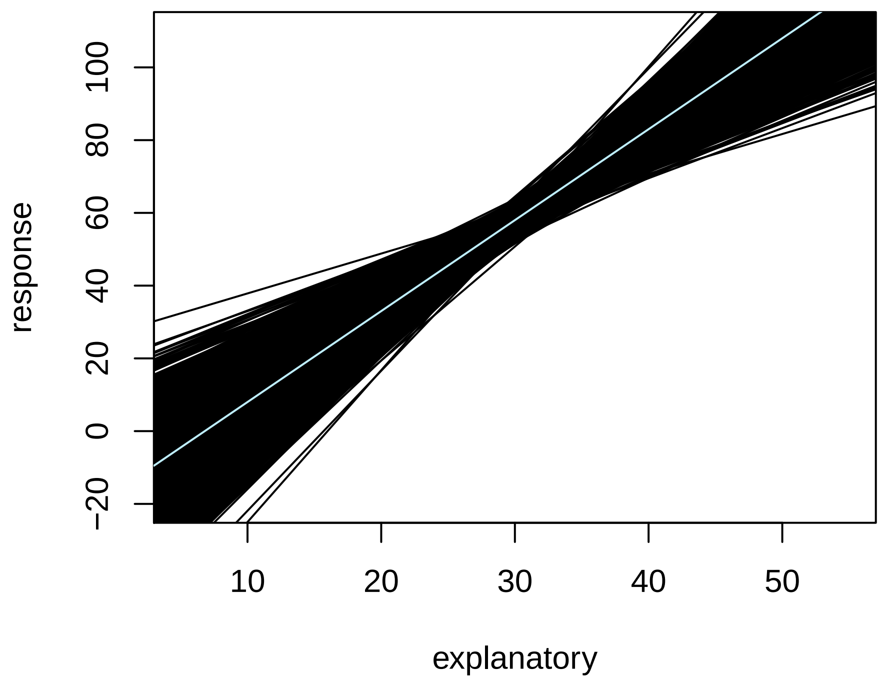
```r
sampleAndRegress <-
  function(
    sigma = 8,
    sampleSize = 18,
    beta0 =  -17,
    beta1 = 2.5
  ) {

    myLabData <- tibble(
      x = rnorm(sampleSize, 30, 6),
      y = beta1 * x + beta0 + rnorm(sampleSize, 0, sigma)
    )

    lmResult <- lm(y ~ x, data = myLabData)
    meanx = mean(myLabData$x)
    meany = mean(myLabData$y)
    ssx = sum((myLabData$x - meanx)^2)
    ssy = sum((myLabData$y - meany)^2)
    ssr = sum(resid(lmResult)^2)
    sigmaEst = sqrt(ssr / (sampleSize - 2))

    data.frame(
      b0 = value(lmResult$coefficients[1]),
      b1 = value(lmResult$coefficients[2]),
      SSModel = sum( (fitted(lmResult) - meany)^2 ),
      SSResiduals = ssr,
      SSTotal = sum( (myLabData$y- meany)^2 ),
      sigmaHat = sqrt(ssr / (sampleSize - 2)),
      SE.b0 = sigmaEst * sqrt(1/sampleSize + meanx^2/ssx),
      SE.b1 = sqrt(sigmaEst^2 / ssx)
    )
  }
```

```r
manyRuns <- do(5000) * sampleAndRegress()
plot(c(5,55),c(-20,110),pch=19,cex=.01, xlab="explanatory", ylab="response")
abline(manyRuns$b0[1], manyRuns$b1[1])
abline(manyRuns$b0[2], manyRuns$b1[2])
abline(manyRuns$b0[3], manyRuns$b1[3])
abline(manyRuns$b0[4], manyRuns$b1[4])
for (j in 5:5000) {abline(manyRuns$b0[j], manyRuns$b1[j])}
abline(-17, 2.5, col="lightblue1")
```


response
−20
0
20
40
60
80
100
10
20
30
40
50
explanatory

Lab on simple linear regression

The lab consists of two parts, and both should be completed by all. The questions begin over again with Number 1 in Part 2 and, indeed, you should treat them like two assignments (i.e., you should produce one .pdf file for each part). Your answers to both sets of questions should be uploaded to Gradescope by 11 pm on Monday, May 3.

