Regression Models (Part XII : Regression Inference)

A. Getting a confidence interval

Recall from your inference class, a fair number of confidence intervals take the form of an estimate plus or minus a t quantile times a standard error. Let's use that formula to create confidence intervals for our regression parameters. Let's first do the intercept.

```
library(UsingR); data(diamond)
y <- diamond$price; x <- diamond$carat; n <- length(y)
beta1 <- cor(y, x) * sd(y) / sd(x)
beta0 <- mean(y) - beta1 * mean(x)
e <- y - beta0 - beta1 * x
sigma <- sqrt(sum(e^2) / (n-2))
ssx <- sum((x - mean(x))^2)
seBeta0 <- (1 / n + mean(x) ^ 2 / ssx) ^ .5 * sigma
seBeta1 <- sigma / sqrt(ssx)
tBeta0 <- beta0 / seBeta0
tBeta1 <- beta1 / seBeta1
pBeta0 <- 2 * pt(abs(tBeta0), df = n - 2, lower.tail = FALSE)
pBeta1 <- 2 * pt(abs(tBeta1), df = n - 2, lower.tail = FALSE)
coefTable <- rbind(c(beta0, seBeta0, tBeta0, pBeta0), c(beta1, seBeta1, tBeta
1, pBeta1))
colnames(coefTable) <- c("Estimate", "Std. Error", "t value", "P(>|t|)")
rownames(coefTable) <- c("(Intercept)", "x")
coefTable
##              Estimate Std. Error   t value      P(>|t|)
## (Intercept) -259.6259   17.31886 -14.99094 2.523271e-19
## x           3721.0249   81.78588  45.49715 6.751260e-40
fit <- lm(y ~ x);
summary(fit)$coefficients
##              Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) -259.6259   17.31886 -14.99094 2.523271e-19
## x           3721.0249   81.78588  45.49715 6.751260e-40
sumCoef <- summary(fit)$coefficients
sumCoef[1,1] + c(-1, 1) * qt(.975, df = fit$df) * sumCoef[1, 2]
## [1] -294.4870 -224.7649
```

Now let's do the slope:

```
(sumCoef[2,1] + c(-1, 1) * qt(.975, df = fit$df) * sumCoef[2, 2]) / 10
## [1] 355.6398 388.5651
```

So, we would interpret this as: "with 95% confidence, we estimate that a 0.1 carat increase in diamond size results in a 355.6 to 388.6 increase in price in (Singapore) dollars".

B. Prediction of outcomes

Finally, let's consider prediction again. Consider the problem of predicting Y at a value of X. In our example, this is predicting the price of a diamond given the carat.

We've already covered that the estimate for prediction at point x_0 is:

$$\hat{\beta}_0 + \hat{\beta}_1 x_0$$

A standard error is needed to create a prediction interval. This is important, since predictions by themselves don't convey anything about how accurate we would expect the prediction to be. Take our diamond example. Because the model fits so well, we would be surprised if we tried to sell a diamond and the offers were well off our model prediction (since it seems to fit quite well).

There's a subtle, but important, distinction between intervals for the regression line at point x_0 and the prediction of what a y would be at point x_0. What differs is the standard error:

For the line at x_0 the standard error is,

$$\hat{\sigma} \sqrt{\frac{1}{n} + \frac{(x_0 - \overline{X})^2}{\sum_{i=1}^{n} (X_i - \overline{X})^2}}$$

