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

InkjetPrinters
expl.: PPM
resp. Price

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(Pri~~s~~ce ~ 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

For RestaurantTips

PctTip ~ Bill

have

estimate of slope $b_1 = 0.04881$

est. of $SE_{b_1} = 0.02871$

- depends on level of confidence
- In R, get using qt( )

Conf. Int. for β_1 has

lower bound $= b_1 - \underbrace{(\quad\quad)}_{\text{critical value}}(SE_{b_1})$

upper bound $= b_1 + (\quad\quad)(SE_{b_1})$

For 95% confidence

$t^* = qt(\underline{0.975}, df =$

$df = n - 1 - \#(\text{of predictor variables})$

$= n-2$ for single-predictor (simple) linear regression.

sampleAndRegress( )

1. Samples 18 coordinate pairs (x, y) from a population where

$$Y = \beta_1 X + \beta_0 + \varepsilon, \quad \boxed{\begin{aligned}\beta_0 &= -17 \\ \beta_1 &= 2.5\end{aligned}}$$

- linear relationship holds
- residuals $\varepsilon \sim \text{Norm}(0, \sigma)$

2. Computes various statistics

```
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")
```

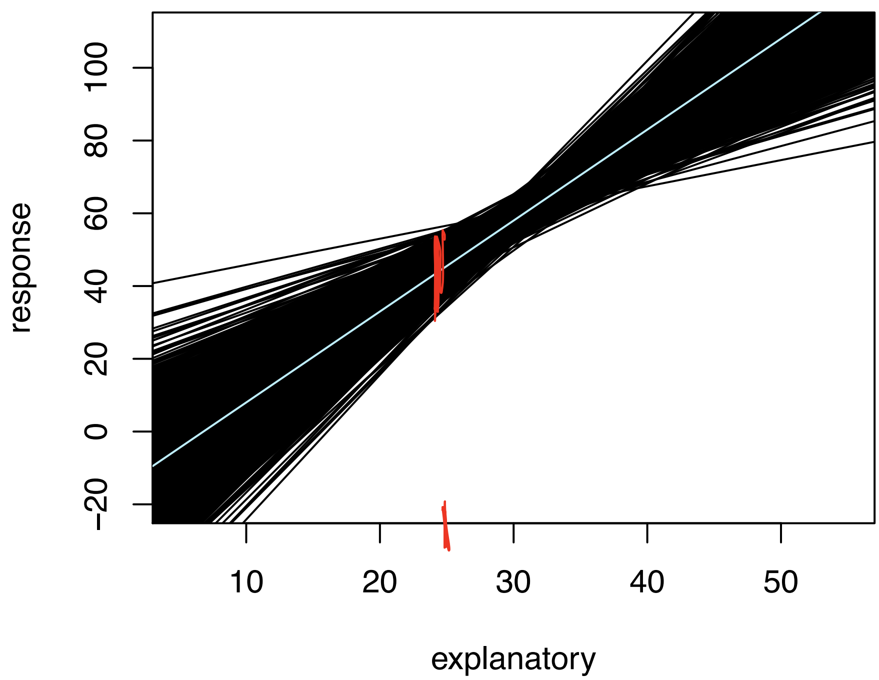
```
sampleAndRegress <-
  function(
    sigma = 8,
    sampleSize = 18,
    beta0 =  -17,
    beta1 = 2.5
  ) {

    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)
    )
  }
```

```
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")
```



estimate y at $X = 25$?

In past: plug $x = 25$ into the regression line

$b_0 + b_1 X$ (gives a point estimate)

Prediction and confidence intervals

If the conditions for the simple linear model are met, and if we have rejected the null hypothesis in the Model Utility Test in favor of the alternative, that the explanatory variable has some usefulness as a predictor of values of the response variable, it is typical to see the model used that way. There are two sorts of *prediction*-type questions we might ask.

1. What is the average response Y at a particular X? We denote this number by $\mu_Y(X)$, which is a parameter specific to the subpopulation of response one can see for that particular value of X.
2. What is the *next* response Y I expect to see for a particular X?

Example. For predicting `Price` of an inkjet printer from its `PPM`, we have the coefficients

```
lm(Price ~ PPM, data=InkjetPrinters)
```

```
##
## Call:
## lm(formula = Price ~ PPM, data = InkjetPrinters)
##
## Coefficients:
## (Intercept)          PPM
##      -94.22        90.88
```

which we express as a linear model

$$\widehat{\text{Price}} = -94.22 + 90.88(\text{PPM}).$$

The best *single number* to

1. estimate the average price for an inkjet printer that prints 3.5 pages per minute is

   $$-94.22 + 90.88(3.5) \;=\; 223.86,$$

   or \$223.86.
2. estimate the price of the next inkjet printer that prints 3.5 pages is, likewise, $-94.22+90.88(3.5) = 223.86$.