If you choose to have it so, your answers to the questions of Part 2 will be included as part of Test 3. Students who choose this option **must** write up their work to Part 2 using R Markdown, producing the .pdf document from a source .Rmd file, including in that write-up all commands and relevant output (within reason—do not include any output that covers a page or more, or output that does does not support any statement made by you), and whatever comments or answers it leads you to assert. (For students not wishing to include answers to Part 2 in their Test 3, use of R Markdown is not required, though it is difficult to see a more convenient way to write up solutions, even for them.) For those **not** choosing this option, the grade on Test 3 will come entirely from the in-class assessment; otherwise, that grade will be a mix, two-thirds from the in-class assessment (which will be the same length for all students) and one-third from Part 2 answers.

Part 1

The following code will be used repeatedly as you work through the questions of Part 1.

```
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 )
```

If you place it in an `R Script` file, then it is easy to run all the lines at once using the "Source" icon. Each time you rerun it, you will have a refreshed data frame `myLabData` with two columns named `x` and `y`, where the coordinate pairs (x, y) represent a sample of size `sampleSize` from a population in which

- the relationship $Y = \beta_0 + \beta_1 X + \epsilon$, with $\epsilon \sim \text{Norm}(0, \sigma)$, is satisfied.
- in the code given, the parameters are $\beta_0 = -17$, $\beta_1 = 2.5$, and $\sigma = 8$. These are easily alterered, but do so only when the problem requires it.

The programming in this R code ensures that, each time you source it, the assumptions of the **simple linear model**, given here as (i)–(iii), are met:

(i) the variables follow a linear association

(ii) residuals are independent

(iii) residuals are normally distributed with mean 0 and common variance σ^2

The goal in Part 1 is for you to get a sense of how the diagnostic plots appear under random sampling when the assumptions of the simple linear model are firmly in place.

1. Run the provided code (i.e., "source" the R script file) five times. After each run, follow that run with these commands (you can even add them to the R script if you like)

   ```
   lmResult <- lm(y ~ x, data=myLabData)
   plot(lmResult, pch=19, cex=.4)
   ```

   allowing you to view the four diagnostic plots provided by R that help in the assessing of model assumptions. (Remember that you are working with data specifically crafted so that the model assumptions hold, which means these plots occur under the most favorable of conditions.) Answer these questions.

   (a) In what portion of the five runs, is *no data point* flagged (i.e., identified by number) in any of the first three plots?

   (b) In what portion of the five runs would you describe the added (red) line in Plots 1 and 3 as truly reflecting "no pattern" (i.e., not much different from a horizontal line)?

   (c) In what portion of the five runs would you describe the dots in the quantile-quantile plot of residuals (Plot 2) as tightly hugging a line (the dashed line that is drawn for you)?

   (d) In what portion of the five runs does at least one dot appear in the top or bottom (usually, but not always, toward the right side) of Plot 4 out beyond the dashed 0.5-curve? Is there ever a point out beind the dashed 1-curve?

2. Change the sample size to 10, and repeat the four parts of Exercise 1.

3. Change the sample size to 40, and repeat the four parts of Exercise 1.

4. Change the values of β_0, β_1 and σ to 260, (-23), and 35, respectively. Repeat the four parts again using sample size 20.

5. Problems 1–4 have been done using synthesized data, chosen specifically to satisfy the model assumptions. The data found at
   `http://scofield.site/teaching/data/csv/heartDiseaseDeathsAndWine.csv`
   is *real* data, so its adherence to the model assumptions is unknown. Using this data (take `winealc` as a predictor of `hddeaths`), view the four plots (view them once—no need for 5 repetitions here). The results should be most directly comparable to the runs of Problem 4. (Why?) Are there any features of the four plots which make this data stand out as different from prior runs—i.e., as perhaps *not* satisfying the model assumptions?

6. (This problem is not in the spirit of the problems preceding it, but as there isn't a lot of practice elsewhere with prediction and confidence intervals, I have included it.) Using tools described in class, generate

(a) a 92% confidence interval for the mean level of `hddeaths` for all countries in which the level of `winealc` is 5.0.

(b) a 96% prediction interval for the level of `hddeaths` in another country whose level of `winealc` matches that of Canada.