For the prediction interval at x_0 the standard error is

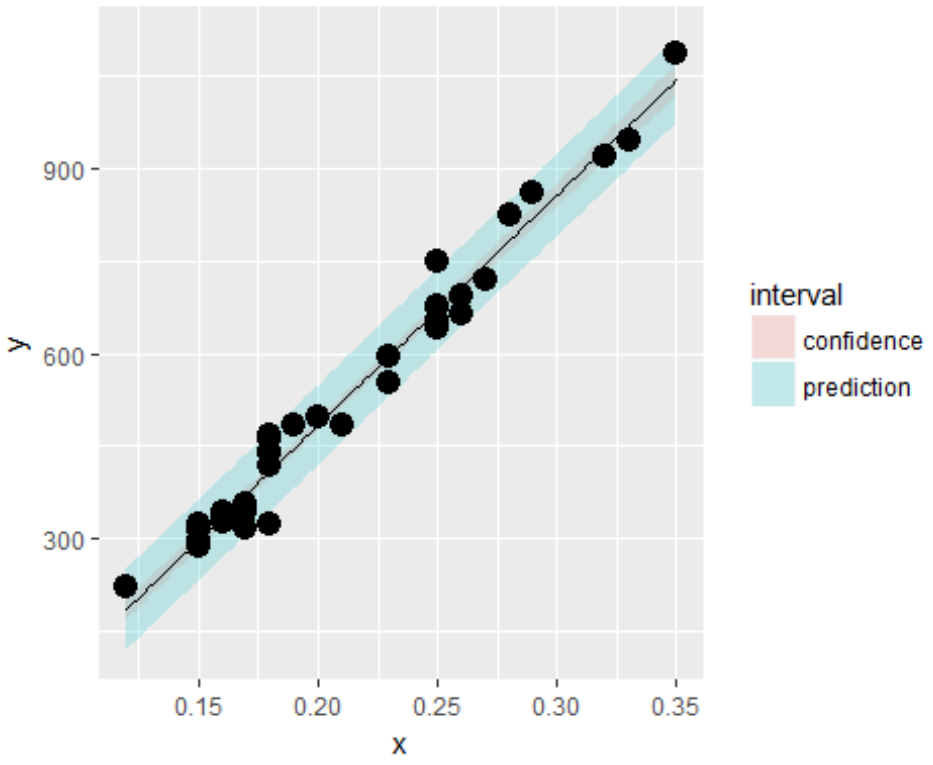
$$\hat{\sigma} \sqrt{1 + \frac{1}{n} + \frac{(x_0 - \overline{X})^2}{\sum_{i=1}^{n} (X_i - \overline{X})^2}}$$

Notice that the prediction interval standard error is a little large than error for a line. Think of it this way. If we want to predict a Y value at a particular X value, and we knew the actual true slope and intercept, there would still be error. However, if we only wanted to predict the value at the line at that X value, there would be no variance, since we already know the line.

Thus, the variation for the line only considers how hard it is to estimate the regression line at that X value. The prediction interval includes that variation, as well as the extra variation unexplained by the relationship between Y and X. So, it has to be a little wider.

For the diamond example, here's both the mean value and prediction interval. (code and plot). Notice that to get the various intervals, one has to use one of the options interval="confidence" or `interval="prediction"` in the `prediction` function.

```
library(ggplot2)
newx = data.frame(x = seq(min(x), max(x), length = 100))
p1 = data.frame(predict(fit, newdata= newx,interval = ("confidence")))
p2 = data.frame(predict(fit, newdata = newx,interval = ("prediction")))
p1$interval = "confidence"
p2$interval = "prediction"
p1$x = newx$x
p2$x = newx$x
dat = rbind(p1, p2)
names(dat)[1] = "y"
g = ggplot(dat, aes(x = x, y = y))
g = g + geom_ribbon(aes(ymin = lwr, ymax = upr, fill = interval), alpha = 0.2
)
g = g + geom_line()
g = g + geom_point(data = data.frame(x = x, y=y), aes(x = x, y = y), size = 4
)
g
```



C. Summary notes

- Both intervals have varying widths. +Least width at the mean of the Xs.
- We are quite confident in the regression line, so that interval is very narrow. +If we knew β_0 and β_1 this interval would have zero width.
- The prediction interval must incorporate the variability in the data around the line. +Even if we knew β_0 and β_1 this interval would still have width. *