But this number is most likely wrong, as it merely *estimates* answers to these questions. We would prefer to give an interval of values, along the lines of a confidence interval,

$$(\text{point estimate}) \pm (\text{margin of error}).$$

As one might expect, the margin of error is larger when predicting the *next* response value at X than it is for the *average* response. Formulas are available for these two margins of errors, but they are ugly. (You can find them incorporated into the interval formulas in the box on p. 553.)

We will use software to generate these intervals, and the easiest approach I know in R is to use the (stored) model to make an *estimator* function:

```
lmResult <- lm(Price ~ PPM, data=InkjetPrinters)
printerPriceEstimator <- makeFun( lmResult )
```

creation step

The resulting function `printerPriceEstimator()` (it seemed an appropriate name, given the situation), can be used to repeat my calculation of a single-number estimate above:

```
printerPriceEstimator(PPM = 3.5)
```

```
##        1
## 223.8515
```

It can also be used to generate a **confidence interval for the mean response** when `PPM` equals 3.5, a better answer to Question 1 than any single number can be:

anova(lm(resp ~ expl., data=_))
summary(   )

create this function
call it by any name
makeFun( )

```
printerPriceEstimator(PPM = 3.5, interval="confidence")
```

```
##        fit      lwr      upr
## 1 223.8515 184.5706 263.1324
```

Finally, the same estimator function, with switch `interval="prediction"`, gives an interval that responds to Question 2, known as a **prediction interval for a response value** at a given choice of the explanatory variable.

```
printerPriceEstimator(PPM = 3.5, interval="prediction")
```

```
##        fit      lwr      upr
## 1 223.8515 94.73146 352.9715
```

Not surprisingly, both the confidence interval and the prediction interval are centered at 223.85 (the single-number estimate), but the prediction interval is (much) wider. For both answers, the level of confidence was 95%. At the time of writing this, I do not know how to set the level of confidence to some other value.

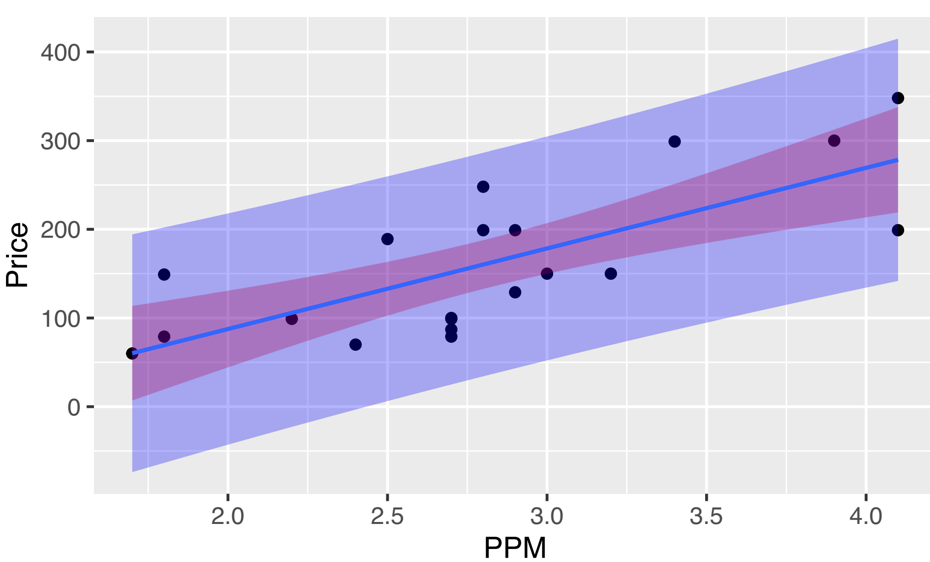
You can add a switch "`level =`" to change the coverage rate. Without that switch, the commands above sought 95% coverage. The next command would produce a 90% prediction interval

```
printerPriceEstimator(PPM = 3.5, interval="prediction", level = 0.9)
```

```
##        fit      lwr      upr
## 1 223.8515 117.2781 330.4248
```

It can be instructive to envision confidence and prediction intervals spread out around the regression line.

```
gf_point(Price ~ PPM, data=InkjetPrinters) %>%
  gf_lm(interval="confidence", fill="red") %>%
  gf_lm(interval="prediction",fill="blue")
```



Exercise

Use commands like those above to both a confidence interval for the mean response, and a prediction interval when `PPM = 3.0`. Compare your answers to those given in Example 9.19 on p. 554.