Part 2

Obtain a copy of the R script at `http://scofield.site/teaching/Rscripts/mySLMSampler.R`, then upload it from your local computer to your R work space. Open the file, "Source" it, then close the file. Now you have the function `sampleAndRegress()` I used in class available at the console. Try running it:

```
sampleAndRegress()
```

The command `sampleAndRegress()` is not native to R. In following the instructions above, you made it available much in the same way certain add-on commands become available when you load packages like `mosaic`, and like those add-on commands, their availability at the console is not mirrored in R Markdown. You'll have to "source" it directly in your .Rmd file in order to be able to use it there.

As described in class, what `sampleAndRegress()` does is draw sample data (sample size $n = 18$, by default) from a population where the two variables, X and Y, have the true linear relationship

$$Y = \beta_1 X + \beta_0 + \epsilon,$$

with $\epsilon \sim \mathsf{Norm}(0, \sigma)$; default values (go ahead and use them) for the paramaters are $\beta_0 = -17$, $\beta_1 = 2.5$, and $\sigma = 8$. Having drawn a sample of X and Y values, it computes, from the sample, various regression-related quantities; output from the command the function includes only these values, not the sampled data points $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$.

In class, I drew not just one, but 5000 bivariate samples using

```
manyRuns <- do(5000) * sampleAndRegress()
```

I used only the `b0` and `b1` values (my purpose was to show the envelope of different best-fit lines the sampling process can yield), but other values from the output come into play for the questions that follow.

1. You have seen `sampleAndRegress()` can be run singly, or repeatedly in conjunction with a `do()` command. When run singly, its output includes multiple columns: `b0`, `b1`, etc. There is code, very much like the code you used in Part 1 of this lab, that precedes the computation of each of these columns/variables, code like this:

```
myLabData <- data.frame(
  x = rnorm(18, 30, 6),
  y = 2.5 * xdata - 17 + rnorm(18, 0, 8)
)
```

```
modelResult <- lm(y ~ x, data=myLabData)
```

The numbers computed by `sampleAndRegress()` could be computed directly from `modelResult` by other means. Indeed, each one of these computations is included somewhere in the output of these commands:

```
summary(modelResult)
anova(modelResult)
```

Run these lines of code, and then, one by one, identify where each of the quantities `b0, b1, SSModel, SSResiduals, SSTotal, sigmaHat, SE.b0`, and `SE.b1` are reported. (Another name for `sigmaHat` is "Residual standard error".)

Note: For the remaining questions, it is presumed that you have run the command above to produce `manyRuns`, and that your answers (lab results) come directly from the values stored therein, not from any further use of `lm()`, `summary()`, or `anova()`.

2. The command

   ```
   manyRuns[1,]
   ```

   gives results from calculations on the first (the first out of 5000 of them) draw of (x, y)-pairs. In your write-up, display the contents of manyRuns[1,], then use the relevant numbers to generate a 95% confidence interval for β_1. Since we manufactured our data using code like that found in Problem 1, we happen to know β_1. (It is 2.5, by default.) Is β_1 inside your interval?

3. Each row of `manyRuns` has data which can be used to construct a confidence interval in much the same way as you used the first row in Problem 2 to construct a 95% CI. Use the `filter()` command to pick out those rows whose corresponding 90% confidence intervals contain β_1. Divide that count by 5000 to find and report the **effective coverage rate**. What did you expect this effective coverage rate to be? Does it match your expectations?

   The relevant filter command might have a skeletal appearance like this:

   ```
   filter(manyRuns, ... <= 2.5 & 2.5 <= ... )
   ```

4. You can use the command

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

   to view a distribution of b_1 values. What would be an appropriate name for this distribution? Use a command on this distribution to estimate the standard error of b_1.

5. Each row of `manyRuns` contains a value for SE_{b_1}, estimated using the sample data from which all values in that row are computed. These are all estimates, computed using a formula, of the number you estimated in the last part. What are the smallest and largest estimates of SE_{b_1} in this column? These estimates were what you used to construct 5000 90% CIs in Problems 3. Is it surprising to you, given the corresponding range in SE_{b_1} estimates, that the effective coverage rate is what you found it to be in Problem 3?

6. Under the assumptions of the simple linear model (SLM),

$$Y = \beta_1 X + \beta_0 + \epsilon,$$

where the residuals, represented by ϵ, have a $\mathsf{Norm}(0, \sigma)$ distribution. In each of the runs/rows of `manyRuns`, the number `sigmaHat` was computed as an approximation of σ using sample data. Investigate your `sigmaHat` numbers carefully, and report whether it seems `sigmaHat` is an unbiased estimator of σ. Support your conclusion.